MECHATRONICS: FUNDAMENTALS, DESIGN, INTEGRATION, AND VALIDATION



Guoming Zhu
Michigan State University



Mechatronics: Fundamentals, Design, Integration, and Validation

Guoming Zhu

Michigan State University

This text is disseminated via the Open Education Resource (OER) LibreTexts Project (https://LibreTexts.org) and like the thousands of other texts available within this powerful platform, it is freely available for reading, printing, and "consuming."

The LibreTexts mission is to bring together students, faculty, and scholars in a collaborative effort to provide an accessible, and comprehensive platform that empowers our community to develop, curate, adapt, and adopt openly licensed resources and technologies; through these efforts we can reduce the financial burden born from traditional educational resource costs, ensuring education is more accessible for students and communities worldwide.

Most, but not all, pages in the library have licenses that may allow individuals to make changes, save, and print this book. Carefully consult the applicable license(s) before pursuing such effects. Instructors can adopt existing LibreTexts texts or Remix them to quickly build course-specific resources to meet the needs of their students. Unlike traditional textbooks, LibreTexts' web based origins allow powerful integration of advanced features and new technologies to support learning.



LibreTexts is the adaptable, user-friendly non-profit open education resource platform that educators trust for creating, customizing, and sharing accessible, interactive textbooks, adaptive homework, and ancillary materials. We collaborate with individuals and organizations to champion open education initiatives, support institutional publishing programs, drive curriculum development projects, and more.

The LibreTexts libraries are Powered by NICE CXone Expert and was supported by the Department of Education Open Textbook Pilot Project, the California Education Learning Lab, the UC Davis Office of the Provost, the UC Davis Library, the California State University Affordable Learning Solutions Program, and Merlot. This material is based upon work supported by the National Science Foundation under Grant No. 1246120, 1525057, and 1413739.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation nor the US Department of Education.

Have questions or comments? For information about adoptions or adaptions contact info@LibreTexts.org or visit our main website at https://LibreTexts.org.

The LibreTexts name, logo, and book covers are trademarked by LibreTexts, Inc. (a 501(c)(3) not-for-profit organization) and protected against unauthorized use.

This text was compiled on 08/04/2026

TABLE OF CONTENTS

Licensing

Acknowledgements

1: Introduction

1.0: Prelude

1.1: Introduction of Mechatronic Systems

1.2: Introduction of Mechatronic Systems

1.3: How to Use This Book

1.4: Chapter References

2: Mechatronic System Modeling

2.0: Prelude

2.1: Laplace Transforms

2.2: Mechanical System Modeling

2.3: Electrical and Electronic System Modeling

2.4: Mechatronic System Modeling

2.5: System Linearization

2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning

2.7: Chapter References

3: System Responses and Controls

3.0: Prelude

3.1: Impulse, Step, and Sinusoidal Responses

3.2: System Stability and Its Performance Measures

3.3: The First and Second Order System Responses and Associated Pole Locations

3.4: PID Control and Its Gain Tuning Methods

3.5: Review of Model-Based Control Design and Gain Tuning Methods

3.6: Machine Learning-Based Control

3.7: Chapter References

4: Signal Processing

4.0: Prelude

4.1: Analog Signals and Their Frequency Analysis

4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)

4.3: Digital Signals and Their Frequency Analysis

4.4: Analog and Digital Filter Design

4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters

4.6: Pulse-Width and Frequency Modulated Signals

4.7: Advanced Analog and Digital Filter Design and Implementation

4.8: Chapter References

5: Mechatronic System Component- Actuators

5.0: Prelude

5.1: Mechatronic Actuator Background

5.2: Mechatronic Actuator Background

5.3: DC-Motor Drive Circuits

5.4: Discrete Electromagnetic Valve Actuation (EMVA)

5.5: Discrete Electrohydraulic Valve Actuation (EHVA)

5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control

5.7: Chapter References

6: Mechatronic System Component- Sensors

6.0: Prelude

6.1: Mechatronic Sensor Background

6.2: Rotational Speed Sensors and Their Speed Calculation

6.3: Acceleration Sensors

6.4: Angular Velocity Sensors (Rate Gyro)

6.5: Sensor Fusion

6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing

6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing

6.8: Vision-Based Sensing

6.9: Chapter References

7: Mechatronic Control Systems

7.0: Prelude

7.1: Mechatronic System Architecture

7.2: Microcontrollers Used for Mechatronic Systems

7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi

7.4: Mechatronic Control System Architecture- Speed Control Example

7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process

7.6: Chapter References

8: Mini-Segway System — Mechatronic System Example I

8.0: Prelude

8.1: Mini-Segway System Modeling

8.2: Model-Free Dual-PID Mini-Segway Control System Architecture

8.3: Model-Based Mini-Segway Control

8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input

8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller

8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers

8.7: Experimental Study of the Mini-Segway Model-Free Control

8.8: Model-Based Control Experiments

8.9: Chapter References

9: Three-Wheel Cart - Mechatronic System Example 2

9.0: Prelude

9.1: Three-Wheel Cart System Modeling

9.2: Dual- and Triple-PID Control of a Three-Wheel Cart

9.3: Model-Based Control of a Three-Wheel Cart

9.4: Simulation Study of PID and Model-Based Control Strategies

9.5: Experimental Investigation of PID Control Strategies

9.6: Experimental Investigation of Model-Based Control Strategy

9.7: Chapter References

10: Appendix A - MATLAB and Simulink Software

11: Appendix B - Mechatronics Lab Sessions

11.0: MATLAB/Simulink Installation

11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink

11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)

11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi

11.4: DC-Motor Control Utilizing Arduino Hardware

11.5: Simulink GUI (Graphic User Interface) Design and Programming

11.6: Sensor Fusion vs. Single Sensor

11.7: PID Tuning for Mini-Segway Simulink Model

11.8: PID Tuning for Mini-Segway Hardware System

11.9: Mini-Segway Stabilization and Drive Project

Index

Glossary

Glossary A

Detailed Licensing

This page titled Mechatronics: Fundamentals, Design, Integration, and Validation (Zhu) is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu.

Licensing

A detailed breakdown of this resource's licensing can be found in ***Back Matter/Detailed Licensing***.

This page titled Licensing is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

Acknowledgements

This textbook is a product of Co-DREAM OER (Collaborative Development of Robotics, Mechatronics, and Advanced Manufacturing Open Educational Resources), an initiative funded by the U.S. Department of Education to develop open educational resource textbooks on robotics, mechatronics, and advanced manufacturing processes. The texts are written for students enrolled in 2-year associate's, 4-year bachelor's, and graduate-level courses. This specific text has been created by a team of scholars, students, support staff, and other professionals from across the country. It is intended for mechatronics courses for 4-year bachelor's and graduate-level programs.

The Co-DREAM OER team extends its sincere gratitude to Dean Christopher Cox of Clemson University's Libraries and Dean Anand Gramopadhye of Clemson University's College of Engineering, Computing and Applied Sciences. They envisioned the project and provided the essential administrative leadership that enabled its realization. The team also recognizes the leadership of Dr. Yang Wu, Open Resources Librarian at Clemson University Libraries. He has been the Principal Investigator for the project. Yang's efforts translated the project's vision into a successful grant proposal. He then guided a multidisciplinary, multi-institutional team in the development of the textbooks. In addition, the project acknowledges the valuable administrative support of Dr. Yue Wang, Associate Dean of Research at Clemson's College of Engineering, Computing and Applied Sciences; and Elias Tzoc Caniz, Associate Dean for Teaching, Learning, and Research at Clemson University Libraries. Their guidance was instrumental throughout the project.

The project acknowledges the development of the mini-Segway mechatronic system used in the book by Dr. Yingxu Wang, a former doctoral student at Michigan State University. David Benkes-Toth, a former Michigan State undergraduate student, contributed to the development of the three-wheel cart mechatronic system. The project also thanks Oluwadara Abimbade, Digital Learning Strategist at Clemson Online for providing instructional design input for the book. Co-DREAM OER extends special thanks to Matt Swank, Senior Art Director, Marketing and Communications; Walker Massey, Digital Accessibility Specialist; and Dan Lewis, Director of Digital Accessibility at Clemson University for making the book compliant with the Americans with Disabilities Act. The project further recognizes the important contributions of copy editor Robert Weinstein, and graphic designer and typesetter Kim Scott, in the creation of the print textbook.

Lastly, the project expresses gratitude to the U.S. Department of Education's Open Textbook Pilot Program for making this work possible.

Dr. Guoming (George) Zhu, Ph.D. (Author)

Gardner Endowed Chair and Professor

Department of Mechanical Engineering

Michigan State University

This page titled Acknowledgements is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

1: INTRODUCTION

Learning Objectives

- Describe engineering disciplines involved in mechatronic systems including mechanical, electric/electronic, computer, and control systems.
- Provide the overall contents included in this book along with descriptions of chapter-specific contents.
- Because this book is written for both undergraduate- and graduate-level classes, recommend contents to be used for these different levels.

This chapter provides an introduction to mechatronic systems (including general system architecture, technical disciplines involved, and sample mechatronic systems), an overview of this book, and recommendations of how to use this book.

1.0: Prelude

1.1: Introduction of Mechatronic Systems

1.2: Introduction of Mechatronic Systems

1.3: How to Use This Book

1.4: Chapter References

This page titled 1: Introduction is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

1.0: Prelude

Mechatronic systems can be found everywhere, including manufacturing factories, airplanes, automobiles, appliances, and our homes. A mechatronic system consists of three subsystems: mechanical, electrical/electronic, and mechatronic. A microcontroller (or multiple microcontrollers) is often used to collect the sensor and reference signals from a mechatronic system, make decisions based upon the signals collected inside the microcontroller, convert these decisions into actuator commands, and send these decisions to the related actuators for execution in a closed-loop manner.

Mechatronic subsystems often convert electrical/electronic signals to mechanical motions (e.g., solenoid valve and DC motor) and mechanical motions to electrical/electronic signals (e.g., automobile wheel speed sensor). Therefore, studying characteristics of mechatronic systems requires knowledges from multiple engineering disciplines. That is, a mechatronics system is an integration of control, computer, electric/electronic, and mechanical systems. In most US universities, the computer, electric/electronic, and mechanical engineering departments are independent within the college of engineering. There are not many multidisciplinary courses available. Thus, mechatronics-related courses are important because research and industry communities need people with multidisciplinary backgrounds.

As shown in Figure 1.1, mechatronic systems require knowledge from four main engineering subjects: control system engineering, computer (microcontroller) engineering, electric/electronic engineering, and mechanical engineering. Interdisciplinary knowledge, such as control electronics between control systems and electric/electronic engineering, is also necessary. Most students, especially mechanical engineering students, lack certain knowledge needed to master mechatronic systems. For example, mechanical engineering students may lack knowledge of digital control system and control electronic hardware knowledge; these gaps need to be addressed in the mechatronic system course.

This book tries to fill these gaps by providing necessary backgrounds in control electronics, digital control systems, electromechanical systems (especially integration of control systems), electric/electronic systems, microcontroller systems, and mechanical systems. In addressing these areas, the book introduces the concept of integrated mechatronic systems. Future engineering graduates will need to have multidisciplinary engineering backgrounds to meet the requirement of engineering research and industrial research/development. For instance, in industrial research and development, system engineers are needed who can integrate a mechatronic system by providing specifications for control electronics, digital control systems, and electromechanical systems.

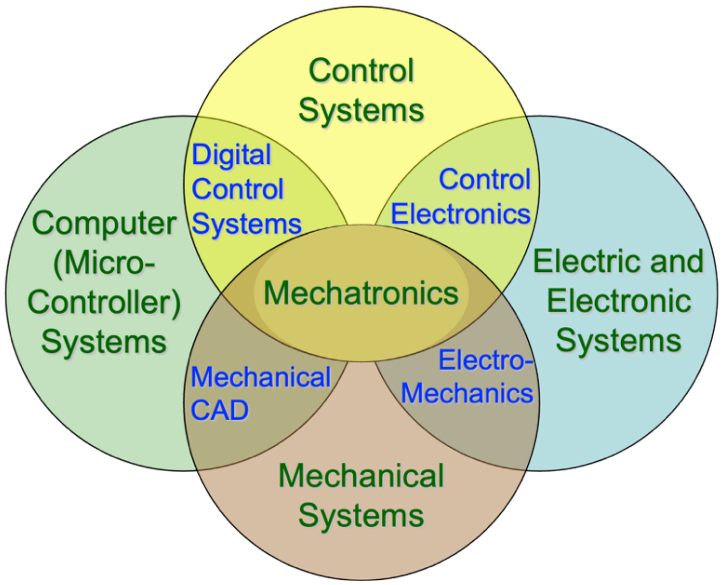


Figure 1.1: Engineering disciplines associated with mechatronic systems.

This chapter is organized as follows. Section 1.1 will discuss the concept of information and physical domains of a mechatronic system, along with examples of mechatronic systems in our daily life—from the passenger vehicles we drive to air conditioning and furnaces used in our home. Section 1.2 outlines the book chapter-by-chapter. Section 1.3 then provides a guideline for how to use this book for both undergraduate and graduate level classes.

Orienting Questions

1. What engineering disciplines are involved in mechatronic systems?
2. What is the general system architecture of a mechatronic system?
3. What are the overall contents of this book?
4. What content is associated with undergraduate and graduate level courses?

This page titled 1.0: Prelude is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

1.1: Introduction of Mechatronic Systems

Learning Objectives

After completing this section, you will be able to:

1. Describe the system architecture of a mechatronic system consisting of both information and physical domains as well as the interface between the two domains.
2. Identify the engineering disciplines involved in mechatronic systems.
3. List a few examples of mechatronic systems.

A mechatronic system normally consists of two major domains with an interface intertwined, as shown in Figure 1.2. The digital information domain mainly processes the information from the system sensors and GUI (graphic user interface). It generates the control command signals that the mechatronic system needs to response to the change of system behaviors. The physical domain of a mechatronic system is related to its actuators, physical plant, and sensors.

The interface layer between the digital information and physical domains connects the two domains properly. The sensor anti-aliasing filters (see Chapter 4 for details) make sure that the data sampled by the information domain does not contain any aliasing frequencies. The electric power amplifiers convert the analog (or digital) control signals up to the required power level so that the actuators (such as a DC motor) can be controlled.

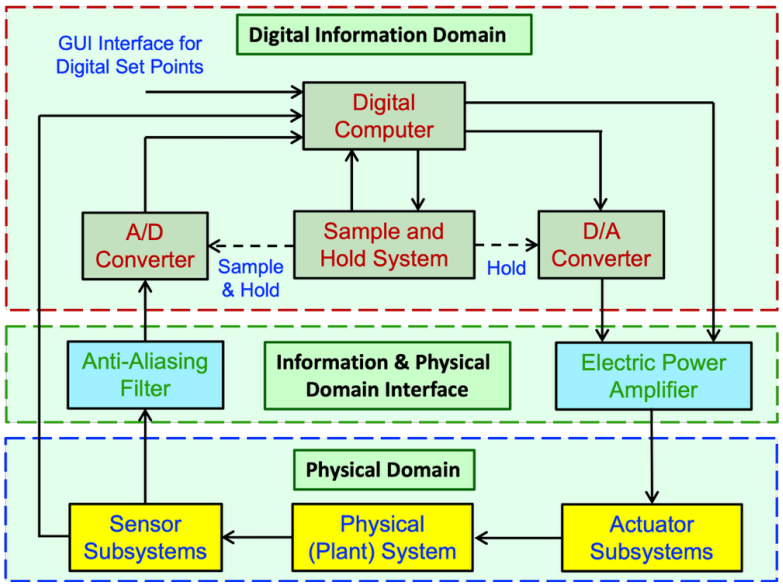


Figure 1.2: Information and physical domain with their interface layer.

Note in Figure 1.2 that there are two paths for the sensors signals to be sampled into the microcontroller (information domain). One is the direct digital path for digital sensors. The other is the A/D (analog-to-digital) conversion path for analog sensors.

Many sensors—such as hall-effect position and speed sensors—are digital in nature; see reference [1] for details. In this case, these sensor signals can be captured directly by the digital input channels of the microcontroller. On the other hand, other sensors are analog. Outputs of these sensor signals—such as analog voltage or current signal—are analog in nature.

One example of an analog sensor is the piezoelectric accelerometer (see both reference [2] and Chapter 2 of this book). It is used to measure the system acceleration, where an analog voltage signal, proportional to the acceleration over certain frequency range, is generated. In this case, the analog voltage signal needs to be converted into a digital signal so that it can be sampled into the microcontroller. Due to the finite sample frequency, an analog anti-aliasing filter is required before the A/D converter in order to eliminate the aliasing frequencies in the digital signal.

Figure 1.2 also shows there are two paths for actuators: digital and analog ones. For those actuators that can be driven by the digital signals directly, the D/A (digital-to-analog) converters are not used. Instead, the digital control signal will be amplified directly to drive them. For instance, a DC motor is often driven by a pulse-width-modulated (PWM) signal that is a square-wave signal with fixed frequency and variable duty-cycle. If the frequency is high enough, the equivalent DC voltage applied to the DC motor can be approximated by PWM duty-cycle multiplying to its peak voltage (see Chapter 5). Because the PWM can be generated by a microcontroller directly, the generated signal can be amplified directly to drive the DC motor.

On the other hand, a DC motor can also be driven by a DC armature voltage. In this case, the digital control command must first be converted into an analog voltage by a D/A converter and then amplified for driving the DC motor. Because there is significant energy loss by driving a DC motor with a DC armature voltage, using the PWM signal is recommended.

Note from Figure 1.2 that there is at least one control system inside a mechatronic system represented by information and physical domains, along the interface layer.

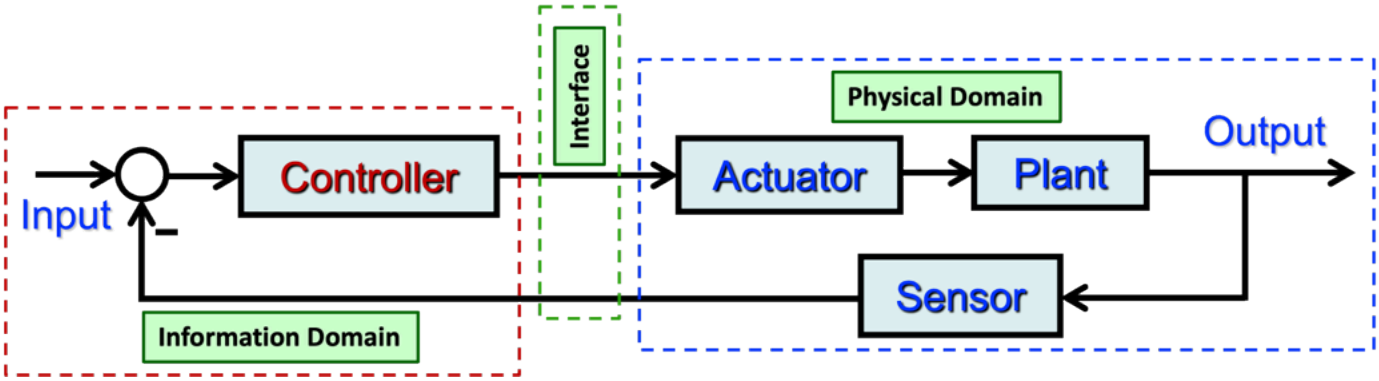


Figure 1.3: A mechatronic system block diagram.

Figure 1.3 shows a mechatronic system in the form of a control system block diagram. Here, the information domain consists of a feedback controller with its outputs to the physical system, a reference input interface, and sensor feedback signals. The physical domain contains the physical plant system along with its actuators and sensors.

A control system block diagram focuses mainly on the mechatronic system dynamics so that the system response performance and stability can be analyzed (see Chapter 3). As a result, the physical domain elements are represented by their information (dynamic characteristics) only. Hence, the interface between the two domains is ignored in the control system block diagram. The control system diagram is easy for system simulations to validate its performance; control strategy can be developed, tuned, and validated within this environment.

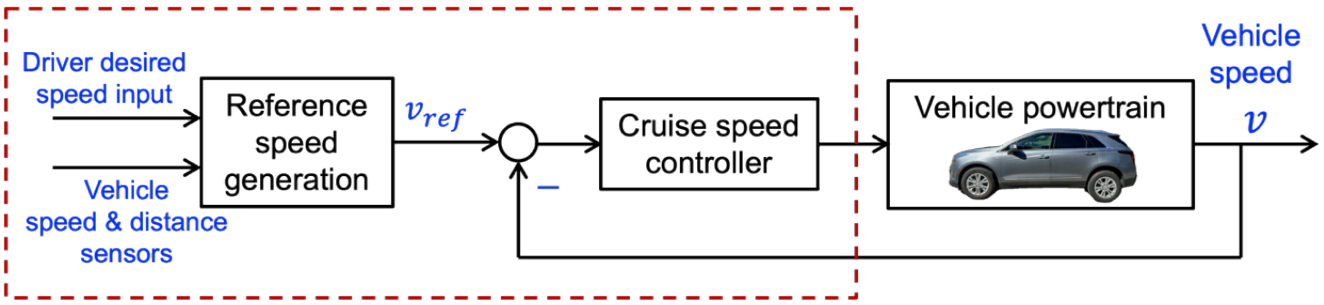


Figure 1.4: Passage vehicle cruise speed control system.

There are many mechatronic systems in our daily life. They are widely used in industries such as automotive, aerospace, medical, defense, material processing, manufacturing, and consumer products. These mechatronic systems can be divided into three groups: a) standalone mechatronic systems, b) mechatronic actuators, and c) mechatronic sensors. In the remainder of this section, we will some of the mechatronic systems typically used in these industries.

Standalone systems

The first group is the standalone mechatronic systems. These consist of a closed-loop control system that performs desired

features. For instance, a passage vehicle cruise speed control system [1], [3], in the automotive industry has a closed-loop control system; see Figure 1.4 for the system block diagram. The components in the red dotted-line box are electronic components whereas the rest are mechanical ones.

In the vehicle cruise speed control system, the driver can set the desired cruise speed for the vehicle. In the case of adaptive cruise control (ACC) [4], [5], the reference speed generation block modifies the cruise reference speed v_{ref}. It reduces it to a safe speed that maintains a safe vehicle distance from the front vehicle based upon measured vehicle speed signals from both vehicles and the distance to the front vehicle. The updated reference vehicle speed v_{ref} is then calculated and used as input to the vehicle cruise speed controller, which normally is a PID (proportional-integral-derivative) controller [6], [7]. With the help of feedback control, the vehicle speed can be maintained at desired speed v_{ref}.

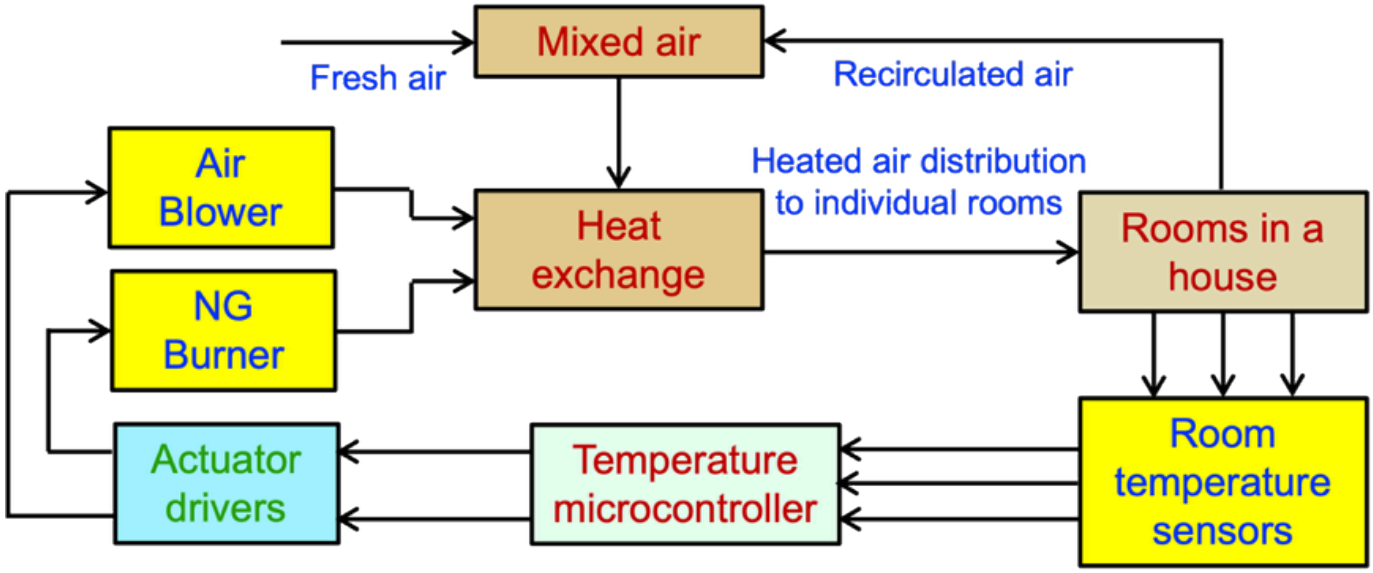


Figure 1.5: Typical home furnace and air conditioning system block diagram.

Home air conditioning and furnace systems [8], called HVAC (heating, ventilation, and air conditioning), are another example of a standalone mechatronic system (see Figure 1.5, where NG represents natural gas). They are used to heat or cool our home and maintain the room temperature at the desired level utilizing either furnace or air conditioning. Furnace systems heat air and distribute the heated air through the house duct system. (By contrast, air conditioning cools air and distributes the cooled air.) Some HVAC systems use a boiler to heat water and provide either hot water or steam for heating. Steam is distributed via pipes to steam radiators. Hot water can be distributed via baseboard radiators, radiant floor systems, or heated air via a coil. Note that a steam boiler operates at a much higher temperature than a hot water boiler and is inherently less efficient. However, high-efficiency versions of all types of furnaces and boilers are currently available.

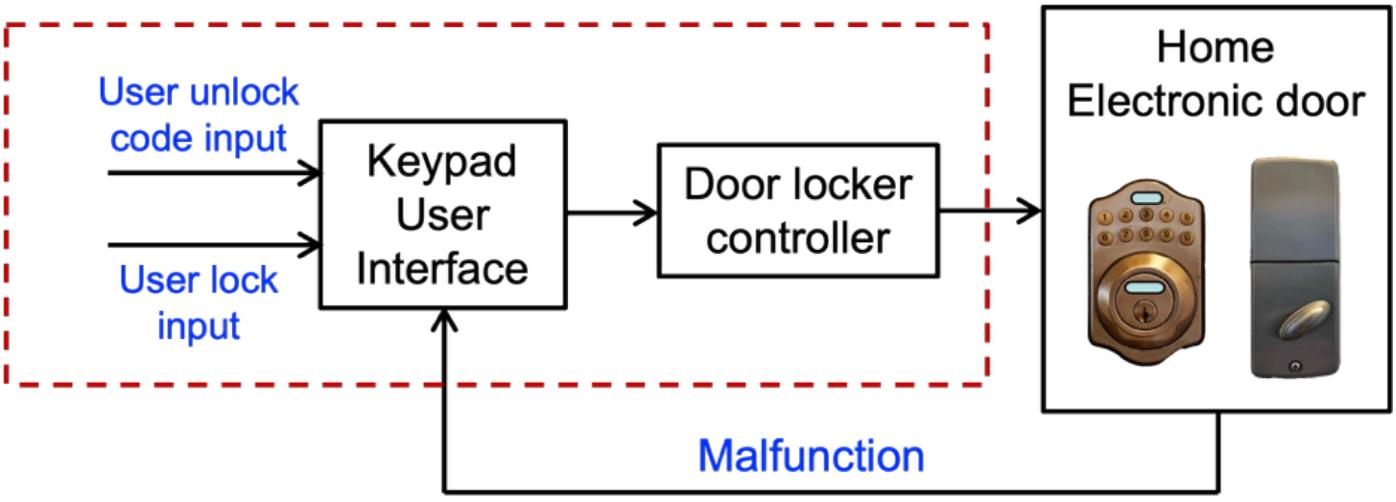


Figure 1.6: Typical home electronic door locker with keypad.

A family of widely used door lockers for the home are now a mechatronic system with an electronic controller inside the red dotted-line box (see Figure 1.6). Users can input the unlock code through the keypad to unlock the door. They can then press a single button on the keypad to lock the door. In this standalone system, the electronic components are the keypad and the microcontroller. The mechatronic part includes the DC motor that moves the locker latch and the position sensor that measures the locker latch position. The mechanical part includes the conventional door locker mechanism.

Comparing this system with the standalone mechatronic systems in Figure 1.4 and Figure 1.5, the electronic door locker mechatronic system in Figure 1.6 is open-loop controlled. That is, the microcontroller sends the one-time control voltage to the DC motor to lock or unlock the door. It will not resend the motor control voltage even if the door is locked or not. If the door locker is jammed, a “beep” sound with a flash on the keypad will occur. As a result, the electronic locker is controlled in open-loop. In contrast, in the passage vehicle cruise speed control mechatronic system seen in Figure 1.4, the vehicle speed is continuously monitored. The driving torque is adjusted based on the monitored vehicle speed to form a closed-loop control system. Concepts of open- and closed-loop control of mechatronic systems will be discussed further in Chapters 3 and 4.

Mechatronic actuators

The second group of mechatronic systems are, in fact, the actuators used in control systems for standalone mechatronic systems to move the mechanical parts. Examples include flow control valves in a chemical processing plant, electric motors used in electric vehicles, and fuel injectors in automotive engines. These actuator converts electrical energy to mechanical movement.

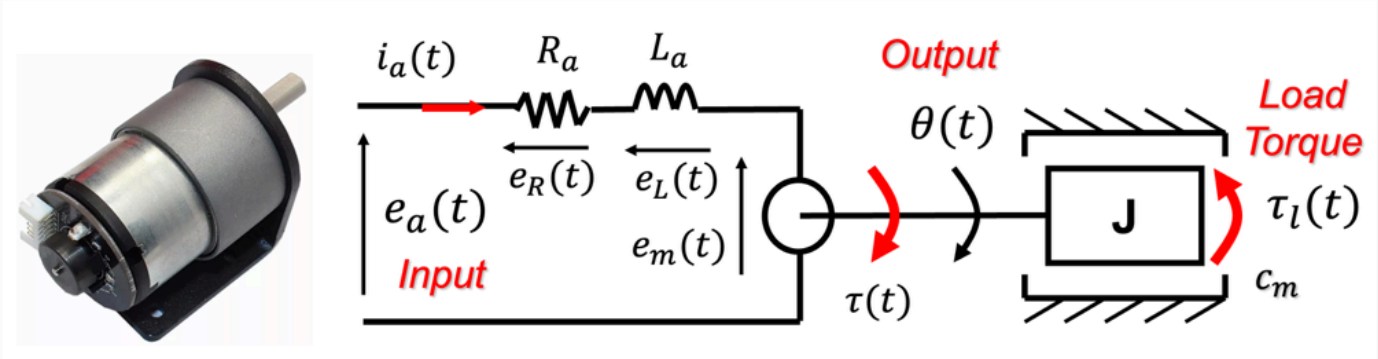


Figure 1.7: A DC-motor actuator.

Figure 1.7 shows a picture and a system diagram for a DC motor, widely used in robotics, that converts the DC electrical energy to rotational torque [6]. The input to the DC motor is the armature and an armature current i_a. The voltage is regulated by a DC power source. The current is generated as a function of the armature winding resistance R_a, the inductance L_a, and the back (or counter) electromotive force (EMF) voltage e_m proportional to the motor speed $\dot{\theta}$.

The generated motor torque $\tau(t)$ is proportional to the armature current $i_a(t)$. Therefore, a DC motor is a mechatronic actuator (device) that converts the command armature voltage e_a (electrical energy) to mechanical torque (mechanical energy). DC motors are widely used in these mechatronic applications with low speed but requiring high torque.



Figure 1.8: An electric variable valve timing actuation system.

The second actuation mechatronic system is an electric variable valve timing (EVVT) actuator for internal combustion engines (see Figure 1.8). The EVVT system consists of a ring gear driven by the engine timing belt through the EVVT pulley, a number of planetary gears connected to the carrier gear that is driven by a permanent magnet electric motor, and a sun gear that is connected directly to the camshaft. Details of the EVVT can be found in references [9] and [10].

The EVVT system is based upon a planetary gear set. The ring gear, which runs at the half speed of the crankshaft, is driven by the crankshaft through the engine timing belt; The ring gear and camshaft are connected through the planetary gear set. The planetary gear carrier is driven by an electric motor and planet gears engaging both the ring and sun gears at the same time, where the sun gear is connected to the camshaft.

To hold the cam-phase at a constant value, the EVVT motor speed should be held to match the ring gear speed. To advance the cam-phase, the EVVT motor speed should be faster than the ring gear speed. To retard the cam-phase, the EVVT motor speed should be slower than the ring gear speed. The EVVT is a mechatronic actuator because it converts the electric energy into the desired mechanical cam shaft phase angle.

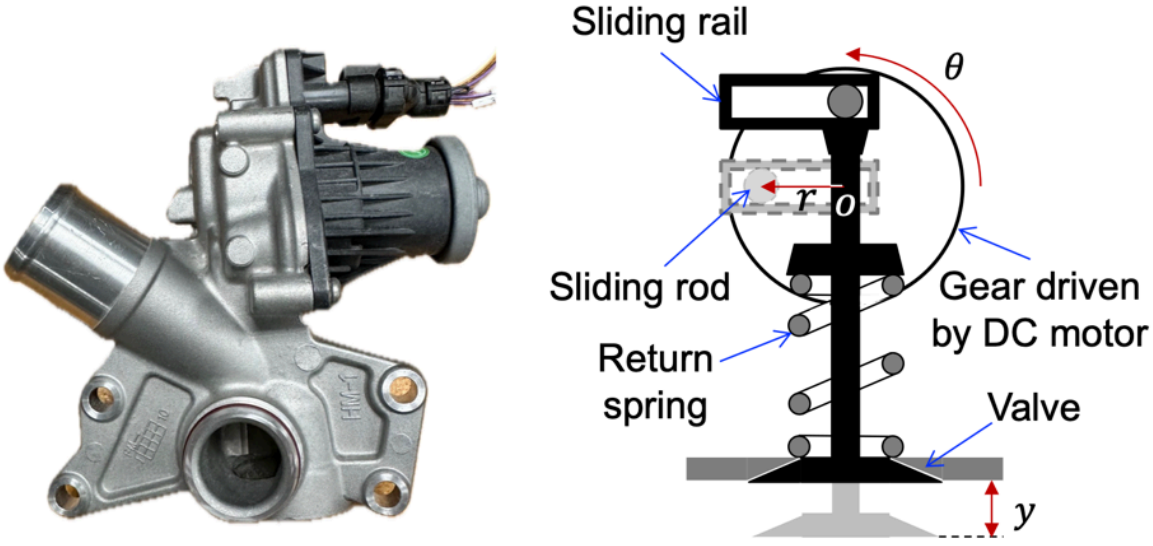


Figure 1.9: An exhaust-gas-recirculation (EGR) valve.

A third example of a mechatronic actuation system is an automotive actuator called an exhaust gas recirculation (EGR) valve [7], [11]. Figure 1.9 shows an architecture of the EGR valve system. Its lift (displacement) is realized through a rotor angular movement

actuated by a DC motor; the valve stem is linked to the rotor through a sliding rail on the valve stem. As the rotor turns in a counterclockwise direction with angle θ, the valve is pushed down by a distance y (marked in Figure 1.9 with dashed-line gray valve position) due to the sliding constraint.

Figure 1.9 also shows the valve movement from its top position (solid-line) to an intermediate one (dashed-line). Therefore, the EGR valve lift can be controlled by the geared DC-motor axle through the slider. As a result, the EGR valve actuator converts electrical energy applied to the DC motor to the desired mechanical valve lift, confirming that it is a mechatronic system by itself. Also, notice that both the engine EVVT and EGR valves use a DC motor to drive the mechanical part to achieve the desired cam-phase and valve lift, respectively.

Mechatronic sensors

The last group of mechatronic systems are mechatronic sensing systems, which convert mechanical variables (motions) into electronic signals to be used in the standalone mechatronic systems. For example, an accelerometer converts the physical acceleration of an object, where the accelerometer is attached, into a DC voltage or a digital number directly. Normally, a closed-loop control system is used in a standalone mechatronic system (see Figure 1.3). The mechatronic sensors are often used to generate the feedback signals inputting to the microcontroller. Therefore, mechatronic sensors are integral parts of mechatronic systems.

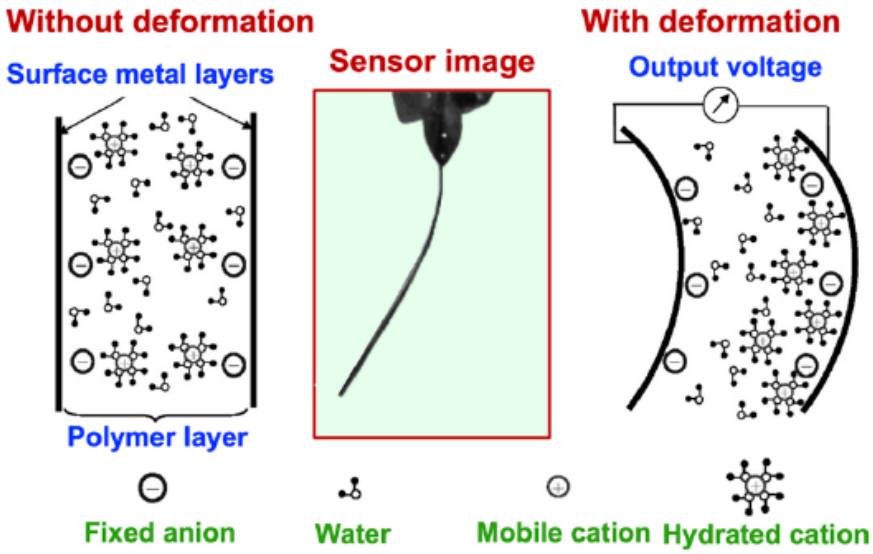


Figure 1.10: A polymer flow sensor.

The first example of a mechatronic sensor is the ionic polymer-metal composite (IPMC) beam flow sensor [12]. Note that IPMC materials have intrinsic sensing and actuation characteristics [13]. As illustrated in Figure 1.10, an IPMC sensor has three layers, with an ion exchange polymer membrane sandwiched by metal electrodes. Negatively charged anions are covalently fixed inside the polymer to polymer chains; these anions are balanced by mobile, positively charged cations [12].

Deformation under a mechanical perturbation redistributes the cations, producing a detectable electric signal (e.g., short-circuit current) that is well correlated with the mechanical stimulus. An IPMC sensor has direct mechanosensory property, which minimizes the complexity in both the sensor construction and its readout circuit. For example, the readout circuit for an IPMC sensor is much simpler than that required for capacitive flow sensing.

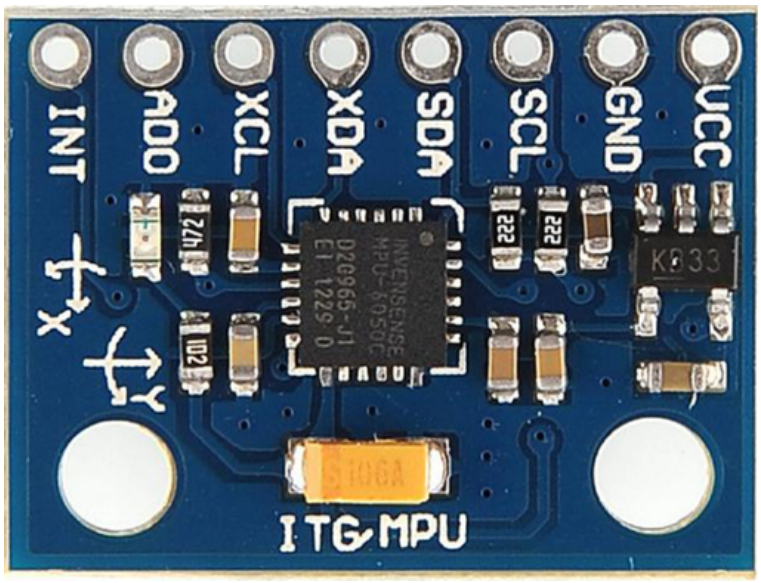


Figure 1.11: An acceleration and rate gyro sensor: MPU 6050.

Another example of a mechatronic sensor is an acceleration and rate gyro sensor: MPU 6050 [14]. As seen in Figure 1.11, the MPU 6050 IMU (inertia measurement unit) has both a three-axis accelerometer and a three-axis gyroscope integrated on a single chip. The gyroscope measures rotational velocity (or rate of change of angular position) over time in three axes of three-dimensional space. It uses MEMS (micro-electro-mechanical systems) technology and the Coriolis effect (force) for measuring. MEMS is the technology of microscopic devices; it incorporates both electronic and moving mechanical parts made up of components between 1 and 100 micrometers in size.

Note that the angular position can be obtained by integrating the angular velocity after the signal bias is filtered. On the other hand, the MPU 6050 accelerometers measure the displacements in a three-dimensional space (three axes) caused by the inertia force in MEMS. With the sensor fusion technology, accurate sensor orientation information can be obtained. Additional accelerometer sensing information can be found in reference [2].

Figure 1.12 shows the last example of a mechatronic sensor: the hall-effect position and speed sensor [16]. The hall-effect sensor is a type of magnetic sensor. It can be used for detecting the strength and direction of a magnetic field produced by a permanent magnet or an electromagnet, with its output varying in proportion to the strength of the magnetic field being detected. Hall-effect sensors are solid state devices. They are becoming more and more popular because they can be used in many different types of applications such as sensing position, velocity, or directional movement.

As illustrated in Figure 1.12, the device can detect rotational speed and direction, where two hall-effect sensors are used to detect the existence of the magnets (blue squares) on the wheel and generate pulse trains A and B. Note that the phase between pulse trains A and B are different when the wheel is rotated clockwise and counter-clockwise. To be more specific, for clockwise rotation, the pulse train A leads pulse train B; for counter-clockwise rotation, the pulse train A lags pulse train B. That is, we can use the phase relationship to find rotational direction. To find the rotational speed, we just need to count the number of pulses for a given time period using either pulse train A or B.

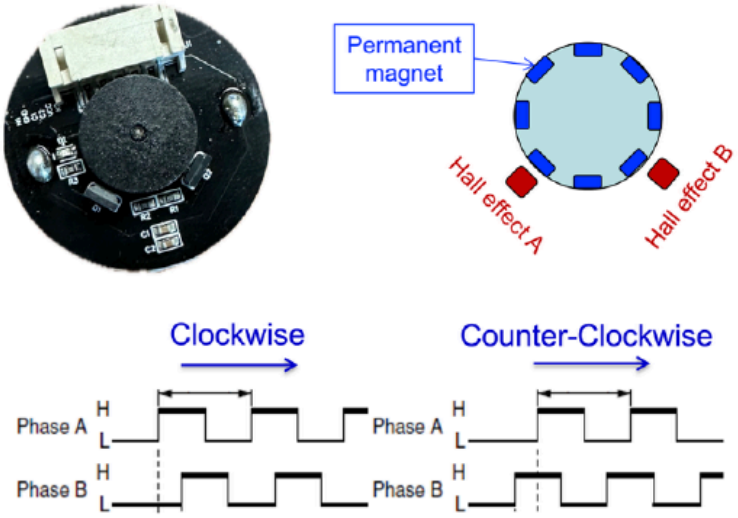


Figure 1.12: A hall-effect position and speed sensor.

More information regarding mechatronic sensors can be found in reference [16]. As a summary, mechatronic system can be standalone systems that perform or complete a task based upon human command or reference signals from an upper-level system. They can also be mechatronic actuators or sensors as part of a standalone mechatronic system. In general, as long as a system or subsystem is involved in converting electronic/electric signals to mechanical movement or vice versa, it is a mechatronic system or subsystem. Mechatronic systems are used in our daily life. In the future, we will have more and more mechatronic systems developed to improve the quality of our lives.

Within each mechatronic system, there is at least one open- or closed-loop control system to perform desired functions. For example, Figure 1.13 shows the hardware architecture of a mechatronic DC-motor speed control system. It consists of a microcontroller (e.g., Arduino MKR 1000 Wi-Fi), a DC-motor drive board (e.g., Arduino MKR motor carrier board), and a DC motor (see Figure 1.7) with a hall-effect speed sensor (see Figure 1.12). Designing such a mechatronic system involves a so-called iterative V-design process, addressed in Chapter 7, from mechatronic system modeling, simulation validation study, prototyping, and design optimization.

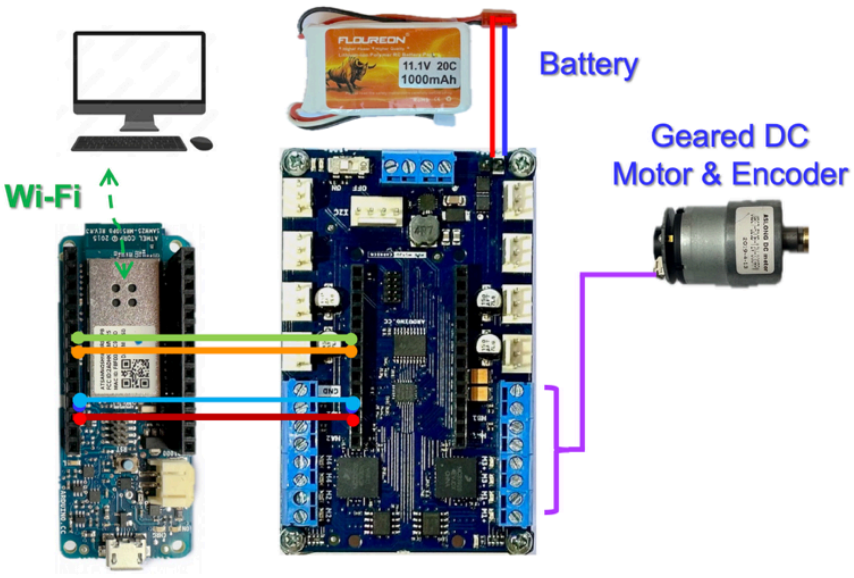


Figure 1.13: Hardware architecture of a DC motor speed control system.

Section Summary

This section provides an overview of mechatronic systems with two main domains—information and physical domains—along with the interface layer between these two domains. Open- or closed-loop control is also a part of mechatronic systems. The feedback control generates commands for actuators, based upon sensor feedback signals. This section also provides examples of three types of mechatronic systems and subsystems. These are standalone mechatronic systems (e.g., vehicle cruise control system, home furnace and air conditioning temperature control system, and electronic door lock system), mechatronic actuators (e.g., a DC-motor actuator, an electric variable valve timing actuation system, and an exhaust-gas-recirculation (EGR) valve), and mechatronic sensors (e.g., including a polymer flow sensor, an acceleration and rate gyro sensor, and a hall-effect position and speed sensor).

Section Keywords

Practice Problems

1. Which of the multidisciplinary mechatronic systems listed below are in engineering systems? Select the correct answer.
   a. Control and electric/electronic systems.
   b. Control and computer systems.
   c. Electric/electronic and mechanical systems.
   d. Control, computer, electric/electronic, and mechanical systems.
2. Which of the following domains make up a mechatronic system? Select the correct answer.
   a. Information domain.
   b. Physical domain.
   c. Information and physical domains.
   d. Information, interface, and physical domains
3. Which of the following mechatronic systems is not a standalone one?
   a. Electronic door locker.
   b. DC motor.
   c. Passage vehicle cruise control system.
   d. Home HVAC system.
4. Which of the following is not a mechatronic actuator?
   a. DC motor.
   b. Digital camera.
   c. EGR valve.
   d. Engine cam eVVT.
5. Which of the following is not a mechatronic sensor?
   a. MPU 6050.
   b. Polymer flow sensor.
   c. DC motor.
   d. Hall-effect sensor.
6. Which component is not part of a DC-motor control system?
   a. MPU 6050.
   b. Microcontroller.
   c. DC motor drive board.
   d. DC motor with hall-effect sensor.

This page titled 1.1: Introduction of Mechatronic Systems is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

1.2: Introduction of Mechatronic Systems

Learning Objectives

After completing this section, you will be able to:

1. Outline chapter contents of this book for Chapters 2 through 9.
2. Describe the two sample mechatronic systems introduced in Chapter 8 and 9.

This book will be used for both undergraduate and graduate level mechatronic system courses. The intension is to cover contents ranging from mechatronic system modeling and control to specific mechatronic system hardware including sensor, actuator, and microcontroller. Two mechatronic system examples will be introduced as examples for the class lab sessions; they are mini-Segway and three-wheel cart mechatronic systems. The book covers the architectures, modeling, control strategy development, and validation through simulation studies and experiments for these two systems.

For undergraduate level classes, we will use the PID (proportional-integral and derivative) control method. For graduate level classes, we will introduce and demonstrate model-based control strategies—such as pole assignment [17], linear quadratic regulation (LQR) [18], linear quadratic Gaussian (LQG) [19], and output covariance constraint (OCC) [20] control. The book has nine chapters; the rest of this section provides brief summaries.

Chapter 2 introduces methodologies for modeling mechatronic systems. Note that the mechanical part of these systems can be modeled based on Newton's law by developing the free-body diagram, then applying Newton's law to obtain the corresponding linear or nonlinear differential equations. The electric/electronic system model can be developed based on Kirchhoff's voltage and current laws [6], using time and frequency domain approaches. The mechatronic system model is often used to describe a mechatronic component that either transfers an electric/electronic signal into mechanical motion (e.g., a DC motor) or converts mechanical motion into an electric/electronic signal (e.g., an electric power generator or accelerometer sensor). Modeling the mechatronic system often utilizes these tools developed for mechanical and electric/electronic systems, resulting in linear differential equations or transfer functions.

All physical systems are nonlinear, and a linear system model approximates the physical one at its operational condition of interest. Most of the system analysis and control design techniques are developed for linear systems due to their simplicity. In order to utilize these tools, we must often linearize a nonlinear system model to obtain its linear approximation. In this case, Taylor expansion at the system operational condition of interest is often applied to nonlinear differential equations. By ignoring the high order terms in the Taylor series, or by setting these terms to zero with order higher than one, we can find a linearized model (or differential equations). For systems operated over a wide range of operational conditions, we can use a linear parameter-varying (LPV) system model [11] and design a gain-scheduling controller for the LPV system.

In addition to white-box models obtained based on physics, as discussed above, a system identification method is often deployed to obtain black- and gray-box models. The gray-box system identification method assumes a structure of system model. For instance, for a second order spring-mass-damper system, we only need to identify model coefficients such as mass, stiffness, and damping coefficient. With the model structure defined, we often use the Least-Squares (LS) method for parameter identification; machine learning methods are also popular [21]. A black-box model assumes no system model architecture, such as model order, during the system identification process with only designed system inputs and corresponding outputs. One of the black-box system identification methods is the so-called q-Markov Cover [22], [23]. It identifies a discrete linear time-invariant system (LTI) model based on the system outputs excited by either white noise, impulse, or pseudo-random binary signals (PRBS), with an analysis tool to select model order during the identification process.

Chapter 3 addresses mechatronic system behaviors, their stability and response performance indices, closed-loop control design, and tuning of a mechatronic control system, including both model-based and model-free approaches. For system performance study, we will discuss the input excitations used to study the performance of a mechatronic system, including impulse, step, and sinusoidal inputs. We will then look at system performance criteria such as step response time, settling time, system overshoot, and system steady-state error.

Because closed-loop system stability is critical for a mechatronic system, two kinds of stability criteria are addressed. They are asymptotical stability and bounded-input bounded-output (BIBO) stability. Note that for linear systems, these two criteria

are equivalent. For control of mechatronic systems, we introduce the PID (proportional-integral-derivative) controller due to its simplicity and easy gain tuning. We then discuss the Ziegler-Nichols PID gain tuning rule in detail along with the direct tuning method to optimize the mechatronic system performance. We also review model-based control methodologies, including LQR (linear quadratic regulation) [18], LQG (linear quadratic Gaussian) [19], and OCC (output covariance constraint) [20] control techniques. At the end of Chapter 3, the machine learning based scheduled PID control gain tuning methodology will be discussed in detail for graduate students.

Chapter 4 discusses the signal processing related topics in mechatronic systems. First, we present the analog signal in a continuous-time physical domain as well as its frequency domain analysis in terms of the signal magnitude and phase. Second, we illustrate the conversion process from analog physical to information domain, using analog-to-digital converters, along with the relationship between the analog and sampled digital signals. Third, we look at how digital signals inside a mechatronic system are processed by the microcontroller through filtering, signal conditioning, and generating control signals; they can be analyzed in a frequency domain.

Next, we discuss analog and digital low-pass, high-pass, band-pass, and band-kill (notch) filters. We then review the Nyquist-Shannon theorem to provide sample frequency condition to avoid sample aliasing. Continuing on, we discuss two classes of signals called pulse-width modulated and frequency modulated signals. These two signals can be directly exchanged between the physical and information domains without using A/D or D/A converters. Last, we present high-order filters, such as Butterworth [24] and Chebyshev [25] filters, to have a clean cut-off around the filter corner frequency, along with characteristics of a moving average filter.

Chapter 5 mainly addresses the actuation subsystems in the physical domain, along with their driving electronics (amplifiers). Note that there are two domains for a mechatronic system: information and physical domains. First, we review various mechatronic actuation systems—including mechanical, magnetic, hydraulic, and pneumatic systems—with examples of all mechatronic actuation systems and their typical practical applications. Then, we look at the DC-motor working principles, along with their torque coefficient (K_τ) and the back EMF (electromotive force) coefficient (K_{EMF}). We also discuss the concepts of maximum motor brake torque (stall torque) and maximum motor speed (no-load speed) as a function of input armature voltage.

Next, we introduce the DC-motor drive methodologies, including low-side drive, PWM (pulse-width-modulated) drive, and the widely used H-bridge drive, along with their implementation circuits. Last, we cover three graduate-level actuation systems. These are electromechanical valve systems [26], [27], [28] for intake and exhaust valves of internal combustion engines (linear actuator); electrohydraulic valve systems [29], [30], [31] for intake and exhaust valves of internal combustion engines (linear actuator); and the permanent magnet synchronous motor actuator [32] (brushless direct current motor—BLDC—with a permanent magnet rotor and wound stators).

Chapter 6 mainly addresses various sensing technologies critical for a closed-loop controlled mechatronic system. These technologies directly or indirectly measure the system's physical variables (e.g., speed and position). They also provide feedback signals for closed-loop controllers. First, we present speed sensors, including analog/digital rotational and linear speed sensors. Then we discuss acceleration and rate gyro sensors with their MEMS (micro-electric-mechanical system) implementation, along with a sample of MPU 6050 MEMS, an integrated three-axis accelerometer and rate gyro sensor.

We also introduce the concept of the sensor fusion to combine multiple-sensor data or data derived from disparate sources; by fusing these sources we can obtain information that is more reliable information using these sources individually. We will present two examples of fusing two sensor signals with different measurement noise levels. They will indicate that sensor fusion, in this case, significantly reduces the measurement noise and error. We will then briefly discuss the working principles of additional sensors, such as temperature, pressure, flow, current, and voltage sensing. At the end of Chapter 6, we also look briefly at another class of sensing technologies that are image-based.

Chapter 7 mainly address mechatronic systems from the control system point of view. That is, how do we use the techniques discussed in Chapters 2 and 3 to design the closed-loop controller and analyze the closed-loop mechatronic system performance? First, we review the architecture of a mechatronic system from both structural and hardware point of view, along with concepts of open- and/or closed-loop control [33], [34]. Next, we introduce the microcontroller architecture (structure) along with a sample microcontroller, Arduino MKR 1000 Wi-Fi [35], and the associated Arduino MKR motor carrier board [36].

We also present the Simulink motor control software to show how the motor drive and hall-effect sensor input blocks in the Arduino Simulink library can be used. Note that the Arduino MKR motor carrier board is open source. At last, we introduce a model-based mechatronic system development process, called V-design process. This process starts with the required mechatronic system specifications (or requirements) and ends with a mechatronic system design, resulting in a life cycle optimized product. By utilizing this iterative design process, the model-based product design process starts with a rough

mathematical model. It then develops a detailed (or complex) mathematical model (also called digital twin of the mechatronic physical system) for evaluating closed-loop mechatronic system performances with added actuator and sensor dynamics, considering the detailed physical system dynamics and nonlinearities.

Chapters 8 and 9 provide two mechatronic system examples: mini-Segway and three-wheel cart systems. Both chapters start with the model development for their respective physical systems, along with their implementation into the Matlab and Simulink environments. The Simulink mini-Segway model also includes the Coulomb friction model. With both Simulink models developed, control strategies were are developed utilizing PID controls; these are followed by model-based control approaches. Simulation validations are conducted based on the developed Simulink model and associated control strategies (PID and model-based). The goal is to find initial PID control gains and stabilize model-based control gains. Once these tuned control gains of PID and model-based control strategies are developed, we can develop real-time control software for both PID and model-based strategies and then validate them for both mini-Segway and three-wheel cart systems. Finally, experiments are conducted for the PID control strategies, followed by the model-based ones.

The book has two appendices. Appendix A introduces the Matlab and Simulink capabilities. Appendix B provides sample lab sessions for the mini-Segway.

Section Summary

This section provides a book overview; it covers contents from mechatronic system modeling and control to specific mechatronic system hardware, including sensors, actuators, and microcontrollers. In addition, the section focuses on two mechatronic system examples for the class lab sessions. These are mini-Segway and three-wheel cart mechatronic systems. Here, the associated mechatronic system architectures, modeling, control strategy development, and validation through simulation studies and experiments are introduced in detail.

Section Keywords

Practice Problems

1. What is the main principle used to model mechanical subsystems? Select the correct answer.
   a. Newton's law.
   b. Kirchhoff's law.
   c. Taylor expansion.
2. List the titles of Chapters 2 through 9.
3. Which domain do analog and digital signals belong to: information or physical?
4. Which of the following actuators are linear and which are rotational? Select the correct answer.
   a. Electromechanical valve actuator.
   b. Electrohydraulic valve actuator.
   c. Permanent magnet synchronous motor.
5. Provide rough descriptions of the two sample mechatronic systems presented in Chapters 8 and 9.

This page titled 1.2: Introduction of Mechatronic Systems is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

1.3: How to Use This Book

Learning Objectives

After completing this section, you will be able to:

1. Describe the structure of each chapter in this book.
2. Select chapter contents for undergraduate and graduate level courses, respectively.

This book was written for both undergraduate and graduate level mechatronics classes. As a result, certain contents are intended for graduate level classes. In this section, we will describe how the contents are appropriate for a) background review, b) undergraduate classes, and c) graduate courses. Note that, for the review materials, if the class prerequisite course covers them, we recommend either to let students review them as homework or to conduct a quick review in class. We also suggest assigning a few homework problems for students to refresh these concepts even if they were previously studied.

The undergraduate level contents are used mainly for undergraduate level classes, and the graduate level contents are used mainly for the graduate level classes. Many references in the book can be used for graduate level courses as term or final projects. In case the class is dual-listed for undergraduate and graduate levels, the in-class teaching should contain most of the undergraduate level contents for undergraduate students. We then suggest reviewing the graduate contents in class and then assigning additional homework for graduate students, with paper review report assignments.

Chapter 2 consists of six sections with both undergraduate and graduate level contents. The Laplace transform contents in Section 2.1 are background review materials for undergraduate level classes. If the prerequisite course covers the Laplace transform, we recommend you only review these contents with a few problems to refresh concepts on Laplace transform for this course. Section 2.2 covers modeling of mechanical systems, where the point mass-damper-spring model for linear and rotational motions is addressed for undergraduate classes. Section 2.3 then discusses modeling of electronic and electrical systems; here, the resistance, capacitance, and inductance circuit models are discussed for undergraduate level classes. Note that the finite element system modeling approach will be used for graduate level classes, and the operational amplifier circuit modeling technique will also be taught at graduate level.

Section 2.4 deals with the mechatronic system model, where the DC-motor modeling can be used for undergraduate level classes, along with the system block diagram manipulation technique. Meanwhile, the accelerometer sensor modeling can be used for graduate level classes. Section 2.5 describes system linearization techniques based on Taylor expansion, which is mainly for the undergraduate class. If additional materials are needed for the graduate level, the LPV modeling methodology discussed in reference [37] can be used for studying the linearization of nonlinear models to generate LPV models with guaranteed error bound.

The last section, Section 2.6, introduces concepts for gray- and black-box system identification techniques; they are mainly for graduate level classes. This section is important for graduate level classes because its methodologies provide a data-based modeling approach aimed to handle situations when analytic modeling approaches cannot be applied easily. Note that for the method introduced in references [22] and [23], the system identified model (black-box model) represents a linearized model if the original system is nonlinear. Therefore, certain black-box modeling techniques provide a model linearization approach if the system identification is applied to a nonlinear model in the simulation environment. Table 1.1 summarizes the selection for teaching materials for both undergraduate and graduate level courses.

Table 1.1: Chapter 2 teaching matrix for both undergraduate and graduate levels.

Section #	Undergraduate level class	Graduate level class	Comments
Section 2.1	✔		Review materials.
Section 2.2	✔	Partial	Finite element modeling approach can be included in graduate classes.

Section #	Undergraduate level class	Graduate level class	Comments
Section 2.3	✔	Partial	Operational amplifier modeling can be used for graduate classes.
Section 2.4	✔	Partial	Accelerometer model can be used for graduate classes.
Section 2.5	✔		This section is mainly for undergraduate classes.
Section 2.6		✔	Graduate level contents.

Chapter 3 consists of six sections with both undergraduate and graduate level contents. The time responses of dynamic systems in Section 3.1 provide background for evaluating closed-loop system performance for undergraduate level classes. If the prerequisite course includes control systems, we recommend only reviewing these contents with an assignment of enough problems to make sure students have the required background. Section 3.2 addresses the closed-loop system performance indices of the step-input. This includes response time, setting time, and percentage overshoot for undergraduate classes. Before reviewing the performance indices in this section, we review the closed-loop system stability criteria (bounded-input and bounded-output stability and asymptotical stability) with a discussion on Lyapunov stability for graduate level courses.

Section 3.3 addresses the performance indices mainly for the first- and second-order closed-loop systems with analytic solutions for settling time, percentage overshoot, etc. It also covers the concept of time constant for the first-order systems. It then notes that there is no overshoot for the first-order systems. Furthermore, the overshoot of the second-order systems is only a function for system damping ratio. The contents of this section are mainly for undergraduate courses, for either teaching or reviewing. Section 3.4 discusses the PID (proportional-integral-derivative) controller widely used in industry due to its simplicity and easy gain tuning. We address the Ziegler-Nichols PID gain tuning rule [6] in detail to be used as initial control gains and the starting point for control gain tuning. The section also presents an ad hoc (direct) PID control gain tuning method. In industrial applications, PID gains are often scheduled as a function of response error to improve closed-loop system performance; this material is graduate level. Examples demonstrate how the gain scheduled PID control can improve system performance.

Section 3.5 introduces model-based control techniques for mechatronic systems such as the LQR (linear quadratic regulation) and LQG (linear quadratic Gaussian) optimal control techniques [17], [18], [19] for linear mechatronic systems. The main challenges of utilizing the LQR or LQG control for mechatronic systems are how to tune the model-based controller. The section also discusses the OCC (output covariance constraint) control technique [20] to address the issue of model-based control gain tuning since a family of OCC controllers can be generated with gains from low to high. To deal with nonlinear mechatronic systems, the LPV (linear parameter-varying) control [11], [37] can be used to design gain-scheduling controllers for handling the mechatronic system nonlinearity. Then, Section 3.6 discusses how to use machine learning to tune gain-scheduled PID controllers [7]. Both Sections 3.5 and 3.6 are for graduate level classes.Table 1.2 summarizes the selection for teaching materials at both undergraduate and graduate levels.

Table 1.2: Chapter 3 teaching matrix for both undergraduate and graduate levels.

Section #	Undergraduate level class	Graduate level class	Comments
Section 3.1	✔		Review materials.
Section 3.2	✔	Partial	Lyapunov stability can be used for graduate classes using reference [38].
Section 3.3	✔		This section is mainly for undergraduate classes.
Section 3.4	✔	Partial	PID gain tuning method should be reviewed for graduate classes.
Section 3.5		✔	This section is mainly for graduate classes.

Section #	Undergraduate level class	Graduate level class	Comments
Section 3.6		✔	This section is mainly for graduate classes.

Chapter 4 consists of seven sections with both undergraduate and graduate level contents. In Section 4.1, the analog signal in continuous-time physical domain and its frequency domain analysis provide methods for analyzing continuous-time signal frequency characteristics. If the prerequisite course covers control systems and Bode plot, we recommend only reviewing these contents with an assignment of enough problems to make sure students have the required background. Section 4.2 addresses the signal conversion between the physical and information domains using A/D and D/A converters. The A/D and D/A conversion circuit contents can be used for graduate level classes. Section 4.3 addresses the digital signal inside a mechatronic system that is processed by the microcontroller through filtering, signal conditioning, and generating control signals. Note that digital signal processing is quite different from analog signal processing. The frequency domain analysis is often conducted through the discrete Fourier transform that maps a discrete-time function into a *z*-transform transfer function, where a stable *z*-transform transfer function has all its poles (roots of the denominiant polynomial) inside of the unit circle of the complex "*z*" plan. To accelerate the discrete Fourier transform, we also discuss the fast Fourier transform used to accelerate the discrete Fourier transform; this material is used for graduate classes.

Section 4.4 discussed analog and digital low-pass, high-pass, band-pass, and band-kill (notch) filters, along with their realization analog circuits and generation of associated discrete-time filter algorithms. Note that both band-pass and band-kill (notch) filter contents can be used for graduate classes. Section 4.5 reviews the Nyquist-Shannon sample theorem, providing required sample frequency condition to avoid sample aliasing. The section also points out that an analog anti-aliasing filter has its cutoff frequency at least two times below the sample frequency. The sample aliasing content can be part of graduate classes.

Section 4.6 addresses two kinds of signals: pulse-width modulated and frequency modulated signals. Note that both signals can be exchanged between the physical and information domains without using A/D or D/A converters. We recommend that the content for frequency modulated signals shall be included in graduate classes. Finally, Section 4.7 discusses high-order filters—such as Butterworth [24] and Chebyshev [25] filters—that have a clean cutoff around the filter corner frequency. We also analyze characteristics of the moving average filter in frequency domains. Note that contents of Section 4.7 are for graduate classes. Table 1.3 summarizes the selection of teaching materials for both undergraduate and graduate levels.

Table 1.3: Chapter 4 teaching matrix for both undergraduate and graduate levels.

Section #	Undergraduate level class	Graduate level class	Comments
Section 4.1	✔		This section is mainly for undergraduate classes.
Section 4.2	✔	Partial	Part of A/D and D/A conversion realization circuits can be used for graduate classes.
Section 4.3	✔	Partial	Fast Fourier transform content can be used for graduate classes.
Section 4.4	✔	Partial	Band-pass and band kill (notch) filter design can be used for graduate classes.
Section 4.5	✔	Partial	Sample aliasing content is for graduate classes.
Section 4.6	✔	Partial	Frequency modulated signal is for graduate classes.
Section 4.7		✔	This section is mainly for graduate classes.

Chapter 5 consists of six sections with both undergraduate and graduate level contents. Section 5.1 reviews various mechatronic actuation systems including mechanical, magnetic, hydraulic, and pneumatic systems. For graduate level classes, it discusses electromagnetic, electrohydraulic, and electro-pneumatic actuations in detail. Section 5.2 introduces working principles of DC motors and addresses how to calculate both the DC-motor torque coefficient (K_τ, a proportional gain of motor armature current)

and the back EMF (electromotive force) coefficient (K_{EMF}, a proportional gain of motor speed).

Section 5.3 addresses the methodologies of driving the DC motor efficiently. It includes low-side drive, PWM (pulse-width-modulated) drive, and the widely used H-bridge drive based on the PWM signal. Note that the H-bridge drive enables you to generate the motor torque in two directions: positive and negative. This is very important for closed-loop control displacement or velocity with the DC motor as the actuator. Both Sections 5.2 and 5.3 are mainly for undergraduate classes.

The remaining three sections have graduate level contents. Sections 5.4 and 5.5 respectively review electromagnetic and electrohydraulic mechanical valve systems used in internal combustion engines, with detailed model and simulation results. Note that both electromagnetic and electrohydraulic mechanical valves are linear actuation.

Meanwhile, Section 5.6 introduces the permanent magnet synchronous motor actuator [32]. This type of motor is widely used in the powertrain of electric vehicles; it has attracted intensive research and development in the automotive industry. An example of a permanent magnet synchronous electric motor is an AC electric motor with its steady-state shaft rotation synchronized with the control current frequency. It can be considered similar to a brushless direct current (BLDC) motor with a permanent magnet rotor and wound stators. Section 5.6 also addresses the vector control (VC) using PWM signals as inputs to the inverter circuit; see references [39], [40], and [41]. Table 4.1 summarizes the selection of teaching materials for both undergraduate and graduate levels.

Table 1.4: Chapter 5 teaching matrix for both undergraduate and graduate levels.

Section #	Undergraduate level class	Graduate level class	Comments
Section 5.1		✔	This section is mainly for graduate classes.
Section 5.2	✔		The section is mainly for undergraduate classes.
Section 5.3	✔	Partial	The H-bridge drive circuit content can be used for graduate classes.
Section 5.4		✔	This section is mainly for graduate classes.
Section 5.5		✔	The section is mainly for graduate classes.
Section 5.6		✔	This section is mainly for graduate classes.

Chapter 6 consists of eight sections. It mainly addresses various sensor technologies critical for a closed-loop controlled mechatronic system. These sensors are used for measuring the system physical variables (e.g., speed and position) directly or indirectly. They also provide feedback signals for the closed-loop controller. Section 6.1 discusses mechatronic sensors, indicating that sensors can be divided into analog sensors with continuous (analog) output signals and digital sensors with discrete (logic high and low) outputs, respectively. This section is for undergraduate classes.

Section 6.2 presents rotational speed sensing, including analog and digital rotational and linear speed sensors. The rotational analog speed sensor can be easily realized by using a DC motor (see Chapter 2 and reference [6]). When the DC motor spins, its armature voltage is proportional to the rotation speed. A digital speed sensor can be realized by an encoder [42] with the help of a so-called hall-effect sensor. Sections 6.3 and 6.4 discuss acceleration and rate-gyro sensors with their MEMS (micro-electric-mechanical system) implementation. Sections 6.2, 6.3, and 6.4 are mainly for undergraduate classes.

Section 6.5 introduces for undergraduate classes the concept of the sensor fusion [43]. It combines multiple-sensor data or data derived from disparate sources to obtain more reliable information, rather than using these sources individually. We present two examples of fusing two sensor signals with different measurement noise levels. These examples indicates that sensor fusion can significantly reduce the measurement noise. Section 6.6 then introduces for undergraduate classes the MPU 6050 MEMS integrated three-axis accelerometer and rate gyro sensor. Section 6.7 provides working principles of additional sensors, such as temperature, pressure, flow, current, and voltage sensing. In turn, Section 6.8 briefly discusses another class of sensing technologies, image-based sensing. Both Sections 6.7 and 6.8 are for graduate classes. Table 5.1 summarizes the selection of teaching materials for both undergraduate and graduate levels.

Table 1.5: Chapter 6 teaching matrix for both undergraduate and graduate levels.

Section #	Undergraduate level class	Graduate level class	Comments
Section 6.1	✔	Partial	The sensing based on image processing part is for graduate classes.
Section 6.2	✔	Partial	This section is mainly for undergraduate classes except for the Doppler effect.
Section 6.3	✔		This section is mainly for undergraduate classes.
Section 6.4	✔	Partial	This section is mainly for undergraduate classes except for vibration based sensing.
Section 6.5	✔	Partial	Only the rotational angular position based on sensor-fusion of accelerometer and rate gyro signals is for undergraduate classes.
Section 6.6	✔		This section is mainly for undergraduate classes.
Section 6.7		✔	This section is mainly for graduate classes.
Section 6.8		✔	This section is mainly for graduate classes.

Chapter 7 utilizes the techniques addressed in Chapters 2 through 6 for implementing the designed closed-loop controllers to the target microcontroller. Section 7.1 reviews the architecture of a mechatronic system from both structural and hardware points of view. We discuss the open- and closed-loop control system architecture and its associated hardware, with a few examples. Section 7.2 introduces the microcontroller architecture (structure) [44] with the difference between a microcontroller and microprocessor. Note that a microcontroller unit (MCU) is essentially a small computer on a single chip. It is designed to manage specific tasks within an embedded system without requiring a complex operating system. By contrast, while a microprocessor [45] is a computer processor with the data processing logic and control included on a single integrated circuit (IC) or a small number of ICs; the microprocessor contains the arithmetic, logic, and control circuitry required to perform the functions of a central processing unit (CPU).

Section 7.3 provides detailed information of a sample microcontroller, the Arduino MKR 1000 Wi-Fi [35]. First, we discuss the Arduino family of microcontrollers, along with its IDE (integrated development environment) software [46] and program environment under MATLAB and Simulink [47]. Section 7.4 addresses the control system architecture for mechatronic systems. We provide an example of DC-motor speed control, noting that the overall mechatronic system architecture of both hardware and software were introduced in Section 7.1.

Section 7.5 introduces a model-based iterative mechatronic system development process, called the V-design process. This process starts with the required mechatronic system specifications (or requirements) and ends with a mechatronic system design, resulting in a life cycle optimized product. By utilizing the iterative design process, the model-based product design process starts with a rough mathematical model. It follows the rough model by developing a detailed mathematical model (also called a digital twin of the mechatronic physical system). This model evaluates closed-loop mechatronic system performances by adding actuator and sensor dynamics and then considering the detailed physical system dynamics and nonlinearities. This chapter is mainly for graduate classes with partial contents for review in undergraduate classes. Table 1.6 summarizes the selection of teaching materials for both undergraduate and graduate levels.

Table 1.6: Chapter 7 teaching matrix for both undergraduate and graduate levels.

Section #	Undergraduate level class	Graduate level class	Comments
Section 7.1	Partial	✔	Most of the contents are for graduate classes.
Section 7.2	Partial	✔	This section is mainly for graduate classes except for the MKR 1000 discussion.

Section #	Undergraduate level class	Graduate level class	Comments
Section 7.3	Partial	✔	This section is mainly for graduate classes except for the MKR motor carrier discussion.
Section 7.4	Partial	✔	This section is mainly for graduate classes except for the speed control discussion.
Section 7.5		✔	This section is for graduate classes.

Chapters 2 through 7 introduced technologies used in mechatronic system design, control, and optimization, including mechatronic system modeling and control, signal processing in both continuous- and discrete-time domains, mechatronic system actuators and sensors, and the control system architecture from both hardware and software points of view. Chapter 8 presents the first example of mechatronic systems, a mini-Segway system, from its modelling to control design to simulation and experimental studies.

Section 8.1 models a mini-Segway, where the modeling techniques addressed in Chapter 2 are used. Note that, mechanically, the mini-Segway can be considered as an inverted pendulum on a cart. Section 8.2 proposes a control architecture of using two independent PID (proportional-integral-derivative) controllers to stabilize the mini-Segway pendulum (maintaining it at the uppright position) and control the mini-Segway forward and backward motion, respectively. Note that the coupling effect of pendulum angle displacement, forward/backward motion, and turning motions are ignored. Sections 8.1 and 8.2 are for undergraduate classes.

Section 8.3 addresses the model-based design process (introduced in Section 3.5) for the mini-Segway, utilizing the LQR (linear quadratic regulation) [20] and LQTI (linear quadratic tracking with integral) [33]. Section 8.4 introduces control design for discrete-time controllers with sample and hold, different from conventional zero-order ones [48], [49]. By utilizing the special sample and hold, a non-minimal phase continuous-time plan can be transferred into a minimal phase (MP) discrete-time plant. This section demonstrates this phenomenon, using the mini-Segway model developed earlier and then proceed to design a stabilizing controller. Sections 8.3 and 8.4 are mainly for graduate classes.

Table 1.7: Chapter 8 teaching matrix for both undergraduate and graduate levels.

Section #	Undergraduate level class	Graduate level class	Comments
Section 8.1	✔	Partial	This section is mainly for undergraduate classes, except for the Coulomb friction model.
Section 8.2	✔	Partial	This section is mainly for undergraduate classes.
Section 8.3		✔	This section is mainly for graduate classes.
Section 8.4		✔	This section is mainly for graduate classes.
Section 8.5	✔		This section is mainly for undergraduate classes.
Section 8.6		✔	This section is mainly for graduate classes.
Section 8.7	✔	Partial	This section is for both undergraduate and graduate classes.
Section 8.8		✔	This section is mainly for graduate classes.

The rest of Chapter 8 validates these controller strategies through simulation and experimental studies. Section 8.5 describes how the dual-PID control strategy is implemented in Simulink and validated, based on the linearized mini-Segway full-state model with detailed PID gain tuning to achieve the desired performance (stabilization and driving). This section is for undergraduate

classes. Section 8.6 then shows how the model-based simulation investigation is conducted utilizing the model-based controllers (LQR and LQTI) designed in Section 8.4; this section is for graduate classs.

In Section 8.7, the dual-PID mini-Segway control strategy is validated through experiments with detailed implementation of dual-PID control strategy into a real-time Simulink model running within the host computer. The Arduino MKR 1000 microcontroller collects the sensor signals and executes the motor drive commands, where communication between the host computer and the MKR 1000 microcontroller is through the Wi-Fi. This section is at the undergraduate level. The last section, Section 8.8, addresses the experimental studies of a model-based control strategy for regulating the mini-Segway pendulum position only.Both standard zero-order sample and hold input (ZOSHI) and the square-pulse sample and hold input (SPSHI) [48], [49] are used. With the dual-loop control scheme, the average cart displacement magnitude is reduced by more than 65% using the SPSHI for the outer-loop controller. This section is at the graduate level. Table 1.7 summarizes the selection of teaching materials for both undergraduate and graduate levels.

Chapter 9 introduces the second example of mechatronic systems. It is a three-wheel cart system with the same electronics as these for the mini-Segway, two driving wheels in the front, and one swivel caster wheel in the back. Section 9.1 starts modeling the three-wheel cart system, considering the cart wheel side-slip on an unlevelled road surface. Note that the three-wheel cart moves aside even when the two driving wheels have the same speed. In addition, the developed cart model is implemented in MATLAB and Simulink.

With the three-wheel cart model developed (considering the tire side slip due to lateral force), Section 9.2 describes both dual- and triple-PID control strategies with their simulation-validation provided in Section 9.4. The dual-PID control architecture uses two PID controllers to regulate the two front driven wheels speeds of the three-wheel cart. These controllers track the wheels' references so that the cart's forward driving and turning can be controlled by adjusting the reference speeds for the two driven wheels. The triple-PID strategy adds another PID controller to track the cart turning speed reference with rate-gyro feedback. Section 9.3 proposes a model-based control strategy, called LQTI (linear quadratic tracking with integral) control; it is based upon the state-space model obtained by linearizing the three-wheel cart model developed earlier. The LQTI controller is designed to track both forward/backward and turning speeds of the three-wheel cart. Simulation studies in Section 9.4 validate this design.

Table 1.8: Chapter 9 teaching matrix for both undergraduate and graduate levels.

Section #	Undergraduate level class	Graduate level class	Comments
Section 9.1		✔	This section is only for graduate classes.
Section 9.2		✔	This section is only for graduate classes.
Section 9.3		✔	This section is only for graduate classes.
Section 9.4		✔	This section is only for graduate level class.
Section 9.5		✔	This section is only for the graduate classes.
Section 9.6		✔	This section is only for graduate classes.

In Section 9.5, the real-time Simulink control software of the dual- and triple-PID control strategies is developed and implemented into a Surface laptop. After debugging the software, both dual- and triple-PID control gains are tuned to achieve the best performance. Then, both dual- and trip-PID control strategies are tested in straight and turning driving operations, where the triple-PID control strategy is capable of compensating uneven motor drive systems of the three-wheel cart. Next, in Section 9.6, we develop the real-time Simulink control software of the model-based control strategy (LQTI). After the software is implemented into the Surface laptop, it is debugged, followed by the experimental study for model-based LQTI control of the three-wheel cart. The test data shows the advantage of the LQTI strategy over the triple-PID strategy. All contents in this chapter are at the graduate level; see Table 1.8.

Appendix A provides a brief introduction of the MATLAB and Simulink software. In addition, Appendix A lists MATLAB toolboxes that can be used for mechatronic system research and development. Both MATLAB's real-time workshop and state flow toolbox ae discussed with respect to their application of developing real-time Simulink software for the Arduino MKR family microprocessors.

Appendix B discusses installing MATLAB and Simulink for machines that use Windows. It then provides nine mini-Segway lab sessions that instructors can use to create lab sessions; the first eight are for undergraduate classes and the last is for graduate classes. These include 1) simulation of 1st and 2nd order systems in MATLAB and Simulink, 2) fast Fourier transformation (FFT) and filter design, 3) filter design and introduction to Arduino MKR 1000 Wi-Fi, 4) DC-motor control utilizing Arduino MKR hardware, 5) Simulink GUI (graphic user interface) design and programming, 6) sensor fusion vs. individual sensors, 7) PID gain

tuning for the mini-Segway Simulink model, 8) PID gain tuning for the mini-Segway hardware system, and 9) a mini-Segway stabilization and drive project.

Section Summary

This section provides recommendations regarding how to use this book, especially since it is written for both undergraduate and graduate level mechatronics classes. Certain contents are for graduate level classes. For each chapter, recommendations are provided to identify which contents are appropriate for a) background review, b) undergraduate classes, and c) graduate classes. Some review materials are considered prerequisite to the class. You may want to review these, whether as homework or in class.

Section Keywords

Practice Problems

1. List one PID controller gain tuning method discussed in Chapter 3.
2. List two advanced filter design methods in Chapter 5.
3. What is MPU 6050? Select the correct answer.
   a. Three-axis accelerometer.
   b. Three-axis rate gyro.
   c. Three-axis accelerometer and three-axis rate-gyro.
4. Is the Arduino MKR 1000 capable of Wi-Fi communication?
5. Does the developed mini-Segway model include the Coulomb friction model?
6. For the three-wheel cart control, between PID and model-based LQTI strategies, which one provides better performance?

This page titled 1.3: How to Use This Book is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

1.4: Chapter References

References

1. A. G. Ulsoy, H. Peng, and M. Çakmakci, *Automotive Control Systems*, Cambridge, U.K.: Cambridge Univ. Press., 2012, pp. 213–224, ISBN-13: 978-1-1070-1011-6.
2. J. V. Candy, "Accelerometer modeling in the state-space," *LLNL*, Accessed: Feb. 10, 2021. [Online] Available: https://www.osti.gov/servlets/purl/1777338, doi:10.2172/1777338.
3. F. J. Riley, "Constant speed regulator," U.S. Patent 2,714,880, Aug. 9, 1955.
4. Almadani, N. Alshammari, and A. Al-Roubaiey, "Adaptive cruise control based on real-time DDS middleware," *IEEE Access* [Online], vol. 11, pp. 75407–75423, 2023.
5. M.A.S. Kamal, K. Hashikura, T. Hayakawa, K. Yamada, and J.-i., Imura, "Adaptive cruise control with look-ahead anticipation for driving on freeways," *MPDI Appl. Sci.,* vol. 12, p. 929, 2022.
6. R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 12th ed., Englewood Cliffs, NJ, USA: Prentice Hall, 2011, ISBN-10: 0-13-602458-0.
7. S. Qu, T. He, and G. Zhu, "Gain-scheduled PID control optimized by iterative genetic algorithm of NSGA-II with application to a nonlinear valve system," *MPDI J. Appl. Sci. (Spec. Issue Res. Appl. Intel. Control Algor.)*, vol. 13, no. 5, p. 6444, May 2023, doi: 10.3390/app13116444.
8. Y. Du, H. Zandi, O. Kotevska, K. Kurte, J. Munk, K. Amasyali, E. Mckee, and F. Li, Yan, "Intelligent multi-zone residential HVAC control strategy based on deep reinforcement learning," *Appl. Energy*, vol. 281, p.116117, 2021.
9. Z. Ren and G. Zhu, "Modeling and control of an electrical variable valve timing actuator," *ASME J. Dyn. Syst., Meas., Control*, vol. 136, no. 2, pp. 021015, 2014, doi: 10.1115/1.4025914.
10. K. Al-Jiboory, A. White, S. Zhang, G. Zhu, and J. Choi, "LMI solutions to the input covariance constraint control: application to electronic throttle control," *ASME J. Dyn. Syst., Meas., and Control*, vol. 137, no. 9, 091010-1, Sep. 2015, doi: 10.1115/1.4030525.
11. S. Qu, T. He, and G. Zhu, "Switched LPV modeling and control of an engine EGR valve with nonlinear dry friction," *IEEE/ASME Trans. Mechatron.*, vol. 25, no. 3, pp. 1668–1678, Jun. 2020, doi: 10.1109/TMECH.2020.2982315.
12. X. Chen, G. Zhu, X. Yang, D. Hung, and X, Tan, "Model-based estimation of flow characteristics using an ionic polymer-metal composite beam," *IEEE/ASME Trans. Mechatron.*, vol. 18, no. 3, pp. 932–943, June 2013, doi: 10.1109/TMECH.2012.2194300.
13. Z. Chen, S. Shatara, and X. Tan, "Modeling of biomimetic robotic fish propelled by an ionic polymer-metal composite caudal fin," *IEEE/ASME Trans. Mechatron.*, vol. 15, no. 3, pp. 448–459, 2010.
14. MPU 6050 data sheet, Accessed: Apr. 18, 2024. [Online] Available: https://invensense.tdk.com/wp-conten...Datasheet1.pdf.
15. E. Ramsden, *Hall Effect Sensors: Theory and Applications*, 2nd ed., Amsterdam, The Netherlands; New York, NY, USA: Elsevier (Science Publishing Co.), 2006, ISBN-13: 978-0-7506-7934-3.
16. P. Regtien and E. Dertien, *Sensors for Mechatronics*, 2nd ed., Amsterdam, The Netherlands; New York, NY, USA: Elsevier (Science Publishing Co.), 2018, IBSN-13: 978-0-12-813810-6.
17. J. P. Hespanha, *Linear Systems Theory*, Prineton, NJ, USA: Princeton Univ. Press, 2009, ISBN-13: 978-0-691-14021-6.
18. M. Athens and P. L. Flab, *Optimal Control: An Introduction to the Theory and Its Applications*, New York, NY, USA: Dover (Publications, Inc.), 2007, ISBN-10: 0-48-645328-6.
19. D. S. Naidu, *Optimal Control Systems*, Boca Raton, FL, USA: CRC Press (Inc.), 2003, ISBN-13: 978-1-3152-1442-9.
20. G. Zhu, M. A. Rote, and R. E. Skelton, "A convergent algorithm for the output covariance constraint control problem," *SIAM J. Control Optim.,* vol. 35, no. 1, pp. 341–361, Jan. 1997.
21. L. Hua, J. Tang, H. Dourra, and G. Zhu, "A gray-box surrogate vehicle energy consumption model capable of real-time updating," *Fourth Edition of Focused Section on TMECH/AIM Emerging Topics, IEEE/ASME Trans. Mechatron.*, vol. 24, no. 4, pp. 2092–2100, Aug. 2023, doi:10.1109/TMECH.2023.3276743.
22. G. Zhu, R. E. Skelton, and P. Li, "Q-Markov cover identification using pseudo-random binary signals," *Int. J. Control,* vol. 62, no. 6, pp. 1273–1290, 1995.
23. G. Zhu, "Weighted multirate *q*-Markov cover identification using PRBS — an application to engine systems," *Math. Prob. Eng.*, vol. 6, pp. 201–224, 2000.
24. S. Butterworth, "On the Theory of Filter Amplifiers," *Exp. Wireless Eng.*, vol. 7, pp. 536–541, 1930.
25. J. G. Proakis and Dimitris G. Manolakis, *Introduction to Digital Signal Processing*, New York, NY, USA: Macmillan (Inc.), 1988, ISBN-10: 0-02-396810-9.
26. W. S. Chang, T. A. Parlikar, M. D. Seeman, D. J. Perreault, J. G. Kassakian, and T. A.Keim, "A new electromagnetic valve actuator," *Power Electron. Transp.*, Auburn Hills, MI, USA, 2022, pp. 109–118, doi: 10.1109/PET.2002.1185558.
27. K. S. Peterson, A. G. Stefanopoulou, and J. Freudenberg, "Current versus flux in the control of electromechanical valve

actuators," In *Proc. 2005 Amer. Control Conf.*, Portland, OR, USA, June 8–10, 2005, pp. 5021–5026.
28. I. Haskara, V. V. Kokotovic, and L. A. Mianzo, "Control of an electro-mechanical valve actuator for a camless engine," *Int. J. Robust Nonlinear Control*, vol. 14, no. 6, pp. 561–579, Feb. 2004.
29. D. Lou and G. Zhu, "Review of advancement in variable valve actuation of internal combustion engines," *Appl. Sci., Mech. Eng.*, vol. 10, p. 1216, Feb. 2020, doi: 10.3390/app10041216.
30. H. Li, Y. Huang, G. Zhu, and Z. Lou, "Adaptive LQT valve timing control for an electro-hydraulic variable valve actuator," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 5, pp. 2182–2194, Sep. 2019, doi: 10.1109/TCST.2018.2861865.
31. Z. Lou, Q. Deng, S. Wen, Y. Zhang, M. Yu, M. Sun, and G. Zhu, "Progress in camless variable valve actuation with two-spring pendulum and electro-hydraulic latching," *SAE Int. J. Engines, SAE 2013-01-0590*, vol. 6, no. 1, 2013 doi:10.4271/2013-01-0590.
32. K. Suman, K, Suneeta, and M. Sasikala, "Direct torque-controlled induction motor drive with space vector modulation fed with three-level inverter," In *Proc. 2012 IEEE Int. Conf. Power Electron., Drives, and Energy Syst. (PEDES),* Sep. 2020, pp. 1–6.
33. C. Gamache, M. Van Nieuwstadt, J. Martz, and G. Zhu, "LQTI boost pressure and EGR rate control of a diesel air charge system with eBoost assistance," *Int. J. of Engine Res.*, vol. 25 no. 9, pp. 1659–1670, Sep. 2024, doi: 10.1177/14680874241241364.
34. J. Chotai and K. Narwekar, "Modelling and position control of brushed DC motor," In *Proc. 2017 Int. Conf. Adv. Comp., Comm., Control (ICAC3)*, Mumbai, India, 2017, pp. 1–5, doi: 10.1109/ICAC3.2017.8318792.
35. Arduino MKR 1000 Wi-Fi microcontroller, Accessed Jan. 17, 2025. [Online] Available: https://store-usa.arduino.cc/product...3ifuAoM7XW9Tsy
36. Arduino MKR Motor Shield (Carrier) Board, Accessed Jan. 22, 2025. [Online] Available: https://store-usa.arduino.cc/product...f91tUDBsREKBez.
37. A. K. Al-Jiboory, G. Zhu, S. SM Swei, W. Su, and N. T. Nguyen, "LPV modeling of a flexible wing aircraft using adaptive model gridding and alignment methods," *J. Aerosp. Sci. Technol.*, vol. 66, pp. 92–104, Jul. 2017, DOI: 10.1016/j.ast.2017.03.009.
38. H. Khalili, *Nonlinear Systems*, 3rd ed., Upper Saddle River, NJ, USA: Pearson, 2002, ISBN-13: 978-0-1306-7389-3.
39. M. A. Perales, M. M. Prats, R. Portillo, J. L. Mora, J. I. León, and L.G. Franquelo, "Three-dimensional space vector modulation in ABC coordinates for four-leg voltage source converters," *IEEE Power Electron. Lett.*, vol. 1, no. 4, pp. 104–109, Dec. 2003.
40. Ó. Lopez, J. Alvarez, J. Doval-Gandoy, and F. D. Freijedo, "Multilevel multiphase space vector PWM algorithm," *IEEE Trans. on Ind. Electron.,* vol. 55, no. 5, pp. 1933–1942, May 2008.
41. V. R. Reddy and V. C. V. Reddy, "Mathematical analysis of SCPWM for inverter fed DTC of induction motor drive," *Int. J. Appl. Eng. Res.*, vol. 13, no. 1, pp. 47–52, 2018.
42. C. Farrell Winder, "Shaft angle encoders afford high accuracy," *Electron. Industries*, vol. 18, no. 10, Philadelphia, PA, USA, Chilton (Co.), pp. 76–80, Oct. 1959.
43. W. Elmenreich, "Sensor fusion in time-triggered systems," Ph.D. thesis, *Vienna Univ. of Tech.,* Vienna, Austria, 2002, pp. 173.
44. R. Oshana, Ed., *DSP for Embedded and Real-Time Systems*, Oxford, U.K.: Newnes, 2012, ch. 3, pp. 29–61, doi: 10.1016/B978-0-12-386535-9.02001-1, ISBN-13: 978-0-1238-6535-9.
45. S. Augarten, *The Most Widely Used Computer on a Chip: the TMS 1000*, New Haven, CT, USA: Ticknor & Fields, 1983, ISBN-13: 970-0-89919-195-9.
46. Arduino IDE software, Accessed Aug. 20, 2025. [Online] Available: https://www.arduino.cc/en/software/.
47. Simulink supporting package for Arduino hardware (including Arduino MKR 1000 Wi-Fi), Accessed Aug. 20, 2025. [Online] Available: https://www.mathworks.com/matlabcent...duino-hardware.
48. Y. Wang, G. G. Zhu, R. Mukherjee, "Experimental study of NMP sample and hold input using an inverted pendulum," In *Proc. 2018 ASME Dyn. Syst. Control Conf.*, Atlanta, GA, USA, Sep.30–Oct. 3, 2018, p. V001T01A004.
49. Y. Wang and G. Zhu, "Performance improvement of an NMP mini Segway using sample and hold," *MPDI J. Appl. Sci.* (*Spec. Issue Recent Adv. Mech. Vib. Control*), vol. 13, no. 2, pp. 1070, Feb. 2023, doi: 10.3390/app13021070.

Solutions to Practice Problems

1. The answer is d).
2. The answer is c).
3. The answer is b).
4. The answer is b).
5. The answer is c).
6. The answer is a).

1. The answer is a).
2. Titles of Chapters 2 through 9 are: 2) Mechatronic System Modeling; 3) System Responses and Controls; 4) Signal Processing; 5) Mechatronic System Component: Actuators; 6) Mechatronic System Component: Sensors; 7) Mechatronic Control Systems, 8) Mini-Segway System: Mechatronic System Example 1; and 9) Three-Wheel Cart: Mechatronic System

Example 2.
3. The analog signal belongs to the physical domain, and the digital signal to the information domain.
4. Actuators a) and b) are linear. Actuator c) is rotational.

1. The mini-Segway mechatronic system consists of two driving wheels. They are powered in front by two DC-motor wheel assemblies with associated hall-effect speed sensors, and in back by one caster wheel with the following electric/electronic components: a high-quality rechargeable sealed lead-acid battery (12V 12Ah), Arduino MKR 1000 Wi-Fi board, MKR motor carrier board, MPU 6050 three-axis acceleration and rate gyro sensor board, and the MSU interface board.

The three-wheel cart mechatronic system consists of two driving wheels.They are powered in front by two DC-motor wheel assemblies with associated hall-effect speed sensors, and in back by one caster wheel with the following electric/electronic components: a high-quality rechargeable sealed lead-acid battery (12V 12Ah), Arduino MKR 1000 Wi-Fi board, MKR motor carrier board, MPU 6050 three-axis acceleration and rate gyro sensor board, and the MSU interface board.

1. Ziegler-Nichols PID control gain-tuning method.
2. They are Butterworth and Chebyshev.
3. The answer is c).
4. The answer is: yes, it is capable
5. The answer is: yes, the developed mini-Segway model includes the Coulomb friction model.
6. The performance of model-based LQTI strategy is better.

This page titled 1.4: Chapter References is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

2: MECHATRONIC SYSTEM MODELING

Learning Objectives

1. Conduct continuous-time Laplace transforms for basic time domain functions such as impulse, step, ramp, and trigonometric functions, as well as associated inverse Laplace transforms using fractional expansion approach.
2. Model basic mechanical system elements of spring, damper, and mass for both linear and rotational motions. Become familiar with the process of generating mechanical system transfer function from the derived ordinary differential equation. At the graduate level, be able to model the continuous mechanical system using the finite element analysis method.
3. Model basic elements of resistors, capacitors, and inductors for electric/electronic system. Become familiar with the process of generating electric/electronic system transfer functions based upon the derived ordinary differential equations as well as using impedance approach in frequency domains. At the graduate level, be able to model the operational amplifier circuits.
4. Model basic mechatronic subsystems such as the DC motor, along with its interaction between the mechanical and electric/electronic components in a mechatronic subsystem.
5. Conduct linearization, using the Taylor expansion, for nonlinear system models described by nonlinear differential equations.
6. Define gray- and black-box modeling techniques at the graduate level, where the system models are developed based upon the system input and output data.

This chapter reviews the modeling techniques for mechanical, electric/electronic, and mechatronic subsystems. It also provides a methodology for linearizing the nonlinear system model at the operational condition of interest so that a linear transfer function model can be obtained.

2.0: Prelude

2.1: Laplace Transforms

2.2: Mechanical System Modeling

2.3: Electrical and Electronic System Modeling

2.4: Mechatronic System Modeling

2.5: System Linearization

2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning

2.7: Chapter References

This page titled 2: Mechatronic System Modeling is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

2.0: Prelude

As discussed in Chapter 1, a mechatronic system consists of mechanical, electric/electronic, and mechatronic subsystems. The mechatronic subsystem often converts an electric/electronic signal to mechanical motion (e.g., a DC motor) or mechanical motion to an electronic signal (e.g., an acceleration sensor called an accelerometer). Because of its complexity, studying characteristics of a mechatronic system often requires a mathematical model. This model can be used for system analysis through simulation studies, model-based control design, and closed-loop system performance evaluation and validation.

For a given input to the physical system shown in Figure 2.1, Output 1 is generated by the physical system. For the same input, the simulated system response is Output 2. Because system modeling errors always exist, the system responses Output 1 and Output 2 are not the same. The objective of the system modeling process is to keep the modeling error as small as possible. However, there exists no exact model; a good model is simple yet captures the essential system dynamics. Sometimes, the model response may not match well with the physical system response, but it could be a good control design model, leading to a high-performance model-based controller. That is, we are not looking for a completely accurate model. Instead, we are trying our best to find a model good for control analysis and synthesis for mechatronic systems, which requires accumulation of modeling and control design experience to get there.

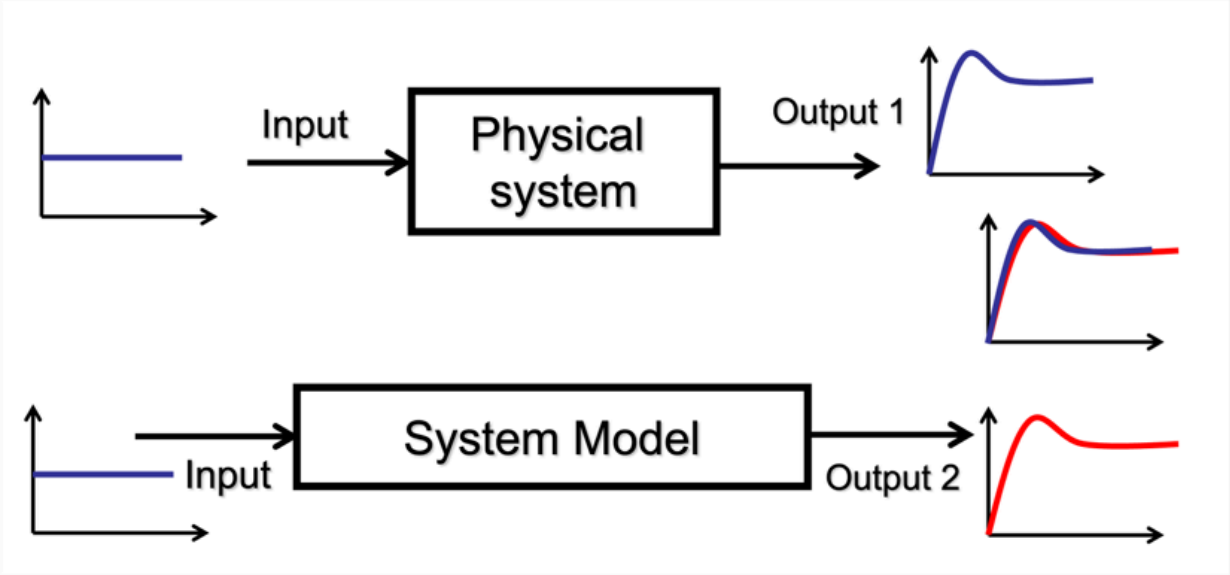


Figure 2.1: Physical system and its model responses.

The point mass-damper-spring mechanical system is often modeled on Newton's law. It first develops the free-body diagram and then applies Newton's law to obtain the corresponding linear or nonlinear differential equations. In addition, the energy conservation law can be applied to obtain equations of motion in the form of differential equations. If the model is linear, Laplace transformation [1] can be applied to obtain the transfer functions of the mechanical system that are ratios of input and output Laplace transforms. For the continuous mechanical system, such as a non-uniform beam, the finite element (FE) modeling technique is often used to get a linearized system model for control design and validation [2] and [3]; also see [4] for a Hubble telescope control design example. This chapter will briefly introduce how an FE model can be developed.

The electric/electronic system model can be developed based on Kirchhoff's voltage and current laws [5]. There are two approaches to be used: time and frequency domain approaches. As with mechanical system modeling, differential equations can be obtained by using Kirchhoff's voltage and current laws based on electric/electronic component differential equations for resistance, capacitance, and inductance. These electric/electronic components can also be modeled in the frequency domain in the form of their impedances. The frequency domain voltage is equal to the frequency domain current multiplying the associated component impedance. As a result, we can obtain algebraic equations in the form of Laplace transforms by applying Kirchhoff's voltage and current laws directly in the frequency domain. The advantage of the frequency domain approach is that no differential equations are involved. Instead, algebraic equations in the form of Laplace transforms are used. Therefore, in most cases of electric/electronic system models, it is preferred to adopt the frequency domain approach.

The mechatronic subsystem model is often used to describe a mechatronic component that either transfers an electric/electronic signal into mechanical motion (e.g., a DC motor) or converts mechanical motion into an electric/electronic signal (e.g., an electrical power generator or accelerometer sensor). Modeling the mechatronic subsystem often utilizes these tools developed for mechanical and electrical/electronic systems, resulting in linear differential equations or transfer functions. These models obtained based upon physics principles are called white-box models.

Physical systems are often nonlinear. In fact, we should say that all physical systems are nonlinear, and a linear system model approximates the physical one. Note that most system analysis and control design techniques are developed for linear systems due to their simplicity. In order to utilize these developed tools, we must often linearize the nonlinear system model to obtain a linear approximation.

In this case, Taylor expansion at the system operational condition of interest is often applied to the nonlinear differential equations obtained. By ignoring the high-order terms in the Taylor series or setting these terms to zero with order higher than one, we can find a linearized model (or a set of linear differential equations). Note that the linearized model is not equivalent to the nonlinear one. However, it provides a good approximation when the system is operated at or near the condition used for linearization. If the system will be run over a wide range of operational conditions, it is not a good idea to use the linearized model. In this case, the original nonlinear model should be used for control design and closed-loop validation studies.

There is a modeling approach that is between linear and nonlinear models; it is called a linear parameter-varying (LPV) system model. An LPV model describes the physical system using linear differential equations with their coefficients as a function of certain varying parameters. Many nonlinear systems can be formulated into LPV models [6], [7], and [8]. Many LPV control design methodologies have guaranteed performance. We recommend modeling the nonlinear system using the LPV approach so that LPV control methodologies, such as gain-scheduling control, can be applied.

Besides white-box models obtained based on physics as discussed above, gray- and black-box models can be obtained through system identification. The gray-box method assumes a structure made of system models, for instance, a second order spring-mass-damper system. The only model coefficients that need to be identified are ones such as mass, stiffness, and damping. With the model structure defined, Least-Squares (LS) method is often used to identify parameters. Machine learning methods are also popular; see [9] for an example.

A black-box model assumes no system model architecture, such as model order, and is modeled based upon system identification utilizing only specific system inputs and corresponding outputs. One of the black-box system identification methods is the *q*-Markov Cover [10], [11]. This method identifies a discrete linear time-invariant (LTI) system model. It is based on the system outputs excited by either white noise, impulse, or pseudo-random binary signals (PRBS). An analysis tool is used to select model order. Reference [11] also extends the *q*-Markov Cover to multi-rate discrete-time systems, where each input (or output) can be sampled at a different rate from others.

There are many other black-box system identification methods published [12]. Table 2.1 summarizes white-box (physics-based), gray-box, and black-box modeling approaches and associated methodologies, along with their resulting model types. In this chapter, we will mainly address the physics-based modeling approaches and provide outlines for gray- and black-box modeling approaches, with references for graduate level students.

Table 2.1: Summary of modeling approaches.

Physics-based model: white-box model	Hybrid approach: gray-box model	System identification approach: black-box model
Kirchhoff's LawNewton's LawNewton Euler equations	System parameter identification with known structure	System identification to get system order and parameters
Partial differential equationsOrdinary differential equations	Differential equationsAlgebraic equations	Nonlinear and linear modelsNeural-network modelsMachine-learning based model

Section 2.1 provides a brief review of the Laplace transform that will be used to describe a linear system model in the form of a transfer function. Section 2.2 presents mechanical system modeling, including both point mass-damper-spring and continuous mechanical systems. Next, Section 2.3 discusses the electric/electronic circuit modeling techniques in both time and frequency domains. Section 2.4 then addresses the modeling approach for mechatronic subsystems using a DC motor as an example. Section 2.5 shows how to linearize a nonlinear system model using Taylor expansion. Finally, Section 2.6 introduces basic concepts of gray- and black-box modeling techniques.

Orienting Questions

1. What are the basic building elements for modeling mechanical systems?
2. What are the basic building elements for modeling electric/electronic systems?
3. How does a DC motor work?
4. How is a DC motor modeled?
5. How is a nonlinear system model linearized?
6. What are the definitions of the gray- and black-box models?

This page titled 2.0: Prelude is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

2.1: Laplace Transforms

Learning Objectives

After completing this section, you will be able to:

1. Provide definitions of the Laplace transform and inverse Laplace transform.
2. Compute Laplace transforms for given time-domain functions.
3. Compute inverse Laplace transforms for given frequency-domain functions.

The Laplace transform is an integral transform that converts a function of a real value variable in the time domain to a function of a complex variable in the frequency domain, also known as the s-domain. The transform has many applications in engineering, especially as a tool for solving linear ordinary differential equations (ODEs). The main advantage of using the Laplace transform to solve ODEs is that it converts ODEs into algebraic equations, making them easier to solve. For a given real variable function $f(t)$ in the time domain, the Laplace transform is defined as follows.

$$F(s) \triangleq L[f(t)] = \int_0^\infty f(t)\, e^{-st} dt \tag{2.1.1}$$

Here, $F(s)$ is the Laplace transform of $f(t)$ and s is a complex variable. The inverse Laplace transform converts a complex value function in the frequency domain into a real value function in the time domain defined as follows.

$$f(t) \triangleq L^{-1}[F(s)] = \frac{1}{2\pi i} \lim_{T\to\infty} \int_{\gamma - iT}^{\gamma + iT} e^{st} F(s)\, ds \tag{2.1.2}$$

Here γ is a real variable. The integration is performed along the vertical line of $Re(s) = \gamma$ in the complex plane such that γ is greater than the real part of all singularities of $F(s)$ and $F(s)$ is bounded on the line. If all singularities are in the left half-plane, then γ can be set to zero; the above inverse integral formula then becomes identical to the inverse Fourier transform. In practice, this complex integral can be calculated by using the Cauchy residue theorem [13].

Example 2.1

Consider a unit step function $u(t)$ defined below.

$$u(t) \triangleq \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases} \tag{2.1.3}$$

Its Laplace transform can be calculated by

$$U(s) \triangleq L[u(t)] = \int_0^\infty e^{-st} dt = \frac{-1}{s} e^{-st}\Big|_0^\infty = \frac{1}{s} \tag{2.1.4}$$

Computing the inverse Laplace transform $U(s)$ is a little complicated based on equation 2.1.2. By applying the Cauchy residue theorem [13], we have

$$u(t) \triangleq L^{-1}[U(s)] = \frac{1}{2\pi i} lim_{T\to\infty} \int_{\gamma - iT}^{\gamma + iT} e^{st} F(s)\, ds \; = Res\left(\frac{1}{s}\right) = 1, \quad t \geq 0 \tag{2.1.5}$$

Because the Laplace transform is defined only over $t \geq 0$, for $t < 0$, we can set $u(t) = 0$, resulting in the original unit step function.

Calculating the inverse Laplace transform is relatively complicated when using definition from equation 2.1.2. However, in practice, the inverse Laplace transform is often based on the Laplace transforms seen in Table 2.2 and the Laplace transform properties seen in Table 2.3, with the help of fractional expansion. The unit impulse function $\delta(t)$ is defined in equation 2.1.7.

$$\delta(t) \triangleq \begin{cases} \infty, & 0 \le t \le 0^+ \\ 0, & \text{elsewhere} \end{cases} \tag{2.1.6}$$

and

$$\int_0^\infty \delta(t)\,dt = \int_0^{0^+} \delta(t)\,dt = 1 \tag{2.1.7}$$

Example 2.2

Consider a Laplace transform $F(s)$, defined below, that can be expanded using the partial fraction expansion technique.

$$F(s) = \frac{1}{(s+1)(s+2)} = \frac{1}{s+1} - \frac{1}{s+2} \tag{2.1.8}$$

Table 2.2: Laplace transforms of typical time-domain functions.

Description	$f(t),\ \ t \ge 0$	$F(s) \triangleq L[f(t)]$
Unit impulse	$\delta(t)$	1
Unit Step	$u(t)$	$\frac{1}{s}$
Unit ramp	t	$\frac{1}{s^2}$
n^{th} order ramp	t^n	$\frac{n!}{s^{n+1}}$
Exponential	e^{at}	$\frac{1}{s-a}$
Sinusoid	$\sin\omega t$	$\frac{\omega}{s^2+\omega^2}$
Cosine	$\cos\omega t$	$\frac{s}{s^2+\omega^2}$

Its inverse Laplace transform can be calculated by using the exponential property in Table 2.3.

$$f(t) \triangleq L^{-1}\left[\frac{1}{s+1} - \frac{1}{s+2}\right] = L^{-1}\left[\frac{1}{s+1}\right] - L^{-1}\left[\frac{1}{s+2}\right] = e^{-t} - e^{-2t} \tag{2.1.9}$$

Example 2.3

Consider the Laplace transform function $F(s)$ defined below.

$$F(s) = \frac{e^{-2s}}{s+2} \tag{2.1.10}$$

Its inverse Laplace transform can be calculated by using the time-shift ("t" shift) property in Table 2.3.

$$f(t) \triangleq L^{-1}\left[\frac{e^{-2s}}{s+2}\right] = e^{-2\tau}\big|_{\tau=t-2} \tag{2.1.11}$$

$$u(t-2) = u(t-2)e^{-2(t-2)} \tag{2.1.12}$$

where $u(t-2)$ is a unit step function with its time shifted by 2 seconds. That is,

$$u(t-2) \triangleq \begin{cases} 0, & t < 2 \\ 1, & t \ge 2 \end{cases} \tag{2.1.13}$$

One of the Laplace transform applications is to solve ordinary differential equations (ODEs), as seen in Example 2.4.

Table 2.3: Laplace transform properties.

Linearity	$L\left[af\left(t\right)+bg\left(t\right)\right]=aL\left[f\left(t\right)\right]+bL[g\left(t\right)]$
"s" shift property	$L\left[e^{at}f\left(t\right)\right]=F\left(s-a\right)=L[f\left(t\right)]\|_{s=s-a}$
"t" shift property	$L\left[f\left(t-a\right)u(t-a)\right]=e^{-as}F\left(s\right)=e^{-as}L[f\left(t\right)]$
Transform of derivatives	$L\left[\dot{f}(t)\right]=sF\left(s\right)-f(0)L\left[\ddot{f}(t)\right]=s^2F\left(s\right)-sf\left(0\right)-\dot{f}(0)L\left[\dddot{f}(t)\right]=s^3F\left(s\right)-s^2f\left(0\right)-s\dot{f}\left(0\right)-\ddot{f}(0)$:
Transform of integrals	$L\left[\int_0^t f\left(\tau\right)d\tau\right]=\frac{1}{s}L[f\left(t\right)]=\frac{1}{s}F(s)$
Convolution	$L^{-1}\left[F\left(s\right)G\left(s\right)\right]=\int_0^t f\left(\tau\right)g\left(t-\tau\right)d\tau=\int_0^t g\left(\tau\right)f\left(t-\tau\right)d\tau$

Example 2.4

Consider the following ODE with its initial condition $\dot{x}\left(0\right)=x\left(0\right)=0$.

$$\ddot{x}\left(t\right)+3\dot{x}\left(t\right)+2x(t)=f(t) \tag{2.1.14}$$

Applying the Laplace transform to both sides of equation 2.1.14 and using the derivative property in Table 2.3 yields the following.

$$X\left(s\right)\left(s^2+3s+2\right)=F(s) \tag{2.1.15}$$

or

$$X\left(s\right)=\frac{F(s)}{s^2+3s+2} \tag{2.1.16}$$

Assuming $F\left(s\right)=1$ to be a unit impulse ($f\left(t\right)=\delta\left(t\right)=L^{-1}[F(s)]$), the time response $x\left(t\right)=L^{-1}[X(s)]$ can be solved by using the inverse Laplace transform below, as based on Example 2.2.

$$L^{-1}\left[X\left(s\right)\right]=L^{-1}\left(\frac{1}{s^2+3s+2}\right)=L^{-1}\left(\frac{1}{s+1}-\frac{1}{s+2}\right)=e^{-t}-e^{-2t} \tag{2.1.17}$$

Laplace transforms significantly simplify the process of solving ODEs; they convert the problem to solving an algebraic equation in the frequency domain. Before concluding this section, let's look at the concept of the *transfer function* that is often used to represent a linear system model. Let the input of a mass-damper-spring system be applied force $f(t)$ and associated output be displacement $x\left(t\right)$. Assuming that the mass equals 1, the damping coefficient equals $3/(2\sqrt{2})$, and spring stiffness equals 2, the resulting equation of motion (EOM) is an ordinary differential equation, as defined in equation 2.1.14, and using the Laplace transform of equation 2.1.15. We can define the system transfer function $G\left(s\right)$, representing the linear system model, as

$$G\left(s\right)=\frac{X(s)}{F(s)}=\frac{1}{s^2+3s+2} \tag{2.1.18}$$

where $F(s)$ is the Laplace transform of system input, $X(s)$ is the Laplace transform of the system output, and $G\left(s\right)$ is the system transfer function. Note that for any transfer function, the system initial condition is assumed to be zero. When $G(s)$ is known, the system response $x(t)$ can be solved by taking the inverse Laplace transform of $G\left(s\right)F(s)$.

Section Summary

This section addresses the Laplace transform for analog signals in continuous-time domain as well as typical transforms for impulse, step, ramp, exponential, sine and cosine functions. The section also discusses Laplace transform properties such as delay, linearity, time-shift, derivative, and integral. In addition, the inverse Laplace transform is introduced along with the fractional expansion method, with its application for conducting inverse Laplace transform.

Section Keywords

Practice Problems

1. Find the Laplace transforms of the following time domain functions using Tables 2.2 and 2.3.
   a. $f(t) = 2e^{-2t} + e^{-3t}$
   b. $g(t) = (2t - 6)\,u(t-3)$, where $u(t)$ is a unit step function.
   c. $f(t) = 2e^{-2t}u(t-2)$, where $u(t)$ is a unit step function.
   d. $f(t) = 10e^{-5t}\cos(t-\pi)\ \ u(t-\pi)$
2. Find the inverse Laplace transforms of the following frequency domain functions using Tables 2.2 and 2.3.
   a. $Y(s) = \dfrac{16}{s(s+2)}$
   b. $Y(s) = \dfrac{s+1}{s^2+2s+5}$
   c. $F(s) = \dfrac{1}{(s+2)^2} + \dfrac{1}{(s+2)^3}$
   d. $F(s) = e^{-3s}\dfrac{s}{s^2+2^2}$

This page titled 2.1: Laplace Transforms is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

2.2: Mechanical System Modeling

Learning Objectives

After completing this section, you will be able to:

1. Describe three key components of mechanical subsystems: mass (or moment of inertial), damper (rotational damper), and spring (rotational spring).
2. Derive differential equations based upon Newton's law for linear and rotational dynamic motions.
3. Model continuous mechanical systems based on finite element analysis.

Mechanical systems are often modeled using differential equations first. The system transfer function can be obtained by applying the Laplace transform to these differential equations. In this section, we discuss how to a model point mass-damper-spring system based on Newton's law. We then briefly discuss how to model continuous mechanical systems using the finite element method. Note that there is another class of mechanical systems—such as fluid dynamic systems—that is often modeled by partial differential equations.

For the point mass-damper-spring system with linear motion, there are three basic components: mass, damper, and spring. We will first model these components individually. Consider that a force $f(t)$ is applied to a point mass m, resulting in a displacement $x(t)$. The left plot of Figure 2.2 illustrates the free-body diagram.

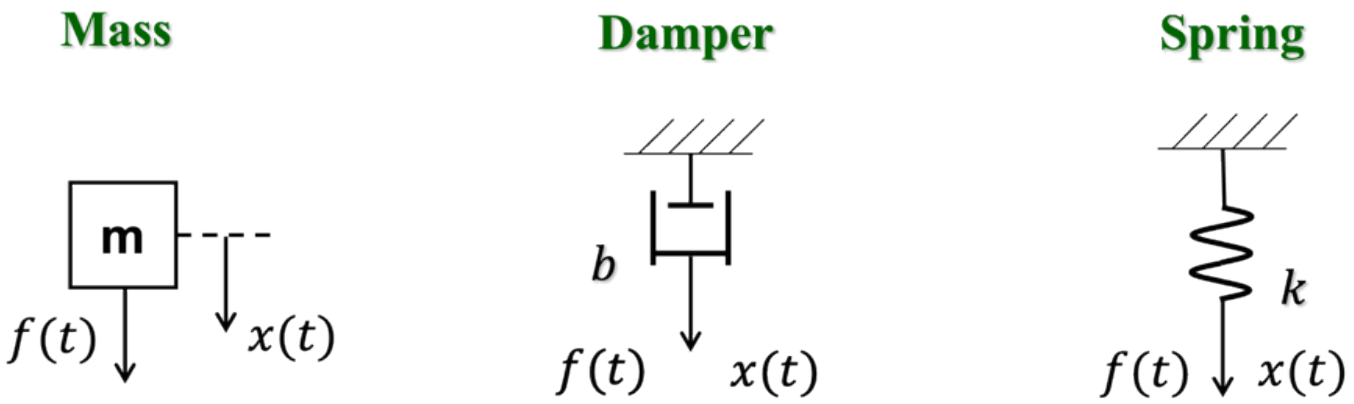


Figure 2.2: Point mass-damper-spring free body diagrams.

Applying Newton's law results in the following equation of motion for mass m.

$$m\ddot{x}\left(t\right)=f(t) \tag{2.2.1}$$

Assuming that $\dot{x}\left(0\right)=x\left(0\right)=0$, equation 2.2.2 shows the Laplace transform of equation 2.2.1.

$$ms^{2}X\left(s\right)=F\left(s\right),\ \frac{X(s)}{F(s)}=\frac{1}{ms^{2}} \tag{2.2.2}$$

For the damper (the middle plot of Figure 2.2), using Newton's law yields the following equation of motion.

$$b\dot{x}\left(t\right)=f(t) \tag{2.2.3}$$

In turn, equation 2.2.4 shows the associated Laplace transform of equation 2.2.3, assuming $x\left(0\right)=0$.

$$bsX\left(s\right)=F\left(s\right),\ \frac{X\left(s\right)}{F\left(s\right)}=\frac{1}{bs} \tag{2.2.4}$$

For the spring (the right plot of Figure 2.2), utilizing Newton's law leads to the following equation of motion.

$$kx\left(t\right)=f(t) \tag{2.2.5}$$

Equation 2.2.6 shows the associated Laplace transform of equation 2.2.5.

$$kX\left(s\right)=F\left(s\right),\ \frac{X(s)}{F(s)}=\frac{1}{k} \tag{2.2.6}$$

Now we are ready to discuss the process of modeling a point mass-damper-spring system to obtain associated ordinary differential equations and transfer functions. The process can be divided into the following three steps: Using the force relationships discussed above, draw the free-body diagram for the target mass-damper-spring system. Using Newton's law ($m\ddot{x}$ $=$ summation of all applied forces), obtain the associated ordinary differential equations. Compute the Laplace transform on the ordinary differential equations to obtain the associated system transfer functions. The following examples demonstrate linear motion mass-damper-spring systems.

Example 2.5

Consider the simple mass-damper-spring system shown in Figure 2.3. Find the equation of motion (EOM) in the form of the ordinary differential equation and the associated transfer function.

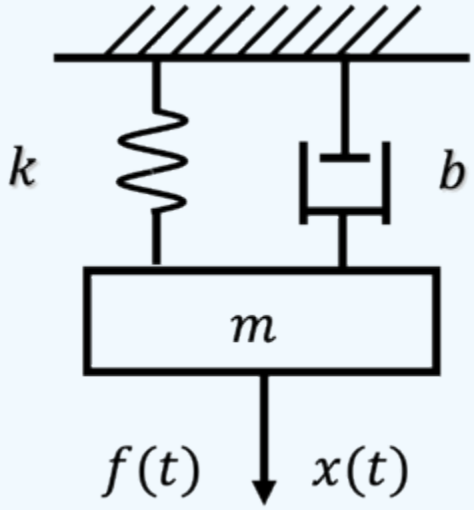


Figure 2.3: Point mass-damper-spring system for Example 2.5.

Draw the free-body diagram

To draw the free-body diagram for the system shown in Figure 2.3, we must find the damping and spring forces applied to the mass m. Using equations 2.2.3 and 2.2.5 results in the following.

$$f_b\left(t\right)=b\dot{x}\left(t\right),f_k\left(t\right)=kx(t) \tag{2.2.7}$$

With damping force $f_b\left(t\right)$ and spring force $f_k\left(t\right)$ calculated in equation 2.2.7, we can draw a free body diagram (see the right diagram in Figure 2.4).

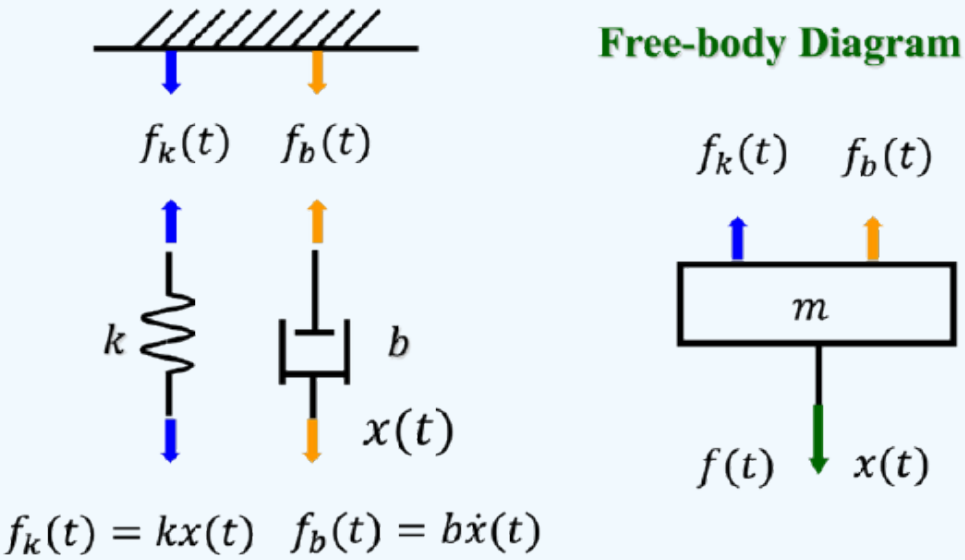


Figure 2.4: Force analysis and free-body diagram for Example 2.5.

Find the equation of motion (EOM)

Applying Newton's law yields

$$m\ddot{x}(t) = f(t) - f_b(t) - f_k(t) = f(t) - b\dot{x}(t) - kx(t) \tag{2.2.8}$$

where the force in the same direction of the displacement $x(t)$ is positive and vice versa. The resulting EOM is shown below.

$$m\ddot{x}(t) + b\dot{x}(t) + kx(t) = f(t) \tag{2.2.9}$$

Find the Laplace transform of EOM

Computing the Laplace transform for equation 2.2.9, assuming zero initial condition, leads to the following.

$$ms^2X(s) + bsX(s) + kX(s) = F(s) \tag{2.2.10}$$

The associated system transfer function with $F(s)$ as input and $X(s)$ as output is shown below.

$$G(s) = \frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + k} \tag{2.2.11}$$

Example 2.6

Consider a quarter-car driving on an uneven road with varying vertical displacement $h(t)$ (see Figure 2.5). Here, the quarter-car and wheel masses are m_b and m_w, respectively. The suspension damping coefficient and spring stiffness are b_s and k_s, respectively.

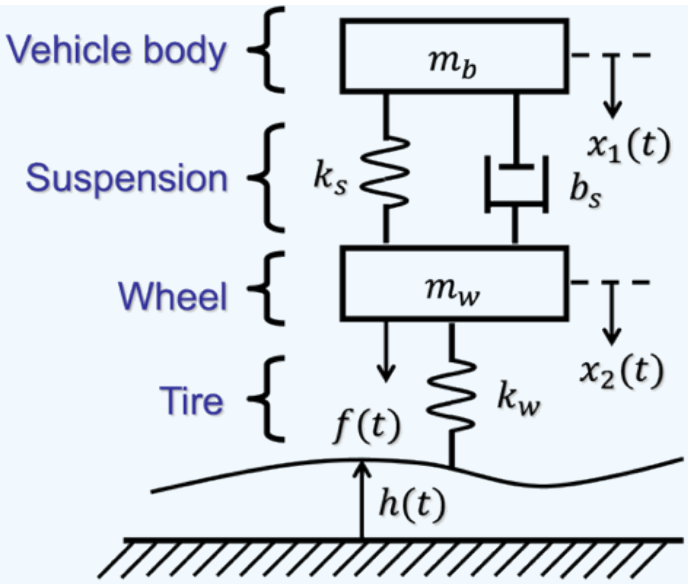


Figure 2.5: Quarter-car driven on an uneven road surface.

The active suspension applied force and tire spring stiffness are $f(t)$ and k_w, respectively. Find the equations of motion and associated transfer functions. Follow the three steps described in Example 2.5 to find the EOM and associated transfer function. First, draw the free-body diagrams for m_b and m_w (see Figure 2.6). There are two point-masses in the system, resulting in two free-body diagrams.

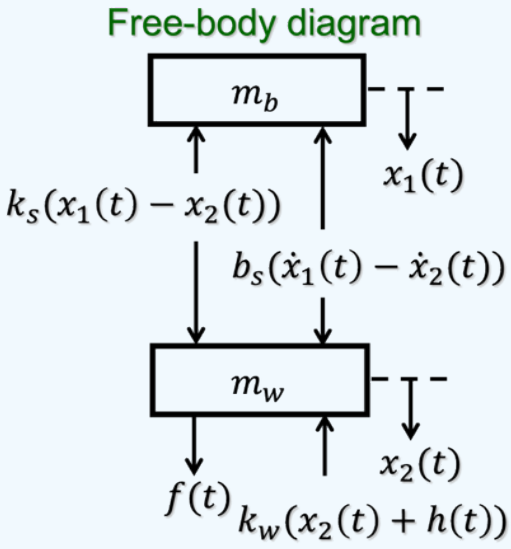


Figure 2.6: Free-body diagrams for a quarter-car driven on an uneven road surface.

With two free-body diagrams developed for m_b and m_w, respectively, applying Newton's law results in the following two differential equations of motion.

$$\begin{aligned} m_b\ddot{x}_1(t) + b_s(\dot{x}_1(t) - \dot{x}_2(t)) + k_s(x_1(t) - x_2(t)) &= 0 \\ m_w\ddot{x}_2(t) + b_s(\dot{x}_2(t) - \dot{x}_1(t)) + k_s(x_2(t) - x_1(t)) + k_w x_2(t) &= f(t) - k_w h(t) \end{aligned} \tag{2.2.12}$$

In turn, conducting the Laplace transform for equation 2.2.12 yields the following, assuming zero initial condition

$$x_1(0) = \dot{x}_1(0) = x_2(0) = \dot{x}_2(0) = 0. \tag{2.2.13}$$

$$m_b s^2 X_1(s) + b_s s X_1(s) + k_s X_1(s) = b_s s X_2(s) + k_s X_2(s)$$
$$m_w s^2 X_2(s) + b_s s X_2(s) + k_s X_2(s) + k_w X_2(s) = F(s) - k_w H(s) + b_s s X_1(s) + k_s X_1(s) \tag{2.2.14}$$

Now, reorganize equation 2.2.14 to obtain the system transfer functions as below.

$$G_1(s) = \frac{X_1(s)}{X_2(s)} = \frac{b_s s + k_s}{m_b s^2 + b_s s + k_s}$$
$$X_2(s) = \frac{F(s)}{m_w s^2 + b_s s + (k_s + k_w)} - \frac{k_w H(s)}{m_w s^2 + b_s s + (k_s + k_w)} + \frac{(b_s s + k_s) X_1(s)}{m_w s^2 + b_s s + (k_s + k_w)} \tag{2.2.15}$$
$$= G_2(s) F(s) - k_w G_2(s) H(s) + G_3(s) X_1(s)$$

The transfer functions in equation 2.2.15 can be represented by the block diagram in Figure 2.7. (Section 2.4 will provide a detailed technique for drawing block diagrams.)

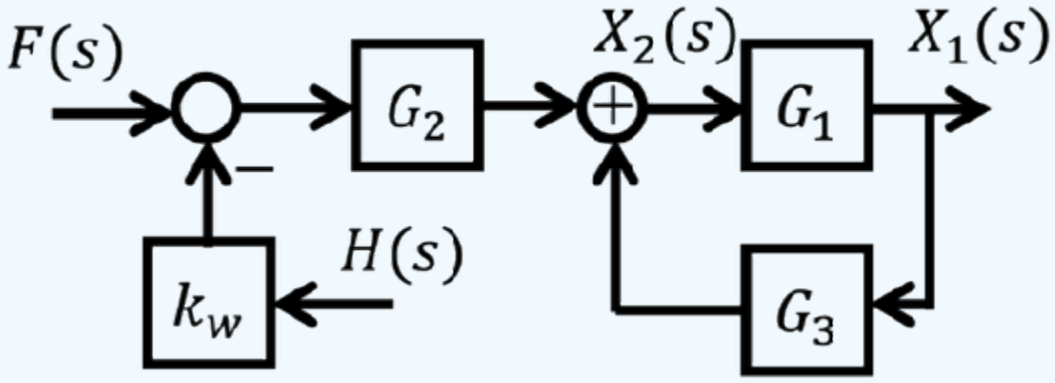


Figure 2.7: System transfer function diagrams for a quarter-car driven on an uneven road surface.

There is another class of mechanical systems with rotational motion, where the moment of inertia J is substituted for mass m, a rotational damper for the linear damper, and a rotational spring for the linear spring. Figure 2.8 shows the free-body diagrams of the moment of inertia, rotational damper, and rotational spring. Using a similar approach to linear motion, we can establish torque τ and rotational displacement θ relationships for the moment of inertial, rotational damper, and rotational spring.

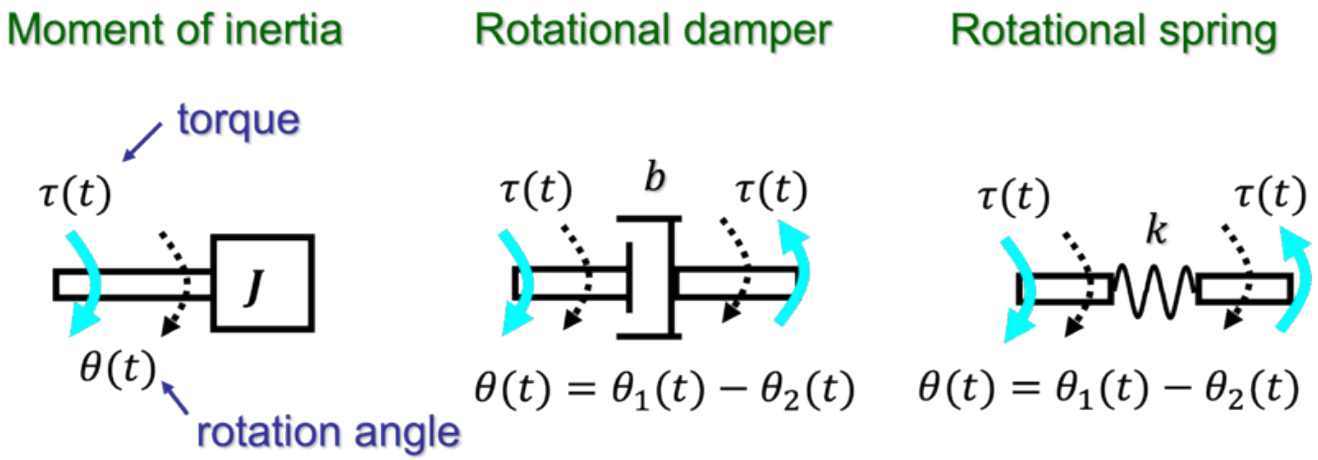


Figure 2.8: Rotational point mass-damper-spring free body diagrams.

For the moment of inertia (the left drawing of Figure 2.8), applying Newton's law results in the following equation of motion:

$$J\ddot{\theta}(t) = \tau(t) \tag{2.2.16}$$

Now assuming that $\dot{\theta}(0) = \theta(0) = 0$, the Laplace transform of equation 2.2.16 is shown below.

$$Js^2\Theta(s) = T(s), \frac{\Theta(s)}{T(s)} = \frac{1}{Js^2} \tag{2.2.17}$$

For the rotational damper (the middle drawing of Figure 2.8), using Newton's law yields the following equation of motion:

$$b\dot{\theta}(t) = \tau(t) \tag{2.2.18}$$

In turn, the associated Laplace transform of equation 2.2.18, assuming $x(0) = 0$, is show below.

$$bs\Theta(s) = T(s), \frac{\Theta(s)}{T(s)} = \frac{1}{bs} \tag{2.2.19}$$

For the rotational spring (the right drawing of Figure 2.8), utilizing Newton's law leads to the following equation of motion:

$$k\theta(t) = \tau(t) \tag{2.2.20}$$

The associated Laplace transform of equation 2.2.20 is as follows.

$$k\Theta(s) = T(s), \frac{\Theta(s)}{T(s)} = \frac{1}{k} \tag{2.2.21}$$

By replacing the linear mass m, damping coefficient b, and spring stiffness k with moment of inertia J, rotational damping coefficient b, and spring stiffness k, respectively, we can see that these three EOMs are similar to those for linear motion.

Example 2.7

Consider a motor with motor torque $\tau_m(t)$, moment of inertia J_m, rotational friction b_m, and angular displacement $\theta_m(t)$ that drives the load moment of inertia J_l through a flexible shaft with its rotational stiffness k; see Figure 2.9.

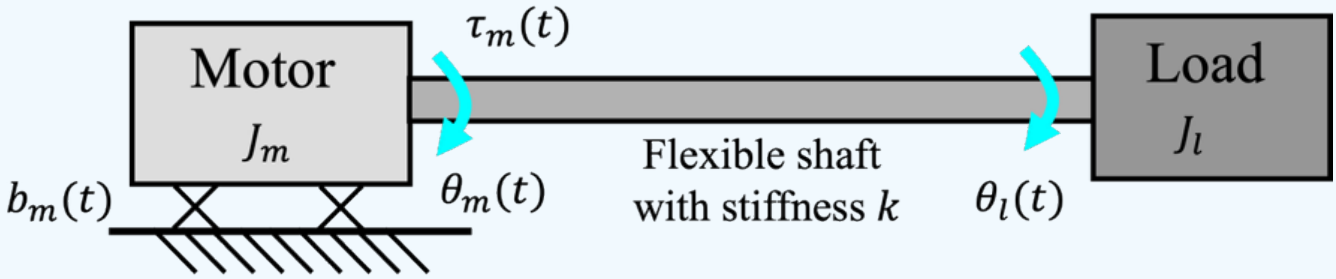


Figure 2.9: Motor and load system.

To find the equations of motion and associated transfer functions, we will follow the three steps described in Example 2.5. First, draw the free-body diagrams for J_m and J_l (see Figure 2.10). Because there are two subsystems of moment of inertia, we have two free-body diagrams.

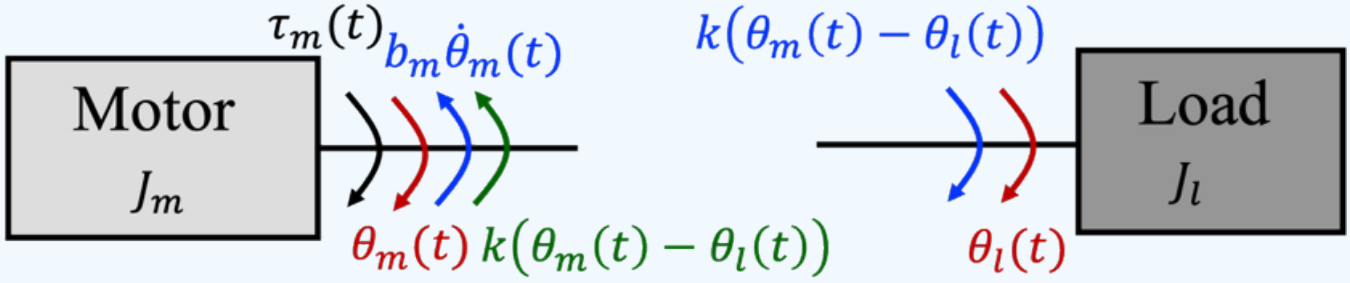


Figure 2.10: Free-body diagrams of motor and load system.

With the two free-body diagrams developed for J_m and J_l, applying Newton's law results in the following two equations of motion.

$$J_m\ddot{\theta}_m(t) + b_m\dot{\theta}_m(t) + k(\theta_m(t) - \theta_l(t)) = \tau_m(t)$$
$$J_l\ddot{\theta}_l(t) + k(\theta_l(t) - \theta_m(t)) = 0 \tag{2.2.22}$$

Computing the Laplace transform for equation 2.2.22 yields the following, assuming zero initial condition

$$(\theta_m(0) = \dot{\theta}_m(0) = \theta_l(0) = \dot{\theta}_l(0) = 0. \tag{2.2.23}$$

$$J_m s^2\Theta_m(s) + b_m s\Theta_m(s) + k(\Theta_m(s) - \Theta_l(s)) = T_m(s)$$
$$J_l s^2\Theta_l(s) + k(\Theta_l(s) - \Theta_m(s)) = 0 \tag{2.2.24}$$

Now, reorganize equation 2.2.24 to obtain the system transfer functions, as in equation 2.2.25.

$$\Theta_m(s) = \frac{T_m(s) + k\Theta_l(s)}{J_m s^2 + b_m s + k}$$
$$\Theta_l(s) = \frac{k\Theta_m(s)}{J_l s^2 + k} \tag{2.2.25}$$

To obtain the system transfer function from $T_m(s)$ to $\Theta_l(s)$, substitute the top equation of 2.2.25 into the bottom equation of 2.2.25. Similarly, find the transfer function from $T_m(s)$ to $\Theta_m(s)$ by substituting the bottom equation of 2.2.25 into the top equation of 2.2.25.

Having discussed modeling of point mass-damper-spring mechanical systems, we are now going to briefly discuss modeling of the continuous mechanical system, using an example. In general, we can use the finite element analysis method to find a linearized model, as mentioned at the beginning of this section. To demonstrate how to use the finite-element method to model a continuous mechanical system, consider a flexible beam interacting with a uniform flow field (the left diagram of Figure 2.11). The finite-element dynamic modeling approach can be adopted in this case. The beam is divided into n rigid-body elements (see the right diagram in Figure 2.11). These elements have equal length. Each is linked with its neighbor elements through joints modeled by a rotational spring k_i and a linear rotational damper b_i, where index i ($i = 1,\ 2,\ \ldots,\ n+1$) denotes the beam node and the ith element is defined by the nodes i and $i+1$. These dampers on the node collectively model the internal damping of the beam material and the hydrodynamic damping due to the interaction with the surrounding fluid (see the right diagram of Figure 2.11). In this case, these dampers could be nonlinear; however, in this section, they are assumed to be linear (or b_i is constant). Applied force f_i^f to the ith beam is generated by the uniform flow field, which is modeled as a lumped load force applied to the element center of mass.

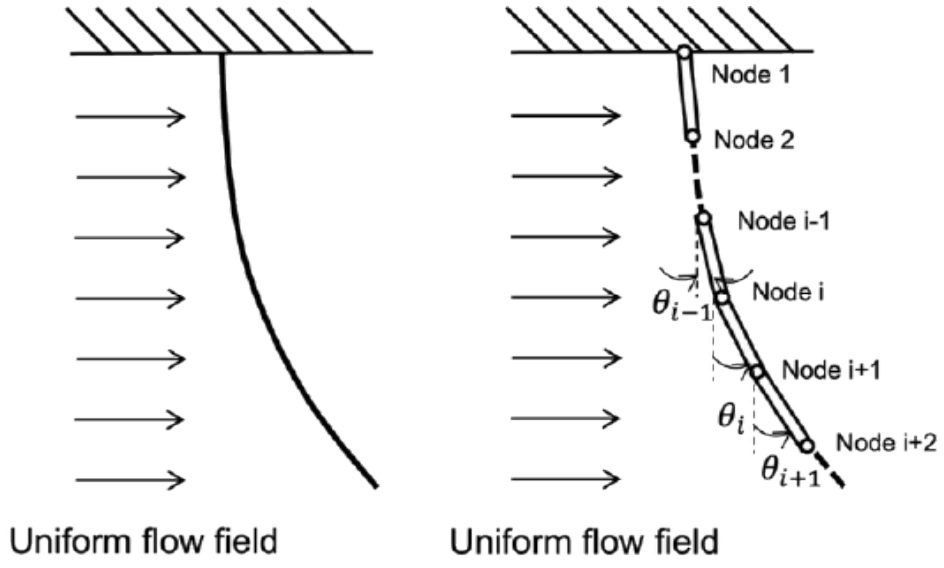


Figure 2.11: A Cantilever beam modeled by a finite number of rigid elements.

Figure 2.12 shows the free-body diagram for the ith element, where (x, y) are the coordinates in the inertial frame, J_i is the moment of inertia for the ith element, l is the beam element length, f_i^x and f_i^y are respectively the reaction forces at the ith node in the

x and y directions, and f_{i+1}^x and f_{i+1}^y are the reaction forces for the $(i+1)^{th}$ node in x and y directions. Also, the solid dot in the middle of the element represents the center of mass for the ith element.

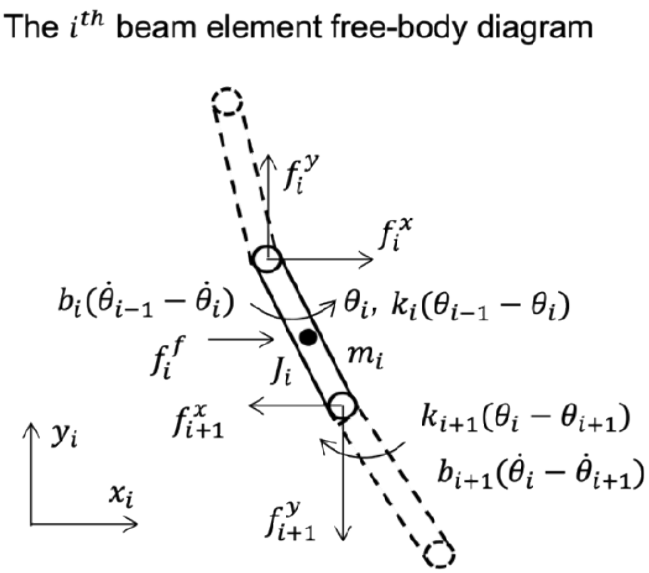


Figure 2.12: Free-body diagram of the ith element.

For the ith element, the governing equation for the rotational motion can be written in the following form based upon Newton's law.

$$\begin{aligned} J_i\ddot{\theta}_i + b_i\left(\dot{\theta}_i - \dot{\theta}_{i-1}\right) + k_i\left(\theta_i - \theta_{i-1}\right) - b_{i+1}\left(\dot{\theta}_{i+1} - \dot{\theta}_i\right) - k_{i+1}\left(\theta_{i+1} - \theta_i\right) = \\ -\frac{f_i^x}{2}l\cos\theta_i - \frac{f_i^y}{2}l\sin\theta_i - \frac{f_{i+1}^x}{2}l\cos\theta_i - \frac{f_{i+1}^y}{2}l\sin\theta_i \end{aligned} \tag{2.2.26}$$

Note that equation 2.2.26 is valid for $i = 1,\ 2, \ldots, n-1$. For $i = n$ (the last element), we need to set $f_{n+1}^x = f_{n+1}^y = 0$. Using Newton's law, the translational motion satisfies the following two equations.

$$\begin{aligned} m_i\ddot{x}_i(t) &= f_i^f(t) + f_i^x(t) - f_{i+1}^x(t) \\ m_i\ddot{y}_i(t) &= f_i^y(t) - f_{i+1}^y(t) \end{aligned} \tag{2.2.27}$$

where m_i is the effective mass of the ith beam element defined in Figure 2.12, and $\ddot{x}_i$ and $\ddot{y}_i$ are the mass acceleration in x and y directions at the center of mass m_i. Note that equation 2.2.26 is nonlinear. To obtain a linear system model, we assume that the bending angle θ_i $(i = 1,\ 2,\ \ldots,\ n)$ is small so that

$$\sin\theta_i \approx \theta_i, \cos\theta_i \approx 1,\ i = 1,\ 2,\ \ldots,\ n. \tag{2.2.28}$$

This approximation leads to a maximum error of 1.5% when the beam angle θ_i $(i = 1,\ 2,\ \ldots,\ n)$ is less than or equal to 10 degrees (or 0.17 rad). With this assumption, both $\ddot{x}_i$ and $\ddot{y}_i$ can be approximated by the following equations.

$$\begin{aligned} \ddot{x}_i(t) &= \sum_{j=1}^{i-1}\ddot{\theta}_j(t)l + \frac{\ddot{\theta}_i(t)l}{2},\ i = 1, 2, \ldots, n \\ \ddot{y}_i(t) &= 0,\ i = 1, 2, .., n \end{aligned} \tag{2.2.29}$$

Because $f_{n+1}^y(t)$ is zero, we have $f_i^y(t) = 0$ $(i = 1,\ 2,\ \ldots,\ n)$ from equations 2.2.27 and 2.2.29; see [3] for details. On the other hand, from equations 2.2.27 and 2.2.29, $f_i^x(t)$ in 2.2.26 can be solved as a linear combination of $f_i^f(t)$ and $\ddot{\theta}_i(t)$ $(i = 1,\ 2,\ \ldots,\ n)$ with $f_{n+1}^x(t) = 0$. The flow field force $f_i^f(t)$ applied to each element is exerted by the surrounding medium; it can be expressed as

$$f_i^f(t) = wlC_d\rho\frac{\tilde{V}_i^2}{2} \tag{2.2.30}$$

where w is the beam element width, l is the beam element length, C_d is the beam drag coefficient, ρ is the fluid density, and $\tilde{V}_i$ is the relative beam velocity with respect to the fluid flow field at the center of the element. Note that $\tilde{V}_i = V_o + \dot{x}_i + l\dot{\theta}_i/2$, where V_o is the fluid flow field velocity, and $\dot{x}_i + l\dot{\theta}_i/2$ is the x-direction component of the beam element linear velocity at its center of mass. Again, assuming that $\dot{\theta}_i$ is relatively small, we can approximate $\dot{x}_i + l\dot{\theta}_i/2$ by $\dot{x}_i$. For the ith element, the moment of inertia J_i, rotational stiffness k_i, and damping coefficient b_i can be expressed in terms of the beam dimensions, the properties of the beam material, and the fluid property as follows, assuming that each beam element has the same mechanical properties.

$$J_i = \frac{m_i l^2}{12} \approx \frac{m_e l^2}{12} = J_e \tag{2.2.31}$$

$$k_i = \frac{\gamma w c^3}{12 l} = k_e \tag{2.2.32}$$

$$b_i = 2\xi\sqrt{J_i k_i} = 2\xi\sqrt{J_e k_e} = b_e \tag{2.2.33}$$

where $m_i = m_e$ is the effective mass for element i; w and c are the beam element width and thickness, respectively; γ is the (effective) Young's modulus of the beam material, and ξ is the critical damping ratio accounting for material and hydrodynamic damping. In addition, note that effective mass m_e represents the sum of the actual mass of element i and the added mass due to beam-fluid interactions [14], [15]. It can be expressed as

$$m_e = m_i + \rho_e w l c \tag{2.2.34}$$

where ρ_e represents the effective density of the beam, which depends on the material density, fluid density, and beam geometry. Because all the beam elements are identical, they have the same moment of inertia J_e, damping coefficient b_e, and stiffness k_e. We can rewrite equation 2.2.26 in a compact matrix form below.

$$J\ddot{\bar{\theta}}(t) + B\dot{\bar{\theta}}(t) + K\bar{\theta}(t) = \Gamma \tag{2.2.35}$$

where J, B, and K are matrices of moments of inertia, damping coefficients, and spring constants, respectively. For the *n*-element beam model, these matrices are expressed as below.

$$J = J_e \begin{bmatrix} 12(n-1)+4 & 12(n-2)+6 & \cdots & 30 & 18 & 6 \\ 12(n-2)+6 & 12(n-2)+4 & \cdots & 30 & 18 & 6 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ & & & 28 & 18 & 6 \\ 30 & 30 & \cdots & 18 & 16 & 6 \\ 18 & 18 & \cdots\ 6\ \ 6\ \cdots & 6 & 6 & 4 \end{bmatrix} \tag{2.2.36}$$

$$B = b_e \begin{bmatrix} 2 & -1 & 0 & & & \cdots & 0 & 0 & & & \\ -1 & 2 & -1 & & & \cdots & 0 & 0 & & & \\ 0 & -1 & 2 & & & \cdots & 0 & 0 & & & \\ \vdots & \vdots & \vdots & & & & & & & & \\ 0 & 0 & 0 & \ddots & \vdots & \vdots\ \cdots & 2 & -1 & \cdots & -1 & 2 \\ 0 & 0 & 0 & & & & & & & & \end{bmatrix} \tag{2.2.37}$$

$$K = k_e \begin{bmatrix} 2 & -1 & 0 & & & \cdots & 0 & 0 & & & \\ -1 & 2 & -1 & & & \cdots & 0 & 0 & & & \\ 0 & -1 & 2 & & & \cdots & 0 & 0 & & & \\ \vdots & \vdots & \vdots & & & & & & & & \\ 0 & 0 & 0 & \ddots & \vdots & \vdots\ \cdots & 2 & -1 & \cdots & -1 & 2 \\ 0 & 0 & 0 & & & & & & & & \end{bmatrix} \tag{2.2.38}$$

The angular displacement vector $\bar{\theta}$ and the fluid flow-induced torque vector Γ are expressed as

$$\overline{\theta}(t)=\begin{bmatrix}\theta_1(t)\\ \theta_2(t)\\ \vdots\\ \theta_{n-1}(t)\\ \theta_n(t)\end{bmatrix},\ \Gamma(t)=\frac{l}{2}\begin{bmatrix}f_1^f(t)+2\sum_{j=2}^{n}f_j^f(t)\\ f_2^f(t)+2\sum_{j=3}^{n}f_j^f(t)\\ \vdots\\ f_{n-1}^f(t)+2\sum_{j=n}^{n}f_j^f(t)\\ f_n^f(t)\end{bmatrix} \tag{2.2.39}$$

It is a bit complex to model a continuous mechanical system such as a cantilever beam. The purpose of this example is to show that the model can be developed by using finite-element analysis. You can find a detailed application of this example in [3]. Another example for a continuous mechanical system model, a Euler-Bernoulli beam with a small pin-joint plate, can be found in [16].

Section Summary

This section considers the modeling process for mechanical point-mass system with spring, damper and mass. It addresses both linear and rotational motion systems with multiple degree-of-freedoms (DOF), based on Newton's law. The section also presents the process of deriving the system transfer function in the form of Laplace transforms. Finally, we model mechanical systems with continuous dynamics, using the finite element analysis (FEA) method; the example for the graduate class uses a beam in a one-direction fluid flow field.

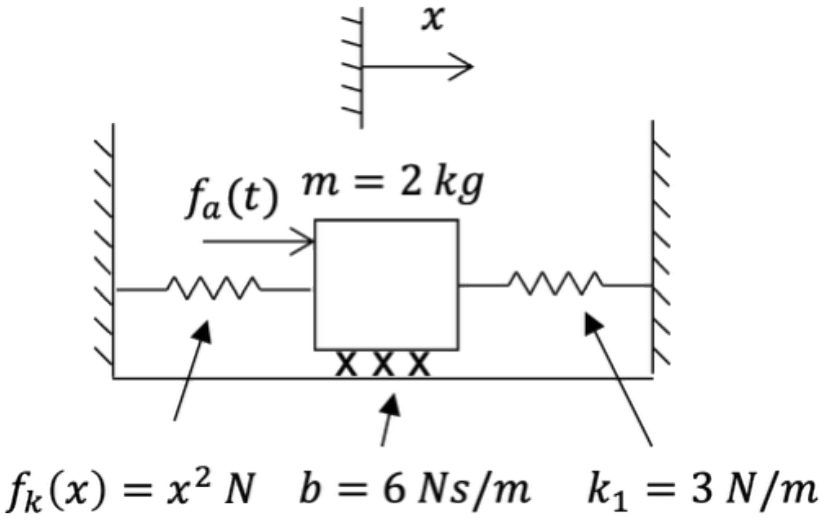
Section Keywords

Practice Problems

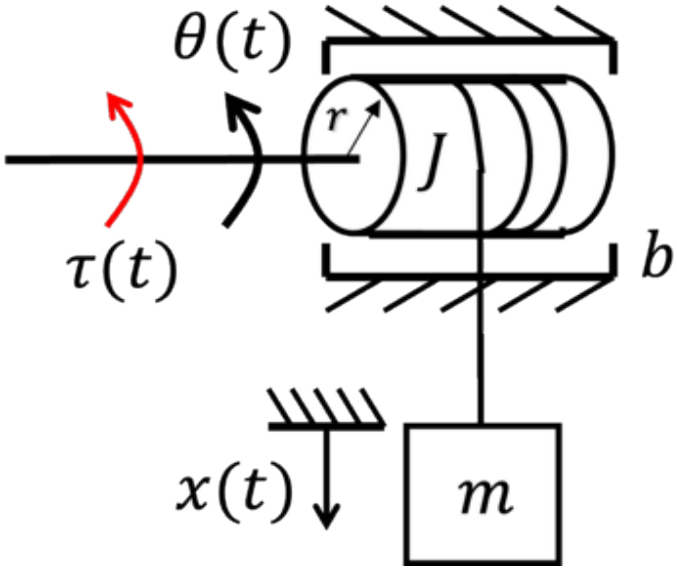
1. For the system diagram of a quarter-car on even road surface, shown below, using the three-step process described in Section 2.2, find the equations of motion in the form of ordinary differential equations and the system transfer functions.

2. Assume that the translational problem shown below has a linear and a nonlinear spring (k_1 and f_x) subjected to the applied force $f_a(t)$. Show that the nonlinear system model is represented by the following differential equation.

$$2\ddot{x}+6\dot{x}+3x+x^2=f_a(t)$$



3. Considering the following mechanical system, find the equations of motion (EOM) as a function of θ and x. Also find the transfer functions from input torque τ to angular displacement θ and linear displacement x.



4. For the linear cantilever beam model shown in equation 2.2.35, find parameter matrices J, B, and K, along with the forcing term Γ(t) for a five-element model ($n = 5$).

This page titled 2.2: Mechanical System Modeling is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

2.3: Electrical and Electronic System Modeling

Learning Objectives

After completing this section, you will be able to:

1. Describe three key components of electrical/electronic systems: resistor, capacitor, and inductor.
2. Introduce the impedances of resistors, capacitors, and inductors.
3. Develop electrical/electronic circuit models using Kirchhoff's law.

Electric and electronic systems are often modeled using differential equations and impedance algebraic equations. In this section we will discuss how to model resistance, capacitance, and inductance systems based on Kirchhoff's voltage and current laws, where circuit differential equations can be generated. These key circuit component differential equations (resistor, capacitor, and inductor) can also be converted into the associated Laplace transforms, called impedances. The resulting model equations will become algebraic in the form of transfer functions. We will also introduce modeling an operational amplifier circuit.

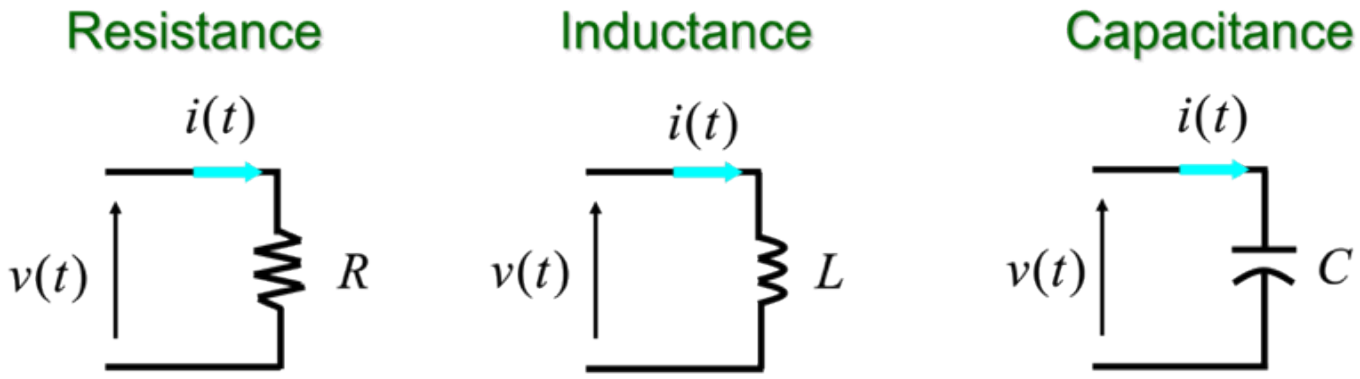


Figure 2.13: Electrical and electronic key components.

Figure 2.13 shows the three basic electrical components: resistance, capacitance, and inductance, with their electric/electronic relationships: voltage $v\,(t)$ and current $i(t)$.

Kirchhoff's Laws

The algebraic sum of voltage drops around any circuit loop is zero; the algebraic sum of currents into any junction is zero.

Applying Kirchhoff's voltage law to the left circuit of Figure 2.13 results in the following equation for resistance.

$$Ri\,(t) = v(t) \tag{2.3.1}$$

The Laplace transform of equation 2.3.1 is

$$RI\,(s) = V\,(s)\,,\ \frac{V(s)}{I(s)} = R \tag{2.3.2}$$

Note that, in the frequency domain, R also represents impedance for resistance. For the inductance, using Kirchhoff's voltage law for the middle circuit in Figure 2.13 yields the following differential equation.

$$L\frac{di(t)}{dt} = v(t) \tag{2.3.3}$$

Equation 2.3.3 indicates that the inductor current cannot change suddenly because its rate of change is equal to its voltage. The associated Laplace transform of equation 2.3.3, assuming $i\,(0) = 0$, is

$$LsI(s) = V(s), \quad \frac{V(s)}{I(s)} = Ls \tag{2.3.4}$$

where Ls is called the impedance of the inductor. Now for the capacitance, utilizing Kirchhoff's voltage law for the right circuit of Figure 2.13 leads to the following differential equation.

$$v(0) + \frac{1}{C}\int_0^t i(\tau)\,d\tau = v(t) \tag{2.3.5}$$

The associated Laplace transform of integral equation 2.3.5, assuming $v(0) = 0$, is

$$\frac{1}{Cs}I(s) = V(s), \frac{V(s)}{I(s)} = \frac{1}{Cs} \tag{2.3.6}$$

where $1/(Cs)$ is called the impedance of the capacitor.

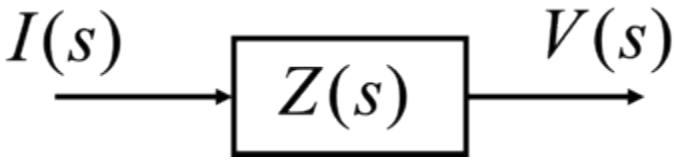


Figure 2.14 The relationship among the frequency domain current $I(s)$, voltage $V(s)$, and impedance $Z(s)$.

Note that in the frequency domain, voltage $V(s)$ always equals the product of current $I(s)$ and component impedance $Z(s)$, where $Z(s) = R$ for resistance, $Z(s) = Ls$ for inductance, and $Z(s) = 1/(Cs)$ for capacitance. Table 2.4 summarizes time and frequency domain relationships for resistance, inductance, and capacitance. In the frequency domain, the relationship of inductance (capacitance) current $I(s)$ and voltage $V(s)$ becomes algebraic, making the frequency domain solution simple.

Table 2.4: Impedance relationships.

Component	Time-domain	Frequency-domain	Impedance $Z(s)$
Resistance	$v(t) = Ri(t)$	$V(s) = RI(s)$	R
Inductance	$v(t) = L\frac{di(t)}{dt}$	$V(s) = LsI(s)$	Ls
Capacitance	$v(t) = \frac{1}{C}\int_0^t i(\tau)\,d\tau$	$V(s) = \frac{1}{Cs}I(s)$	$\frac{1}{Cs}$

In an electric circuit, there are often parallel and serial connections of different electric/electronic components including resistors, inductors, and capacitors. We will represent them using impedances $Z_1(s)$, $Z_2(s)$, ..., and $Z_n(s)$. Figure 2.15 shows n-component parallel and serial connection diagrams, along with their equivalent impedances.

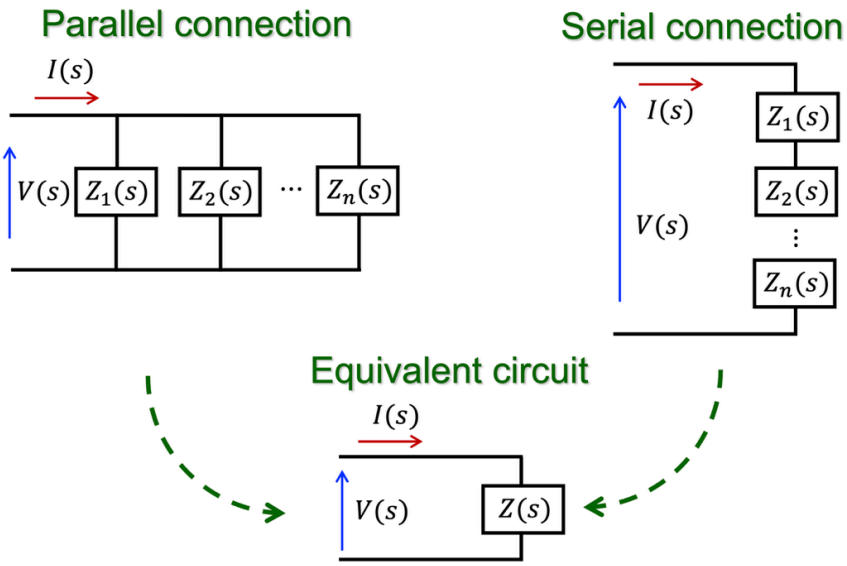


Figure 2.15: Parallel and serial impedance connections

Equation 2.3.7 shows the equivalent impedance for the parallel connection.

$$\frac{1}{Z\left(s\right)}=\frac{1}{Z_1\left(s\right)}+\frac{1}{Z_2\left(s\right)}+,\ \cdots,\ +\frac{1}{Z_n\left(s\right)}\ , \tag{2.3.7}$$

Equation 2.3.8 then shows the associated equivalent impedance for a serial connection.

$$Z\left(s\right)=Z_1\left(s\right)+Z_2\left(s\right)+,\ \cdots,\ +\ Z_n\left(s\right) \tag{2.3.8}$$

In turn, equation 2.3.9 shows the widely used formula for a two-component parallel connection.

$$Z\left(s\right)=\frac{Z_1\left(s\right)Z_2\left(s\right)}{Z_1\left(s\right)+Z_2\left(s\right)} \tag{2.3.9}$$

Note that equation 2.3.9 can be obtained using equation 2.3.7 by setting $n=2$. We will use this equation in the following example.

Example 2.8

Find the transfer function from $V_s(s)$ to $V_C(s)$ for the circuit presented in Figure 2.16.

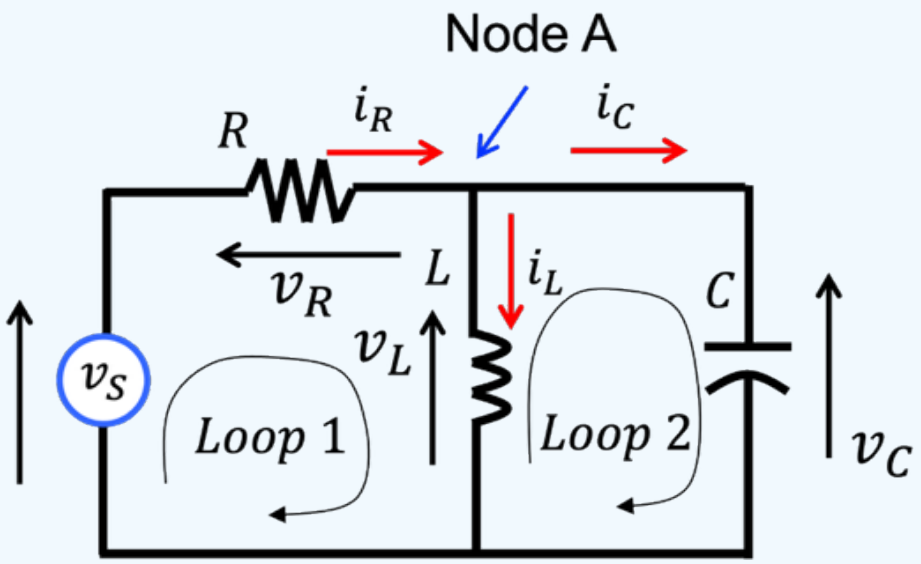


Figure 2.16: Circuit for Example 2.8

We will solve this problem using two approaches: time-domain and frequency domain approaches. Starting with the time-domain approach, let's use Kirchhoff's laws to find the required differential equations, conduct Laplace transforms of these differential equations, and solve for the transfer function.

Time-domain approach

First, we apply Kirchhoff's law to Loop 1 in Figure 2.16.

$$v_s(t) - v_R(t) - v_L(t) = 0 \;\rightarrow\; v_s(t) = Ri_R(t) + L\frac{di_L(t)}{dt} \tag{2.3.10}$$

Next, using Kirchhoff's voltage law for Loop 2 in Figure 2.16 leads to the following, assuming $v_C(0) = 0$. Note that for deriving a transfer function, all initial conditions are assumed to be zero.

$$v_L(t) - v_C(t) = 0 \;\rightarrow\; L\frac{di_L(t)}{dt} = \frac{1}{C}\int_o^t i_C(\tau)\,d\tau = v_C(t) \rightarrow\; L\frac{d^2i_L(t)}{dt^2} = \frac{1}{C}i_C(t) \tag{2.3.11}$$

Now, applying Kirchhoff's current law to Node A results in the following.

$$i_R(t) - i_C(t) - i_L(t) = 0 \;\rightarrow\; i_R(t) = i_L(t) + i_C(t) = i_L(t) + LC\frac{d^2i_L(t)}{dt^2} \tag{2.3.12}$$

To obtain $v_s(t)$ as a function of $i_L(t)$ only, substitute equation 2.3.12 into equation 2.3.10.

$$v_s(t) = Ri_L(t) + L\frac{di_L(t)}{dt} + LCR\frac{d^2i_L(t)}{dt^2} \tag{2.3.13}$$

Rewriting equation 2.3.11 leads to equation 2.3.14.

$$v_C(t) = L\frac{di_L(t)}{dt} \tag{2.3.14}$$

Applying the Laplace transform to equations 2.3.13 and 2.3.14 leads to the following equations, assuming zero initial conditions.

$$V_s(s) = RI_L(s) + LsI_L(s) + LCRs^2I_L(s) = (LCRs^2 + Ls + R)I_L(s) \quad V_C(s) = LsI_L(s) \tag{2.3.15}$$

We can find the transfer function by calculating the ratio of $V_C(s)$ and $V_s(s)$ based on equation 2.3.15 as follows.

$$G(s) = \frac{V_C(s)}{V_s(s)} = \frac{LsI_L(s)}{(LCRs^2 + Ls + R)I_L(s)} = \frac{Ls}{LCRs^2 + Ls + R} \tag{2.3.16}$$

Frequency Domain Approach

As discussed earlier, we can also calculate the transfer function in the frequency domain. Let us mark the impedances of the resistor, inductor, and capacitor on to the circuit; see the left diagram of Figure 2.17.

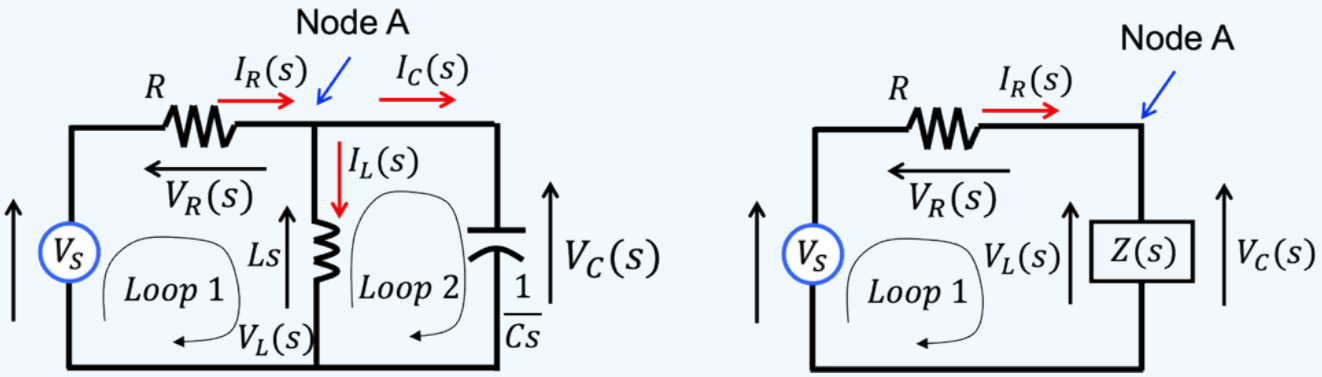


Figure 2.17: Frequency approach solution for Example 2.8

Now, we calculate the equivalent impedance of the parallel connection of inductor $Z_1(s) = Ls$ and capacitor $Z_2(s) = 1/(Cs)$ as follows.

$$Z(s) = \frac{Z_1(s)Z_2(s)}{Z_1(s) + Z_2(s)} = \frac{Ls/Cs}{Ls + 1/Cs} = \frac{Ls}{LCs^2 + 1} \tag{2.3.17}$$

Applying Kirchhoff's voltage law to Loop 1 (the right plot of Figure 2.17 and calculating the voltage drop $V_C(t)$ on $Z(s)$ results in the following.

$$V_s(s) = RI_R(s) + Z(s)I_R(s) = (R + Z(s))I_R(s), V_C(s) = Z(s)I_R(s) \tag{2.3.18}$$

Now we can calculate the transfer function using equation 2.3.18 by canceling $I_R(s)$,

$$G(s) = \frac{V_C(s)}{V_s(s)} = \frac{Z(s)}{R + Z(s)} = \frac{Ls/(LCs^2 + 1)}{R + Ls/(LCs^2 + 1)} = \frac{Ls}{LCRs^2 + Ls + R} \tag{2.3.19}$$

which is the same as that from the time domain approach. However, the frequency-domain approach is much simpler than the time-domain one, and it is also favorable.

Example 2.9

Find the transfer function from $V_i(s)$ to $V_o(s)$ for the circuit shown in Figure 2.18.

We will solve this problem using the time-domain approach in three steps. Note that the frequency domain approach essentially starts from step b) in this example using impedances of resistors and capacitor. Example 2.9.

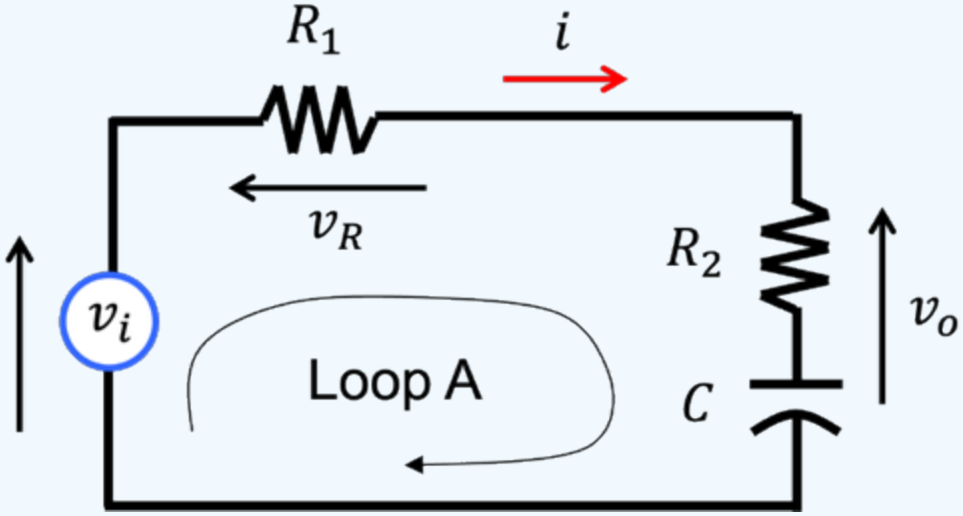


Figure 2.18: Circuit for Example 2.9

1. Using Kirchhoff's voltage law for Loop A in Figure 2.18 yields equation 2.3.20, assuming $v_C(0) = 0$.

$$v_i(t) = R_1 i(t) + R_2 i(t) + \frac{1}{C}\int_0^t i(\tau)\, d\tau \tag{2.3.20}$$

Using Ohm's law leads to equation 2.3.21, assuming $v_C(0) = 0$.

$$v_o(t) = R_2 i(t) + \frac{1}{C}\int_0^t i(\tau)\, d\tau \tag{2.3.21}$$

1. Conducting the Laplace transform on equations 2.3.20 and 2.3.21, assuming zero initial conditions, leads to the following.

$$\begin{aligned} V_i(s) &= R_1 I(s) + R_2 I(s) + \frac{1}{Cs} I(s) = \left(\frac{1}{Cs} + R_1 + R_2\right) I(s) \\ V_o(s) &= \left(\frac{1}{Cs} + R_2\right) I(s) \end{aligned} \tag{2.3.22}$$

1. Now, we can find the transfer function by calculating the ratio of $V_i(s)$ and $V_o(s)$ based on 2.3.22.

$$G(s) = \frac{V_o(s)}{V_i(s)} = \frac{\left(\frac{1}{Cs} + R_2\right) I(s)}{\left(\frac{1}{Cs} + R_1 + R_2\right) I(s)} = \frac{CR_2 s + 1}{C(R_1 + R_2)s + 1} \tag{2.3.23}$$

The transfer function can also be calculated in the frequency domain, starting with step b) above.

Operational Amplifier (OA) Circuit Assumptions

An important electronic device, called an operational amplifier (OA), is widely used in mechatronic systems such as low- and high-pass filters. An operational amplifier is represented by the triangle in Figure 2.19 with positive and negative input terminals. Note that the OA output voltage (relative to the ground) is equal to its input voltage $V_d(s)$ (see Figure 2.19) multiplied by the OA gain K, which can be expressed as below.

$$V_o(s) = KV_d(s) \tag{2.3.24}$$

There are two important assumptions made for modeling the OA circuit below.

First, the input resistance R_d of an operational amplifier is large enough to be close to infinity ($R_d \approx \infty$). Second, the operational amplifier gain K is also large enough to be close to infinity ($K \approx \infty$).

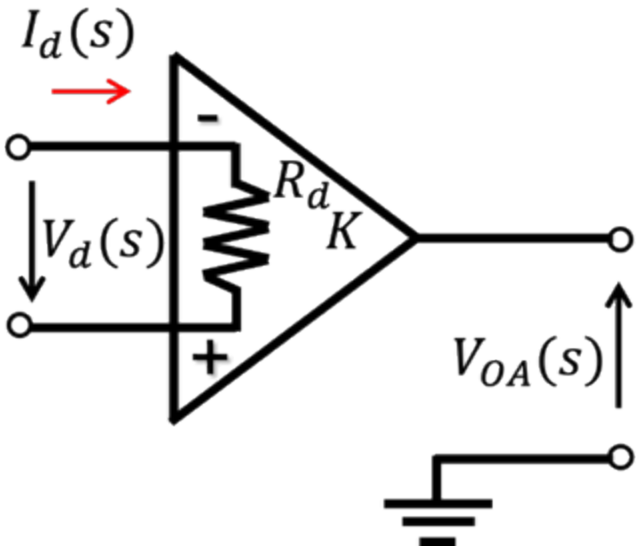


Figure 2.19: Operational amplifier

With these two OA circuit assumptions above, if we are to have a finite input voltage $V_d(s)$, then $I_d(s)$ has to be zero because $V_d(s) = RI_d(s)$. On the other hand, if we are to have a finite output voltage $V_{OA}(s)$, then $V_d(s)$ has to be zero because $V_{OA}(s) = KV_d(s)$. As a summary, in future operational amplifier circuit modeling, we can assume that $I_d(s) = 0$ and $V_d(s) = 0$, or in the time domain that $I_d(t) = 0$ and $V_d(t) = 0$.

Next, we will discuss the negative and positive feedback operational amplifier circuits. We'll look at those circuits where the negative feedback circuit has an equivalent impedance device connected between output and negative input, and the positive feedback circuit has an equivalent impedance device connected between output and positive input.

First, let's address the negative feedback circuit. Figure 2.20 shows a typical negative feedback OA circuit, where $V_i(s)$ and $V_o(s)$ are circuit input and output voltages, respectively. In addition, $I_i(s)$ and $I_o(s)$ are circuit input and output currents, respectively, and $Z_i(s)$ and $Z_o(s)$ are circuit input and output (feedback) impedances, respectively. Based on previous discussions, both the current $I_d(s)$ going to the OA and the voltage $V_d(s)$ between OA inputs can be assumed to be zero.

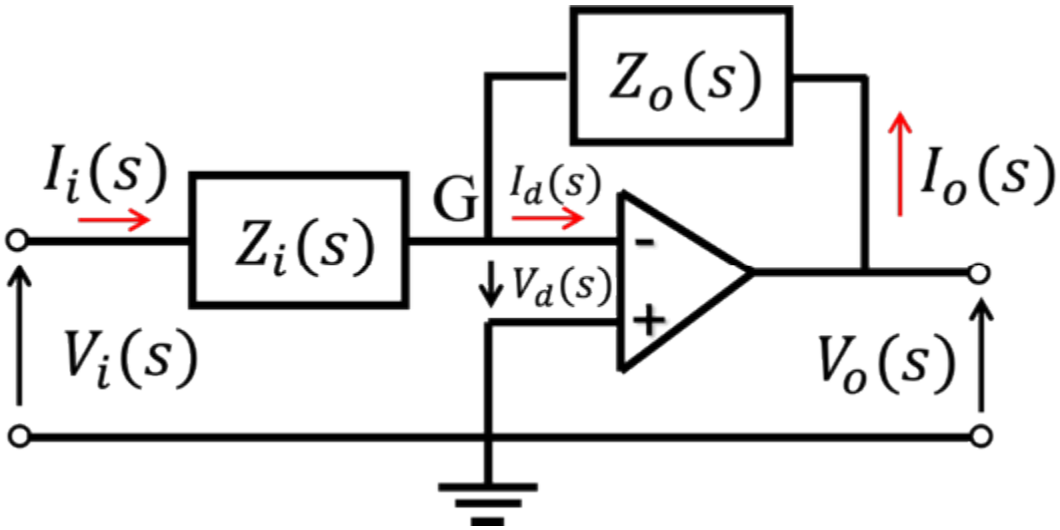


Figure 2.20: Negative feedback operational amplifier circuit.

For this negative feedback OA circuit, because $V_d(s)$ is assumed to be zero, the negative OA input location "G" is equivalent to the ground of the circuit. In this case, using Kirchhoff's voltage law, we have the following.

$$V_i(s) = Z_i(s)\, I_i(s)\,,\ V_o(s) = Z_o(s)\, I_o(s) \tag{2.3.25}$$

With the assumption that $I_d(s) = 0$, we have $I_i(s) = -I_o(s)$ based on Kirchhoff's current law. Therefore, the transfer function $G(s)$

from $V_i(s)$ to $V_o(s)$ can be calculated as below.

$$G(s) = \frac{V_o(s)}{V_i(s)} = \frac{Z_o(s) I_o(s)}{Z_i(s) I_i(s)} = -\frac{Z_o(s)}{Z_i(s)} \tag{2.3.26}$$

Example 2.10

Find the transfer function $G(s)$ from $V_i(s)$ to $V_o(s)$ for the circuit shown in Figure 2.21.

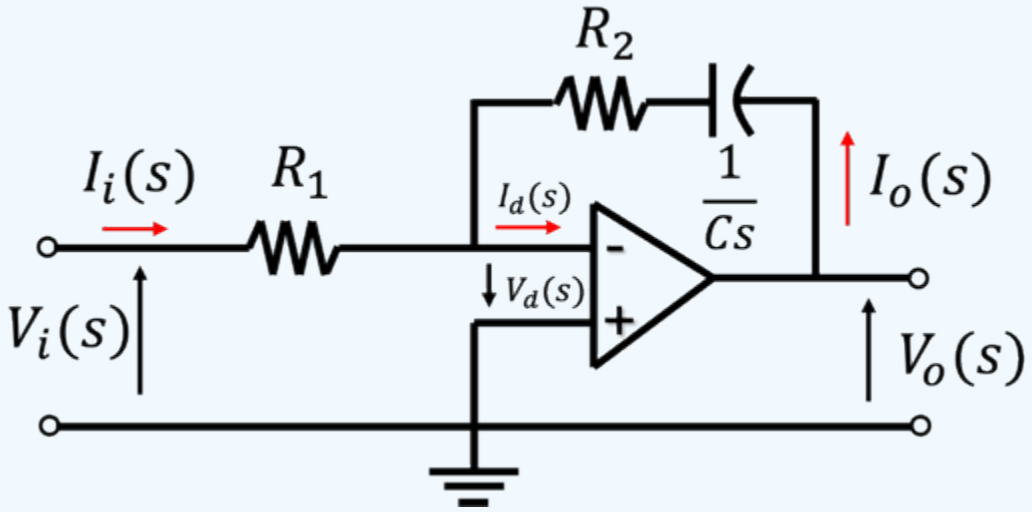


Figure 2.21: Negative feedback OA circuit for Example 2.10.

We will solve this problem in the frequency domain as below, using the serial connection formula in equation 2.3.8.

$$Z_i(s) = R_1, Z_o(s) = R_2 + \frac{1}{Cs} \tag{2.3.27}$$

Therefore,

$$G(s) = \frac{V_o(s)}{V_i(s)} = -\frac{Z_o(s)}{Z_i(s)} = -\frac{R_2 + \frac{1}{Cs}}{R_1} = -\frac{R_2 Cs + 1}{R_1 Cs} = -\left(\frac{R_2}{R_1} + \frac{1}{R_1 C} \cdot \frac{1}{s}\right) \tag{2.3.28}$$

This is also called a negative gain continuous-time PI (proportional and integral) controller. It is realized using an analog negative-feedback OA circuit, with its proportional gain, $-R_2/R_1$, and integral gain, $1/(R_1 C)$. Note that PI controllers will be addressed in Chapter 3.

Example 2.11

Find the transfer function $G(s)$ from $V_i(s)$ to $V_o(s)$ for the circuit shown in Figure 2.22.

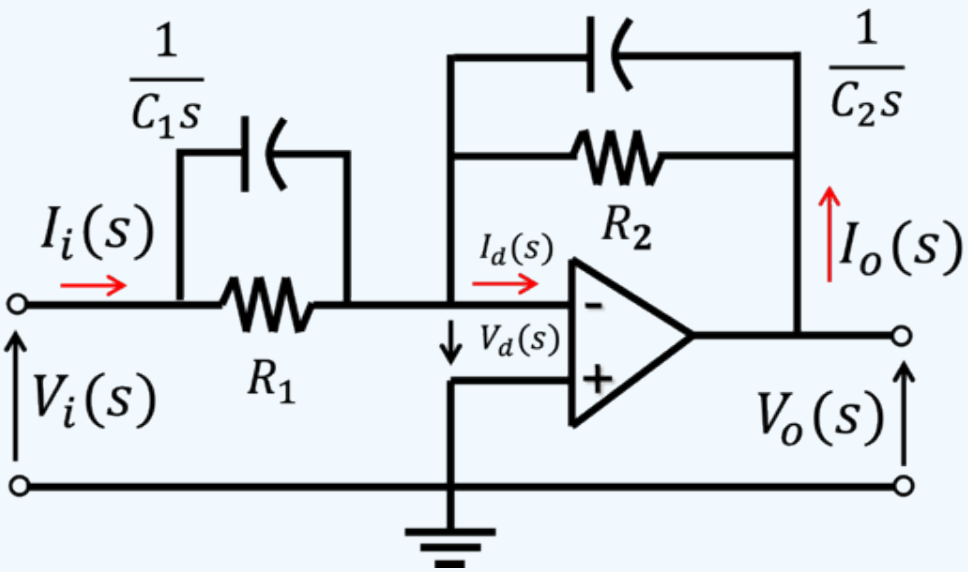


Figure 2.22: Negative feedback OA circuit for Example 2.11.

We will solve this problem in the frequency domain using the parallel connection formula in equation 2.3.7, as below.

$$Z_i(s) = \frac{\frac{R_1}{C_1 s}}{R_1 + \frac{1}{C_1 s}} = \frac{R_1}{R_1 C_1 s + 1}, Z_o(s) = \frac{R_2}{R_2 C_2 s + 1} \tag{2.3.29}$$

Therefore,

$$G(s) = \frac{V_o(s)}{V_i(s)} = -\frac{Z_o(s)}{Z_i(s)} = -\frac{\frac{R_2}{R_2 C_2 s + 1}}{\frac{R_1}{R_1 C_1 s + 1}} = -\frac{R_2}{R_1} \cdot \frac{R_1 C_1 s + 1}{R_2 C_2 s + 1} \tag{2.3.30}$$

This is a negative gain lead-leg (or lag-lead) filter circuit, which will be addressed in Chapter 4.

Now, we will discuss the positive input operational amplifier circuit. Figure 2.23 shows a typical positive input OA circuit, where $V_i(s)$ and $V_o(s)$ are circuit input and output voltages, respectively. In addition, $I_i(s)$ and $I_o(s)$ are circuit input and output currents, respectively, and $Z_i(s)$ and $Z_o(s)$ are circuit input and feedback impedances, respectively. Based on previous discussions, the current $I_d(s)$ going to the negative OA input and the voltage $V_d(s)$ between OA inputs can be assumed to be zero.

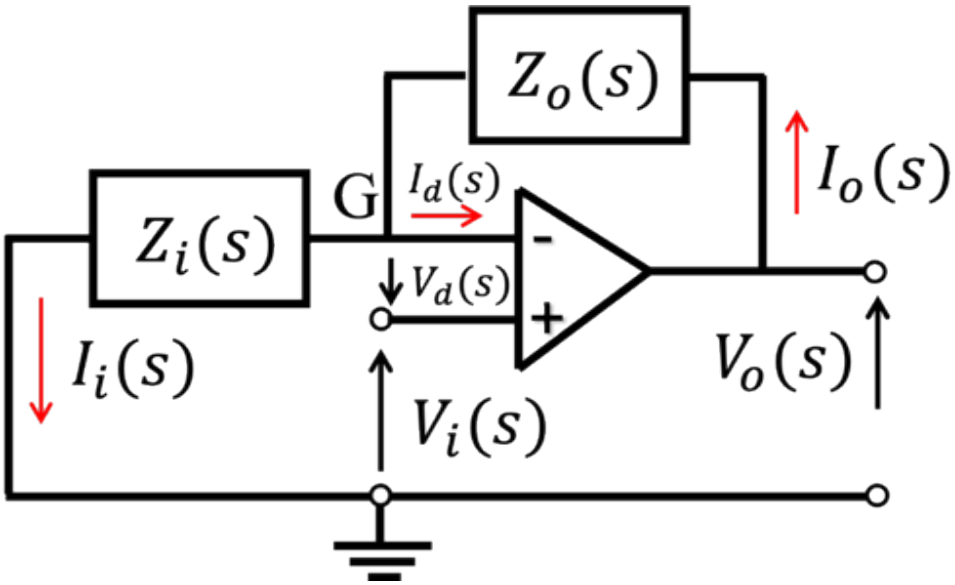


Figure 2.23: Positive input operational amplifier circuit

For this positive input OA circuit, because $V_d(s)$ is assumed to be zero, the voltage at the negative OA input location "G" is equal to the input voltage $V_i(s)$. Assuming that $I_i(s) = I_o(s)$, since $I_d(s) = 0$, using Kirchhoff's voltage law, we have

$$V_i(s) = Z_i(s) I_o(s), \; V_o(s) = (Z_i(s) + Z_o(s)) I_o(s) \tag{2.3.31}$$

Therefore, the transfer function $G(s)$ from $V_i(s)$ to $V_o(s)$ can be calculated as below.

$$G(s) = \frac{V_o(s)}{V_i(s)} = 1 + \frac{Z_o(s)}{Z_i(s)} \tag{2.3.32}$$

Example 2.12

Find the transfer function $G(s)$ from $V_i(s)$ to $V_o(s)$ for the circuit shown in Figure 2.24.

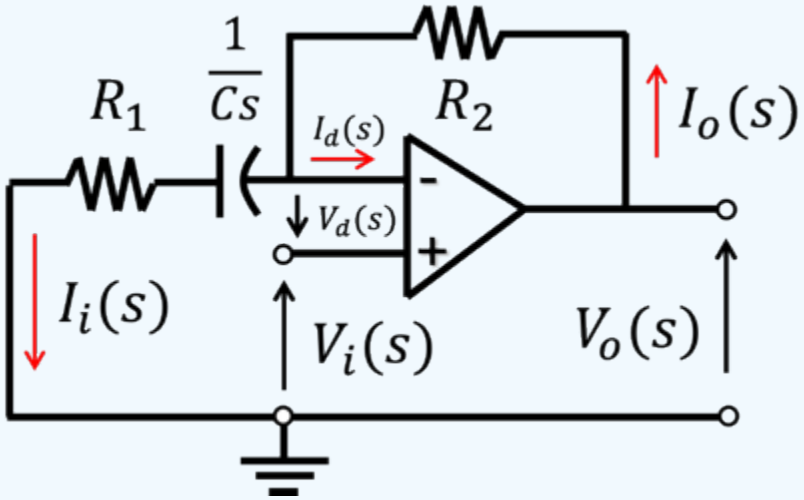


Figure 2.24: Positive input OA circuit for Example 2.12.

We will solve this problem in the frequency domain using the serial connection formula in equation 2.3.8, as below.

$$Z_i(s) = R_1 + \frac{1}{Cs}, Z_o(s) = R_2 \tag{2.3.33}$$

Therefore,

$$G(s) = \frac{V_o(s)}{V_i(s)} = 1 + \frac{Z_o(s)}{Z_i(s)} = 1 + \frac{R_2 Cs}{R_1 Cs + 1} = \frac{(R_1 + R_2)Cs + 1}{R_1 Cs + 1} \tag{2.3.34}$$

This is a realization of a lead-lag filter because both resistance values need to be positive; see Chapter 4 for more information.

Example 2.13

Find the transfer function $G(s)$ from $V_i(s)$ to $V_o(s)$ for the circuit shown in Figure 2.25.

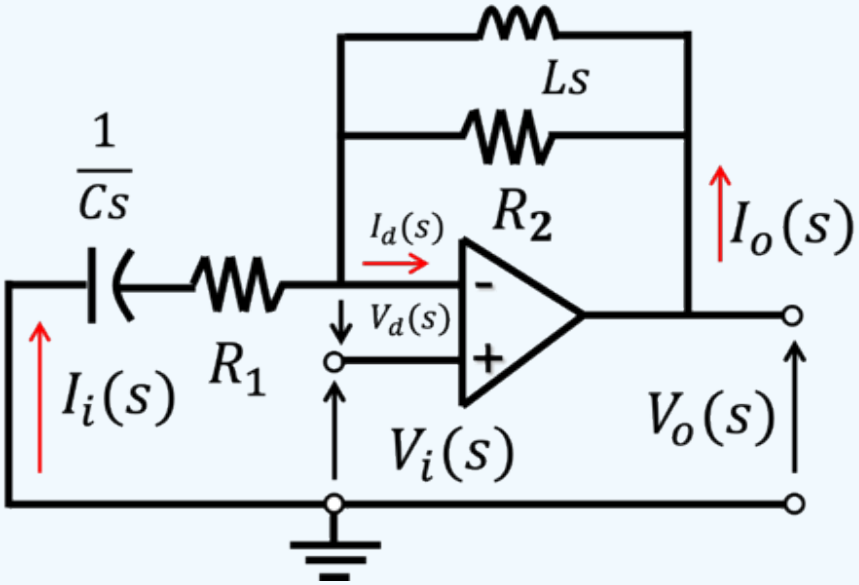


Figure 2.25: Positive input OA circuit for Example 2.13.

We will again solve this problem in the frequency domain using the parallel and serial connection formulae in equations 2.3.7 and 2.3.8, as below.

$$Z_i(s) = R_1 + \frac{1}{Cs} = \frac{R_1 Cs + 1}{Cs}, \quad Z_o(s) = \frac{R_2 Ls}{R_2 + Ls} \tag{2.3.35}$$

Therefore, for the positive feedback circuit above,

$$G(s) = \frac{V_o(s)}{V_i(s)} = 1 + \frac{Z_o(s)}{Z_i(s)} = 1 + \frac{\frac{R_2 Ls}{R_2 + Ls}}{\frac{R_1 Cs + 1}{Cs}} = 1 + \frac{R_2 LCs^2}{(Ls + R_2)(R_1 Cs + 1)} \tag{2.3.36}$$

Equation 2.3.36 can be reorganized into the following.

$$G(s) = \frac{V_o(s)}{V_i(s)} = \frac{(R_1 + R_2)LCs^2 + (L + R_1 R_2 C)s + R_2}{(R_2 + Ls)(R_1 Cs + 1)}, \tag{2.3.37}$$

Equation 2.3.37 leads to a second order transfer function. Note that the order of a transfer function is determined by its denominator's polynomial order. For a causal system, the numerator's polynomial order is smaller or equal to its denominational polynomial order.

Figure 2.26 shows a special positive feedback OA circuit, where $Z_i(s) = \infty$ and $Z_o(s) = 1$. This circuit is widely used to isolate the output from input because it can be easily seen that the input impedance of this circuit is equal to the OA input impedance that is close to infinity.

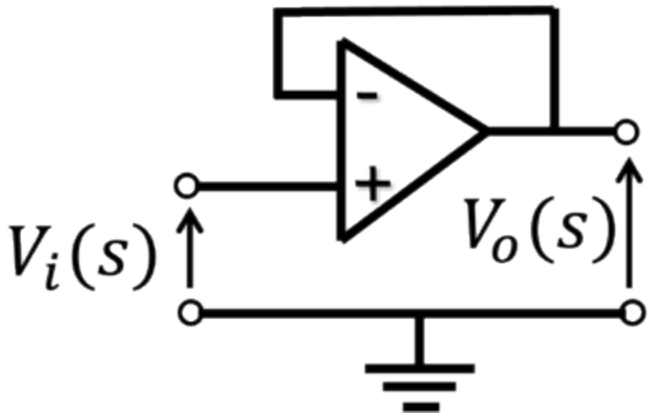


Figure 2.26: OA input isolator (follower) with close to infinity input impedance.

The transfer function of this circuit can be found using the transfer function 2.3.32 of a positive feedback OA circuit by setting $Z_i(s) = \infty$ and $Z_o(s) = 1$ as follows.

$$G(s) = \frac{V_o(s)}{V_i(s)} = 1 + \frac{Z_o(s)}{Z_i(s)} = 1 + \frac{1}{\infty} = 1 \tag{2.3.38}$$

Hence, this circuit is also called a unit gain following circuit used to isolate two connected circuits.

Section Summary

This section considers the modeling process for electrical/electronic systems consisting of resistors, capacitors, inductors, and operational amplifiers. Both time domain and frequency domain modeling approaches are addressed, where the concept of component impedance is introduced so that the frequency domain modeling becomes the derivation of algebraic equations. The infinity-gain and input impedance characteristics of operational amplifiers are discussed to make modeling easier, which is important for the physical realization of analog filters.

Section Keywords

Electric/electronic circuit modeling: This process creates a virtual representation of a circuit for analyzing and simulating its behavior before building a physical prototype.

Impedance: It is the effective resistance of an electric circuit or component to alternating current due to the combined effects of ohmic resistance and reactance, such as inductance and capacitance.

Operational amplifier (OA): A high-gain voltage amplifier with a differential input and a single output, it is designed to perform mathematical operations in analog circuits.

Operational amplifier circuit modeling: This process creates a virtual representation of a circuit containing operational amplifier(s) for analyzing and simulating its behavior before building a physical prototype.

Practice Problems

1. Find the transfer function $G(s)$ from $V_i(s)$ to $V_o(s)$ for the circuit shown below.
2. Find the transfer function $G(s)$ from $V_i(s)$ to $V_o(s)$ for the circuit shown below.
3. Find the transfer function $G(s)$ from $V_i(s)$ to $V_o(s)$ for the negative-feedback OA circuit shown below.
4. Find the transfer function $G(s)$ from $V_i(s)$ to $V_o(s)$ for the positive-input OA circuit shown below.

This page titled 2.3: Electrical and Electronic System Modeling is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

2.4: Mechatronic System Modeling

Learning Objectives

After completing this section, you will be able to:

1. Define a mechatronic subsystem such as the DC motor.
2. Draw a system block diagram based on its Laplace transforms and derive a feedback system transfer function based on the developed block diagram.
3. Model a DC-motor mechatronic subsystem.

This section describes how to model mechatronic systems. A mechatronic system consists of interconnected mechanical and electric/electronic systems (or subsystems). For example, a DC motor—from controlled voltage input to resulting angular velocity and displacement—is a mechatronic subsystem. In this section we will discuss modeling a DC motor (actuator) and accelerometer (sensor); both are widely used in mechatronic systems.

First, we will review transfer function block diagrams because they are very useful for deriving the transfer function of a complex mechatronic system.

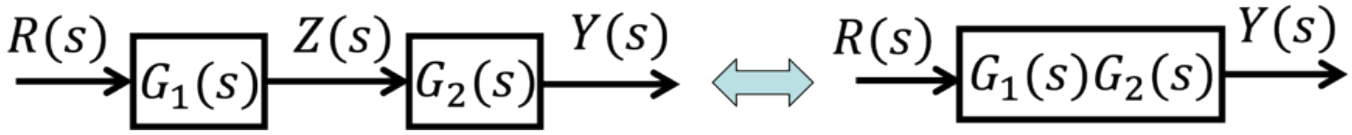


Figure 2.27: Serial connected block diagram.

Consider the serial connection of two transfer functions, $G_1(s)$ and $G_2(s)$, as illustrated by the left plot of Figure 2.27.

$$G_1(s) = \frac{Z(s)}{R(s)}, G_2(s) = \frac{Y(s)}{Z(s)} \tag{2.4.1}$$

For the serial connection, the combined transfer function is shown below.

$$G(s) = G_2(s)\,G_1(s) = \frac{Y(s)}{Z(s)}\frac{Z(s)}{R(s)} = \frac{Y(s)}{R(s)} \tag{2.4.2}$$

Now consider the parallel connection of $G_1(s) = Z_1(s)/E(s)$ and $G_2(s) = Z_2(s)/E(s)$, the two transfer functions shown in the left plot of Figure 2.28. Here the summation in the diagram represents the signal summation of $Z_1(s)$ and $Z_2(s)$ such that $Y(s) = Z_1(s) + Z_2(s)$.

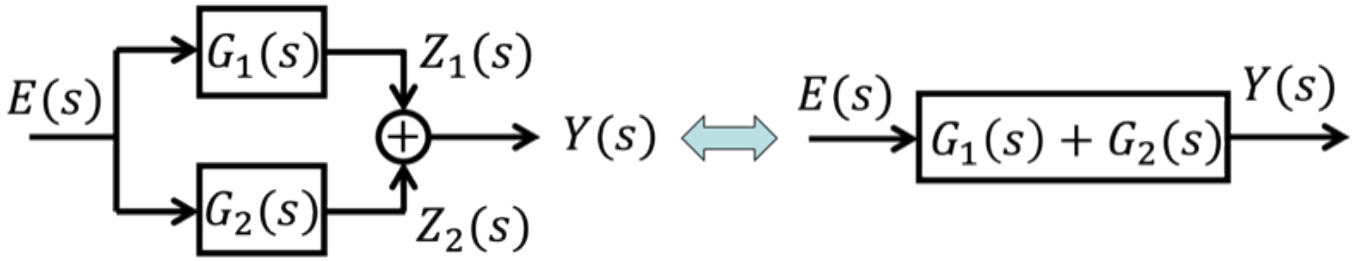


Figure 2.28: Parallel connected block diagram.

The equivalent transfer function is shown in the right plot of Figure 2.28.

$$G(s) = \frac{Y(s)}{E(s)} = \frac{Z_1(s) + Z_2(s)}{E(s)} = G_1(s) + G_2(s) \tag{2.4.3}$$

Now we are ready to introduce an important block diagram—the negative feedback block diagram (see Figure 2.29)—and the associated formula used to calculate the equivalent transfer function from $R(s)$ to $Y(s)$.

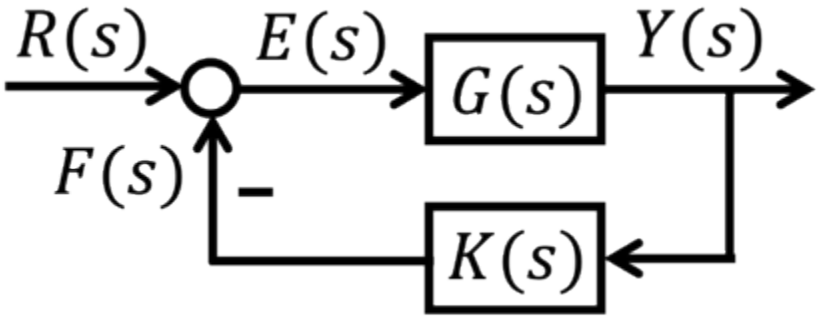


Figure 2.29: Negative feedback block diagram.

In Figure 2.29, the error signal $E(s) = R(s) - K(s)Y(s)$, where the negative sign is near the summation circle, indicates the subtraction of $F(s) = K(s)Y(s)$ from $R(s)$. From the above diagram, because $Y(s) = G(s)E(s)$, we have

$$E(s) = R(s) - K(s)Y(s) = R(s) - K(s)G(s)E(s) \tag{2.4.4}$$

Reorganizing equation 2.4.4 yields the following.

$$\frac{Y(s)}{R(s)} = \frac{G(s)}{1 + K(s)G(s)} \tag{2.4.5}$$

Now let's define forward gain $F_g(s)$ from $E(s)$ to $Y(s)$ and loop gain $L_g(s)$ from $E(s)$ to $F(s)$. For the block diagram in Figure 2.29, the forward gain from $E(s)$ to $Y(s)$ and loop gain from $Y(s)$ back to $Y(s)$ can be calculated as follows.

$$F_g(s) = G(s), L_g(s) = -K(s)G(s) \tag{2.4.6}$$

Then, we can generalize the transfer function from $R(s)$ to $Y(s)$ as following.

$$\frac{Y(s)}{R(s)} = \frac{F_g(s)}{1 - L_g(s)} = \frac{G(s)}{1 + K(s)G(s)} \tag{2.4.7}$$

Example 2.14

Find the transfer function $G(s)$ from $R(s)$ to $Y(s)$ for the block diagrams shown in Figure 2.30.

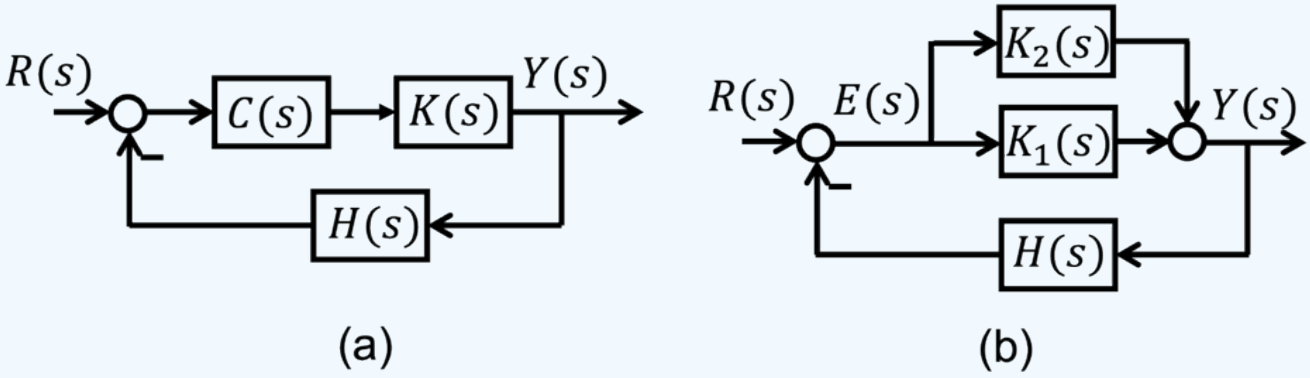


Figure 2.30: Block diagram for Example 2.14.

First, consider block diagram (a) in Figure 2.30, where $F_g(s)$ and $L_g(s)$ can be calculated as below.

$$F_g(s) = K(s)C(s), L_g(s) = -H(s)K(s)C(s) \tag{2.4.8}$$

Now using the formula in equation 2.4.7 yields the following.

$$G(s) = \frac{Y(s)}{R(s)} = \frac{F_g(s)}{1 - L_g(s)} = \frac{K(s)C(s)}{1 + H(s)K(s)C(s)} \tag{2.4.9}$$

For the circuit (b) in Figure 2.30, we have the following.

$$F_g(s) = K_1(s) + K_2(s), L_g(s) = -H(s)(K_1(s) + K_2(s)) \tag{2.4.10}$$

Using equation 2.4.7 results in the following.

$$G(s) = \frac{Y(s)}{R(s)} = \frac{F_g(s)}{1 - L_g(s)} = \frac{K_1(s) + K_2(s)}{1 + H(s)(K_1(s) + K_2(s))} \tag{2.4.11}$$

We can see that using equation 2.4.7 makes it easy to calculate the negative or positive feedback transfer function.

Now we are ready to address DC motor modeling. A DC motor is often coupled with an encoder for speed measurement (see Chapter 6). Figure 2.31 shows a DC motor. At this point, however, we focus on how to model a DC motor; its detailed working principles will be addressed in Chapter 5.



Figure 2.31: DC motor with encoder.

A DC motor is an electric machine which converts electrical energy into mechanical energy. Most small DC motors do not have stator windings. Instead, permanent magnets are placed at the stator location to form a magnetic field. As shown in Figure 2.32, applying the armature voltage $e_a(t)$ to the rotor windings through the brushed communicator on the rotor axle results in the armature current $i_a(t)$. This current, along with the magnetic field generated by the stator permanent magnets, produces the motor torque $\tau(t)$ on the motor axle. (See Chapter 5 for a more detailed discussion of brushed communicators.)

Figure 2.32 presents the armature circuit and mechanical load block diagram for a DC motor. Here, R_a and L_a are the rotor winding resistance and inductance, respectively. In addition, $e_m(t)$ is the back EMF (electromotive force) voltage caused by rotor movement inside the magnetic field; $\theta(t)$ is the angular displacement of the motor axle; J and b_m are the rotor axle combined moment of inertia and rotational damping (friction) coefficient, respectively; and $\tau_l(t)$ is the external load torque applied to the rotor axle. Therefore, a DC motor is a typical mechatronic subsystem. In this example, its input is armature voltage $e_a(t)$ and its output is motor axle angular speed $\dot{\theta}(t)$ and displacement $\theta(t)$, where the electrical and mechanical subsystems are coupled by the motor back EMF voltage $e_m(t)$ and motor output speed $\dot{\theta}(t)$. Note that subscript "a" indicates the armature side of the circuit.

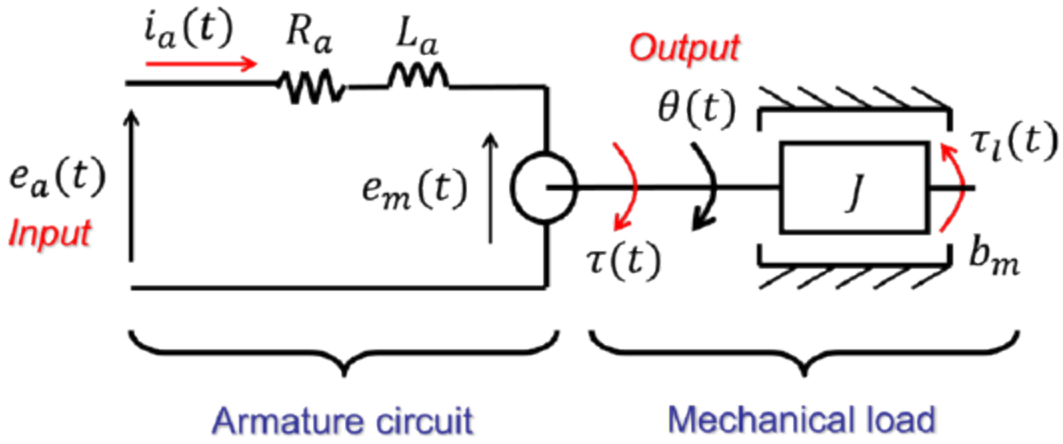


Figure 2.32: DC motor circuit and mechanical diagram.

Now we are ready to derive the DC motor model from armature voltage $e_a(t)$ to motor axle angular speed $\dot{\theta}(t)$ and displacement $\theta(t)$. Applying Kirchhoff's voltage law to the armature circuit in Figure 2.32 leads to the following equation.

$$e_a(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + e_m(t) \tag{2.4.12}$$

The corresponding Laplace transform of equation 2.4.12 is

$$E_a(s) = R_a I_a(s) + L_a s I_a(s) + E_m(s) \tag{2.4.13}$$

Based on the Lorentz force law (see Chapter 5), the motor output torque $\tau(t)$ is linearly proportional to armature current $i_a(t)$. That is,

$$\tau(t) = K_\tau i_a(t) \tag{2.4.14}$$

where K_τ is the DC motor torque coefficient. Again, using the Lorentz force law, the motor back EMF voltage $e_m(t)$ is also linearly proportional to the motor rotor speed $\dot{\theta}(t) = \omega(t)$. That is,

$$e_m(t) = K_m \dot{\theta}(t) = K_m \omega(t) \tag{2.4.15}$$

where K_m is the DC motor back EMF coefficient. Note that after the DC motor is designed and fabricated, coefficients K_τ and K_m are fixed. The associated Laplace transforms of equations 2.4.14 and 2.4.15 are shown below.

$$\begin{aligned} T(s) &= K_\tau I_a(s) \\ E_m(s) &= K_m s\Theta(s) = K_m \Omega(s) \end{aligned} \tag{2.4.16}$$

Now, let's consider the mechanical block diagram in Figure 2.32. Applying Newton's law to the free body diagram with moment of inertia J results in the following.

$$J\ddot{\theta}(t) = \tau(t) - b_m \dot{\theta}(t) - \tau_l(t) \tag{2.4.17}$$

The associated Laplace transform is shown below.

$$Js^2\Theta(s) = T(s) - b_m s\Theta(s) - T_l(s) \tag{2.4.18}$$

Reorganizing equations 2.4.13, 2.4.16, and 2.4.18 leads to the following set of equations describing the DC-motor dynamics.

$$\begin{aligned} I_a(s) &= \frac{1}{L_a s + R_a}\left(E_a(s) + E_m(s)\right) \\ T(s) &= K_\tau I_a(s), E_m(s) = K_m \Omega(s) \\ \Omega(s) &= \frac{1}{Js + b_m}\left(T(s) - T_l(s)\right) \\ \Omega(s) &= s\Theta(s) \end{aligned} \tag{2.4.19}$$

To derive the transfer function from $E_a(s)$ to $\Omega(s)$ and $\Theta(s)$, we will use the block diagram approach introduced at the beginning of this section. Based on equation 2.4.19, we can draw the system block diagram seen in Figure 2.33, where the colors in the block diagram are associated with the colors in equation 2.4.19.

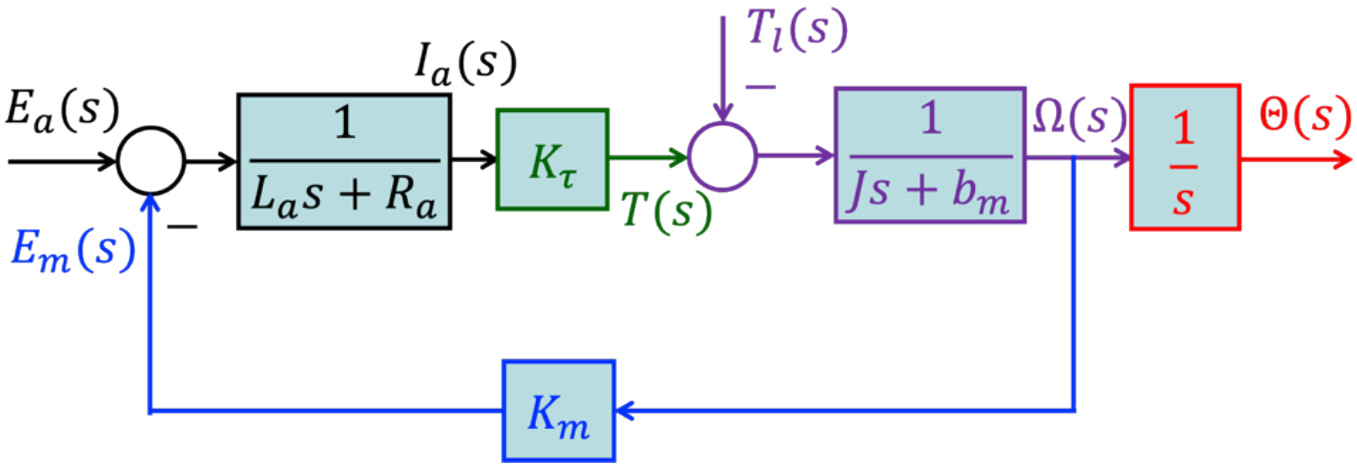


Figure 2.33: DC-motor system block diagram.

Now, we can derive the transfer function from $E_a(s)$ to $\Omega(s)$ by setting $T_l(s) = 0$. From Figure 2.33, for the system transfer function from $E_a(s)$ to $\Omega(s)$, we can calculate the forward gain $F_g(s)$ as

$$F_g(s) = \frac{K_\tau}{L_a s + R_a}\frac{1}{Js + b_m} = \frac{K_\tau}{(L_a s + R_a)(Js + b_m)} \tag{2.4.20}$$

and the associated loop gain $L_g(s)$ as

$$L_g(s) = \frac{-K_\tau K_m}{(L_a s + R_a)(Js + b_m)} \tag{2.4.21}$$

Then, using equation 2.4.7, we can obtain the transfer function from $E_a(s)$ to $\Omega(s)$.

$$G_1(s) = \frac{\Omega(s)}{E_a(s)} = \frac{F_g(s)}{1 - L_g(s)} = \frac{K_\tau}{(L_a s + R_a)(Js + b_m) + K_\tau K_m} \tag{2.4.22}$$

We can find transfer function $G_2(s)$ from $E_a(s)$ to $\Theta(s)$ by using the relationship $\Omega(s) = s\Theta(s)$.

$$G_2(s) = \frac{\Theta(s)}{E_a(s)} = \frac{K_\tau}{s\left[(L_a s + R_a)(Js + b_m) + K_\tau K_m\right]} \tag{2.4.23}$$

We derive the transfer function from load torque $T_l(s)$ to $\Omega(s)$ using the same approach by setting $E_a(s) = 0$. From Figure 2.33, for the system from $-T_l(s)$ to $\Omega(s)$, we can calculate the forward gain $F_g(s)$ as

$$F_g(s) = \frac{1}{Js + b_m}, \tag{2.4.24}$$

and the loop gain $L_g(s)$ as

$$L_g(s) = \frac{-K_\tau K_m}{(L_a s + R_a)(Js + b_m)} \tag{2.4.25}$$

Next, using equation 2.4.7 we can obtain the transfer function from $T_l(s)$ to $\Omega(s)$ as below.

$$G_3(s) = \frac{\Omega(s)}{T_l(s)} = -\frac{F_g(s)}{1 - L_g(s)} = \frac{-(L_a s + R_a)}{(L_a s + R_a)(Js + b_m) + K_\tau K_m} \tag{2.4.26}$$

Note that the negative sign in 2.4.26 is due to conversion from $-T_l(s)$ to $T_l(s)$. We can now easily find the transfer function $G_4(s)$ from $T_l(s)$ to $\Theta(s)$ using the relationship $\Omega(s) = s\Theta(s)$ as below.

$$G_4(s) = \frac{\Theta(s)}{T_l(s)} = \frac{-(L_a s + R_a)}{s\left[(L_a s + R_a)(Js + b_m) + K_\tau K_m\right]} \tag{2.4.27}$$

The DC-motor mechatronic system model is linear. Hence, the superposition rule (see Section 2.5) holds. In this case,

$$\Omega(s) = G_1(s) E_a(s) + G_3(s) T_l(s) = \frac{K_\tau E_a(s) - (L_a s + R_a) T_l(s)}{(L_a s + R_a)(Js + b_m) + K_\tau K_m} \tag{2.4.28}$$

and similarly,

$$\Theta(s) = G_2(s) E_a(s) + G_4(s) T_l(s) = \frac{K_\tau E_a(s) - (L_a s + R_a) T_l(s)}{s\left[(L_a s + R_a)(Js + b_m) + K_\tau K_m\right]} \tag{2.4.29}$$

Practically, for most DC motors, L_a is much smaller than R_a, that is, $L_a \ll R_a$. In this case, we set $L_a = 0$, which results in the following approximated transfer function for $G_1(s)$

$$G_1(s) = \frac{K_\tau}{JR_a s + R_a b_m + K_\tau K_m} = \frac{K}{Ts + 1} \tag{2.4.30}$$

where

$$K = \frac{K_\tau}{R_a b_m + K_\tau K_m}, T = \frac{JR_a}{R_a b_m + K_\tau K_m} \tag{2.4.31}$$

Note that the resulting system in equation 2.4.30 is a first-order system model because its denominator's polynomial has an order of one and the original transfer function in 2.4.22 is second order. Therefore, setting $L_a = 0$ reduces the system model order from two to one. In addition, the transfer function in equation 2.4.30 is in the standard form for the first-order system to be discussed in Chapter 3.

Example 2.15

Find the transfer functions $G_\Theta(s)$ from $E_a(s)$ to $\Theta(s)$ and $G_X(s)$ from $E_a(s)$ to $X(s)$ for the DC-motor system shown in Figure 2.34.

First, consider the circuit and mechanical system diagram in Figure 2.34. For the electrical circuit on the left, we can have the same circuit equation in the frequency domain as below (see the derivation earlier for equation 2.4.19).

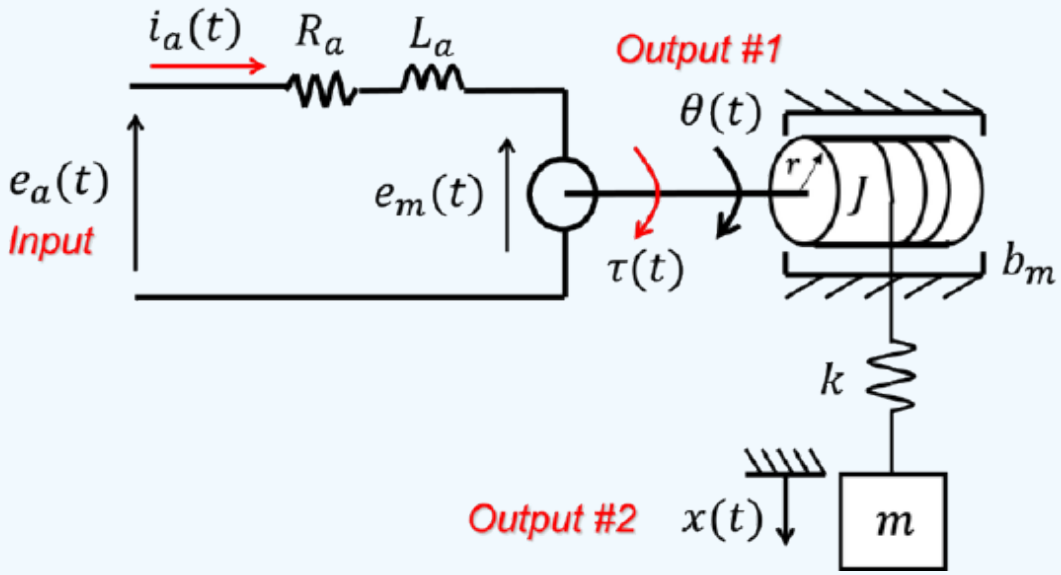


Figure 2.34: DC motor system for Example 2.15.

$$I_a(s) = \frac{1}{L_a s + R_a}\left(E_a(s) + E_m(s)\right) \tag{2.4.32}$$

The relationships coupling the electric circuit and mechanical system are as follows.

$$\begin{aligned} T(s) &= K_\tau I_a(s) \\ E_m(s) &= K_m \Omega(s) = K_m s \Theta(s) \end{aligned} \tag{2.4.33}$$

For the mechanical system modeling, we can have the free body diagrams seen in Figure 2.35

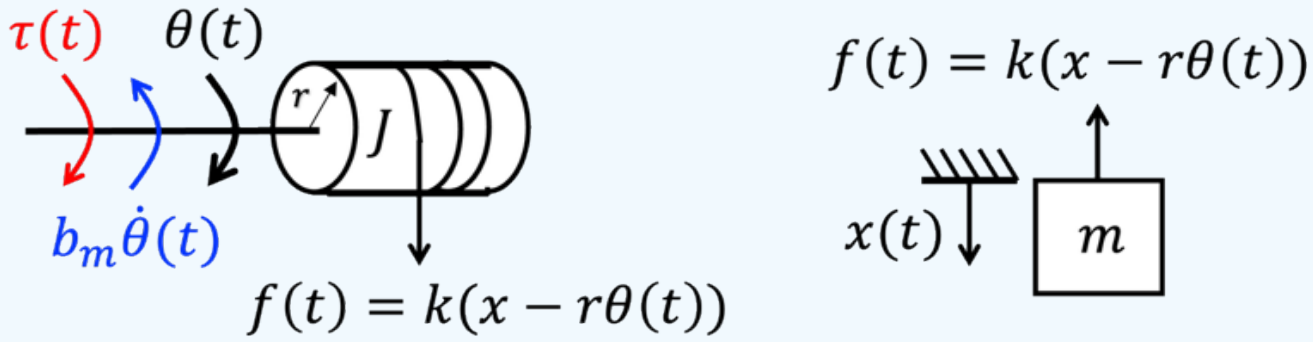


Figure 2.35: Free body diagrams for Example 2.15.

Applying Newton's law to these two free body diagrams leads to the following differential equations.

$$\begin{aligned} J\ddot{\theta}(t) &= \tau(t) - b_m\dot{\theta}(t) + kr\left(x(t) - r\theta(t)\right) \\ m\ddot{x}(t) &= -k\left(x(t) - r\theta(t)\right) \end{aligned} \tag{2.4.34}$$

In turn, the associated Laplace transforms are shown below.

$$\Theta(s) = \frac{T(s) + krX(s)}{Js^2 + b_m s + kr^2}, X(s) = \frac{kr\Theta(s)}{ms^2 + k} \tag{2.4.35}$$

Using equations 2.4.32, 2.4.33, and 2.4.35, we can obtain the block diagram in Figure 2.36.

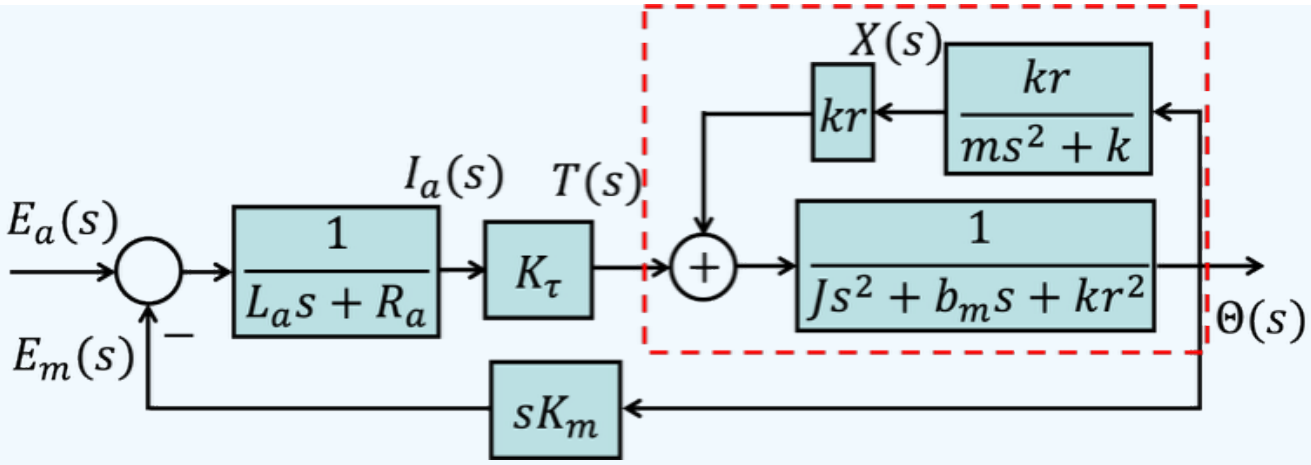


Figure 2.36: System block diagram for Example 2.15.

The transfer function from $T(s)$ to $\Theta(s)$ inside the red dotted box (Figure 2.36) can be represented by the following transfer function using equation 2.4.7. In using this negative feedback transfer function formula, we let $F_g(s) = 1/(Js^2 + b_m s + kr^2)$ and $L_g(s) = k^2r^2/[(Js^2 + b_m s + kr^2)\,(ms^2 + k)]$.

$$\frac{\Theta\,(s)}{T(s)} = \frac{ms^2 + k}{s\,[Jms^3 + mb_m s^2 + (mr^2 + J)ks + kb_m]} \tag{2.4.36}$$

For the large feedback loop, let

$$F_g(s) = \frac{ms^2 + k}{s\,[Jms^3 + mb_m s^2 + (mr^2 + J)ks + kb_m]}\,\frac{K_\tau}{L_a s + R_a} \tag{2.4.37}$$

and

$$L_g\,(s) = -\frac{ms^2 + k}{s\,[Jms^3 + mb_m s^2 + (mr^2 + J)ks + kb_m]}\,\frac{sK_m K_\tau}{L_a s + R_a} \tag{2.4.38}$$

Using equation 2.4.7 yields the following transfer function from $E_a(s)$ to $\Theta\,(s)$.

$$G_\Theta\,(s) = \frac{\Theta\,(s)}{E_a\,(s)} = \frac{F_g\,(s)}{1 - L_g\,(s)} = \frac{K_\tau\,(ms^2 + k)}{\{s\,[Jms^3 + mb_m s^2 + (mr^2 + J)ks + kb_m]\}\,(L_a s + R_a) + sK_m K_\tau\,(ms^2 + k)} \tag{2.4.39}$$

Note that the above equation is a fifth-order transfer function. It consists of the first-order DC-motor dynamics, second-order dynamics of moment of inertia J, and second-order dynamics of mass m. Given equation 2.4.35, the transfer function $G_X\,(s)$ from $E_a(s)$ to $X\,(s)$ is equal to $krG_\Theta\,(s)\,/\,(ms^2 + k)$, as shown below.

$$G_X\,(s) = \frac{K_\tau kr}{s\,\{[Jms^3 + mb_m s^2 + (mr^2 + J)ks + kb_m]\,(L_a s + R_a) + K_m K_\tau\,(ms^2 + k)\}} \tag{2.4.40}$$

We have discussed how to model a DC motor—a mechatronic actuator—based on the block diagram manipulation formula of equation 2.4.7. Next, we will discuss how to model a mechatronic sensor system, an accelerometer. In general, an accelerometer is a sensor that converts the mechanical body acceleration to an electronic signal. Its operational principle is based on both D'Alembert's force principle and Newton's laws of motion. The detailed working principle can be found in [17]. In this section, we will derive a sensor transfer function from $s^2X_a(s)$ ($\ddot{x}_a(t)$ in the time domain) to the filtered piezoelectric sensor output voltage $V_p(s)$ based on the sensor system illustrated in the left plot of Figure 2.37.

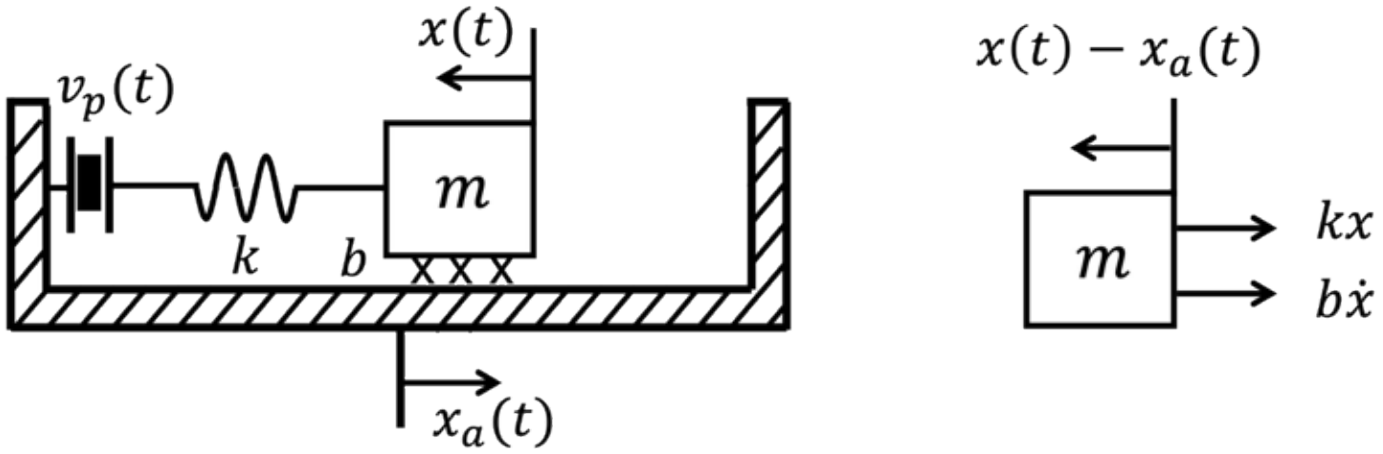


Figure 2.37: An accelerometer structural diagram.

In Figure 2.37, m is a moving mass inside the sensor, k is the stiffness of the spring connecting the moving mass to the sensor body, and b is the friction coefficient between the mass and sensor body. In addition, $v_p(t)$ is the output voltage of the piezoelectric sensor that is proportional to the force applied within a certain frequency range, that is, with a known frequency range, $v_p(t) \approx kx(t)$, where $kx(t)$ is the spring force applied. Note that $x(t)$ is the mass displacement relative to the sensor body and $x_a(t)$ is the displacement of the sensor body. Applying Newton's law to the free body diagram shown in the right plot of Figure 2.37, we have the following. (Newton's law will be applied to the inertia coordinate for mass m.)

$$m\left(\ddot{x}(t) - \ddot{x}_a(t)\right) = -b\dot{x}(t) - kx(t) \tag{2.4.41}$$

Reorganizing equation 2.4.41 results in the following.

$$m\ddot{x}(t) + b\dot{x}(t) + kx(t) = m\ddot{x}_a(t) \tag{2.4.42}$$

We can then have the following transfer function,

$$X(s) = \frac{m}{ms^2 + bs + k}\left\{s^2 X_a(s)\right\} = G_a(s)\left\{s^2 X_a(s)\right\} \tag{2.4.43}$$

In general, a piezoelectric sensor has a transfer function with its frequency response shown in Figure 2.38 in a Bode diagram format [1], where the horizontal axis is the frequency of applied force $kX(s)$ in logarithm and the vertical axis is the sensor output voltage magnitude in 20 times the logarithm magnitude (that is, $20\log_{10}|V_p(\omega)|$). Reference [18] provides Bode plot details. Note that the flat region between ω_{min} and ω_{max} is the linear frequency range of the sensor output. That is, within this frequency range of absolute acceleration $s^2 X_a(s)$, the piezoelectric sensor will output a voltage signal proportional to the applied force $kx(t)$.

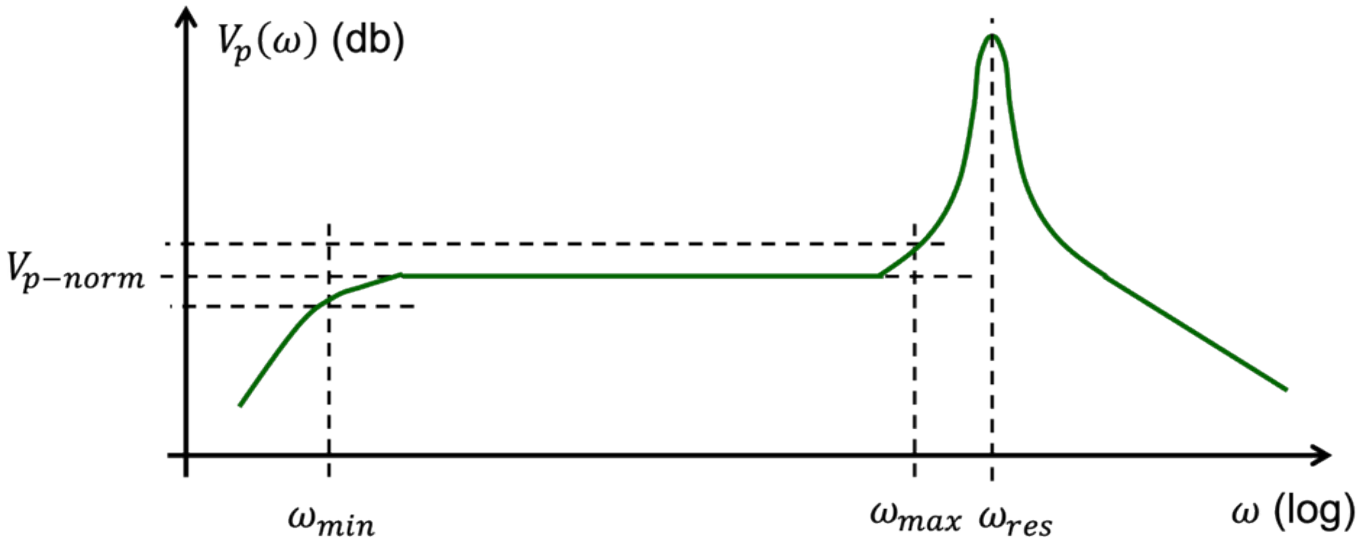


Figure 2.38: A piezoelectric sensor frequency response (magnitude).

To model the transfer function from the absolute acceleration $s^2X_a(s)$ to piezoelectric sensor output $V_p(s)$, we need to model the piezoelectric sensor transfer function from sensor applied force $kX(s)$ to $V_p(s)$. Based on the piezoelectric sensor frequency response in Figure 2.38, we can use the following transfer function to approximate it. Here, $G_{p1}(s)$ represents a first-order high-pass filter transfer function with its corner frequency of ω_{min}, and $G_{p2}(s)$ represents a second-order system transfer function with its natural frequency of ω_{max} and damping ratio of ζ. Note that the frequency response shown in Figure 2.38 can be obtained experimentally. As a result, $G_{p1}(s)$ and $G_{p2}(s)$ can be found based on frequency response.

$$G_p(s) = \frac{V_p(s)}{X(s)} = \underbrace{\frac{V_{p-norm}\omega_{min}s}{s+\omega_{min}}}_{G_{p1}(s)} \cdot \underbrace{\frac{\omega_{max}^2}{s^2+2\zeta\omega_{max}^2 s+\omega_{max}^2}}_{G_{p2}(s)} \cdot k \tag{2.4.44}$$

In this case the transfer function from the absolute acceleration $s^2X_a(s)$ to piezoelectric sensor output $V_p(s)$ can be generated below by substituting equation 2.4.43 into 2.4.44.

$$G_{ap}(s) = \frac{V_p(s)}{s^2X_a(s)} = \frac{V_{p-norm}\omega_{min}s}{s+\omega_{min}} \cdot \frac{\omega_{max}^2\left(ms^2+bs+k\right)/m}{s^2+2\zeta\omega_{max}^2 s+\omega_{max}^2} \cdot k \tag{2.4.45}$$

In addition, a capacitance-based accelerometer sensor will be introduced in detail in Section 6.6, utilizing MEMS (Micro-Electro-Mechanical Systems) technology.

Section Summary

This section addresses the modeling process of mechatronic systems. These include DC motors that convert electric power to mechanical power and accelerometers that convert mechanical acceleration into an electronic signal. Both DC motor and accelerometer system transfer functions are derived in detail using time-domain and frequency domain approaches. Note that the accelerometer portion of the content can be used for graduate classes.

Section Keywords

Accelerometer transfer function: This mathematical representation, typically a ratio of Laplace transforms in the s-domain, describes the relationship between the input acceleration and the sensor output voltage.

Back electromotive force coefficient (K_m): This constant relates a DC-motor rotational speed to the back EMF (electromotive force) voltage generated, which is the voltage that opposes the armature voltage applied.

DC-motor torque coefficient (K_τ): This constant value is expressed in Newton-meter per Ampere (Nm/A), which represents the

relationship between the DC-motor armature current and the produced torque (produced torque is equal to the product of K_τ and armature current).

DC-motor transfer function: This system transfer function has DC-motor armature voltage as input and motor speed or displacement as output.

Practice Problems

1. Consider the following DC-motor mechatronic system, where $e_a(t)$ is the applied armature voltage, and $e_m(t)$ is the back electromotive force voltage with back electromotive force coefficient K_m and torque coefficient K_τ.
2. Draw free body diagrams of J and m, and find their EOMs.
3. Find the Laplace transformation equations of $I_a(s)$, $T(s)$, $E_m(s)$, $\Theta(s)$, and $X(s)$.
4. Draw a block diagram of the above system.
5. Find system transfer function $\Theta(s)/E_a(s)$.
6. Find the transfer function from $R(s)$ to $Y(s)$ for the following system diagram.

This page titled 2.4: Mechatronic System Modeling is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

2.5: System Linearization

Learning Objectives

After completing this section, you will be able to:

1. Describe the motivation of linearizing a nonlinear system model.
2. Linearize a nonlinear system model with the given operational condition by following the six-step process described in this section.

This section discusses how to linearize a nonlinear system model in the time domain. Most practical physical systems are nonlinear in nature. However, most analysis and design tools for mechatronic systems are limited to linear systems. Thus, we often need to convert a nonlinear system model into a linear one near its operational condition of interest. Before diving into linearization, we need to define a linear system with $u(t)$ as input and $y(t)$ as output.

Definition 2.1

If a dynamic system is linear, the *principle of superposition* is satisfied. Assume the input is the linear combination of $u_1(t)$ and $u_2(t)$, that is, $\alpha_1 u_1(t) + \alpha_2 u_2(t)$. In this case, α_1 and α_2 are real numbers, and the associated system output is $\alpha_1 y_1(t) + \alpha_2 y_2(t)$, assuming that $y_1(t)$ and $y_2(t)$ are time responses associated with inputs $u_1(t)$ and $u_2(t)$, respectively.

Note that a nonlinear system does not satisfy the principle of superposition. Furthermore, a linear system with affine input does not satisfy the principle of superposition. Consider a linear system with affine excitation described by the differential equation 2.5.1.

$$\dot{y}(t) + y(t) = au(t) + b \tag{2.5.1}$$

Let $y(t) = y_1(t)$ and $y(t) = y_2(t)$ be output time responses associated with inputs $u(t) = u_1(t)$ and $u(t) = u_2(t)$, respectively. In this case, the time response of system 2.5.1, with input $u(t) = \alpha_1 u_1(t) + \alpha_2 u_2(t)$, is not equal to $\alpha_1 y_1(t) + \alpha_2 y_2(t)$. They are equal when $b = 0$. That is, a bias input b makes the system nonlinear.

Now we would like to introduce the concept of a time-invariant system. The system response of a time-invariant system is not affected by time. Suppose $y(t)$ is the system response with control input $u(t)$. The left plot of Figure 2.39 shows the input is shifted by time T —that is, $u(t+T)$. In turn, the associated system response is $y(t+T)$, as seen in the right plot of Figure 2.39.

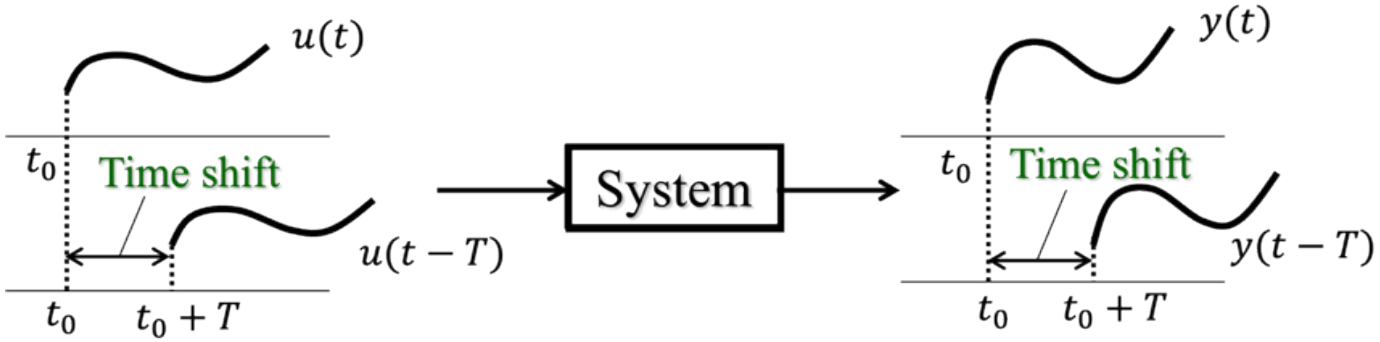


Figure 2.39: Time response of a time-shifted (delayed) input.

Definition 2.2

For a time-invariant system, suppose $y(t)$ is the system response with control input $u(t)$. When the control input is shifted by time T—that is, $u(t+T)$— the associated system response is $y(t+T)$.

A linear system that is time-invariant is called a linear time-invariant (LTI) system; it can be represented by a transfer function in the form of a Laplace transform. An LTI system is easy to understand. It utilizes existing analysis and control design tools. Note that a linear system can be described by ordinary differential equations (ODEs). We can divided solutions of an ordinary different equation into two parts: homogeneous and particular responses.

The homogeneous solution is related to the system transient response introduced by the system initial condition; the particular solution is the response caused by the forced (control) input. Due to the principle of superposition, solutions of linear combinations of multiple forced inputs can be solved individually and combined afterwards. In addition, system stability (see Chapter 3) can be easily analyzed based on the system transfer function by calculating its pole locations (roots of the denominator polynomial).

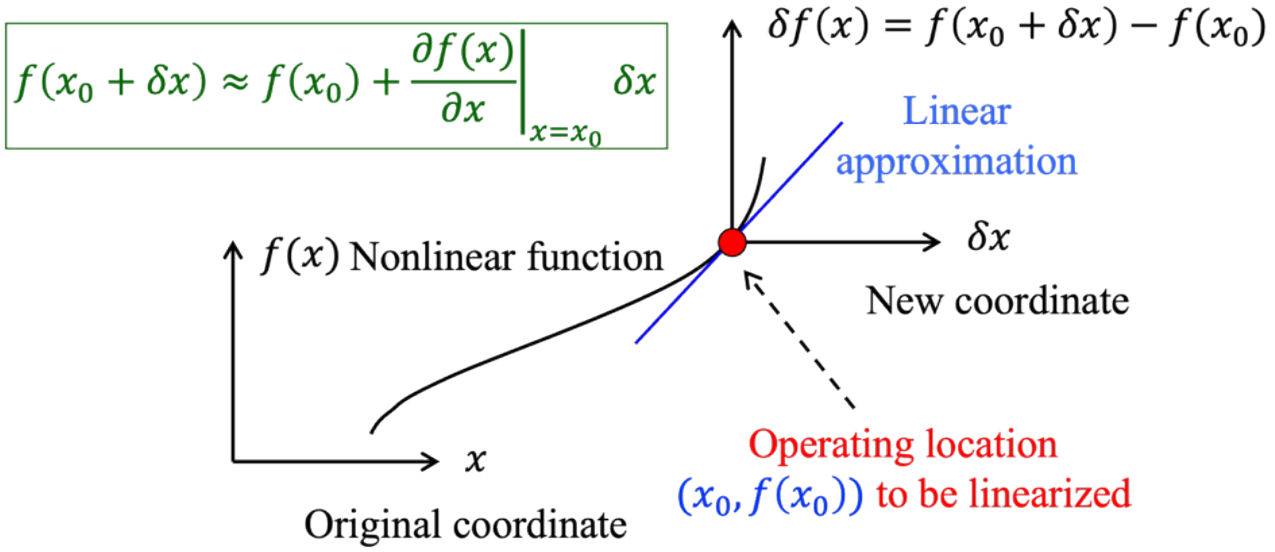


Figure 2.40: Linearization of a nonlinear function.

The actual physical system is always nonlinear, but it could be assumed to be linear over a small operational range. Often, a mechatronic system is designed to operate near the linear range. For example, a DC motor can be considered to be linear when it is operated within a certain power output range. When its power output is outside that range (e.g., beyond its output power rating), the DC-motor system is no longer linear. The main motivation of linearizing a nonlinear system is for easy control system analysis and design; many control design techniques are available only for linear system models. Note that nonlinear system models are difficult to deal with mathematically. In practice, we often linearize the nonlinear system model before conducting analysis and control design. Now the question is how to linearize a nonlinear system.

Before discussing how to linearize a nonlinear system model, we will address the question of linearizing a nonlinear function $f(x)$; see Figure 2.40. Let x_0 be the operating location to be linearized. Deploying the Taylor expansion around the operating condition x_0 leads to the following

$$f(x) = f(x_0) + \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_0} \delta x + HOT(\delta x) \tag{2.5.2}$$

where the term $HOT(\delta x)$ in equation 2.5.2 represents high order terms of $\delta x = x - x_0$. That is,

$$\lim_{\delta x \to 0} \frac{HOT(\delta x)}{\delta x} = 0 \tag{2.5.3}$$

When δx is small, we can approximate $f(x)$ by setting $HOT(\delta x)$ to zero.

$$\delta f(\delta x) = f(x) - f(x_0) \approx \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_0} \delta x \tag{2.5.4}$$

Because the above function is linear in δx, we call it a linearization of $f(x)$ at x_0. Note that the linearized function $\delta f(\delta x)$ is a function of δx (not a function of x).

For a nonlinear dynamic system model with input $u(t)$ and output $y(t)$, we can use the following six-step process for linearization.

Six-Step Linearization Process

1. Identify the system model input $u(t)$ and output $y(t)$ with $\dot{u}_0 = 0, \dot{y}_0 = 0, \ddot{u}_0 = 0, \ \ddot{y}_0 = 0, ...$
2. Express the model differential equation in the form of $f(u, \ \dot{u}, ..., y, \dot{y}, ...) = 0$ by shifting the terms on the right side to left.
3. Find equilibrium operational points u_0 and y_0 and set all their corresponding derivatives to zero at the equilibrium point $(\dot{u}_0 = 0, \ddot{u}_0 = 0, ..., \dot{y}_0 = 0, \ \ddot{y}_0 = 0, ...)$.
4. Perform Taylor expansion about (u_0, y_0) and retain only the first-order derivatives.
5. Change the system variables to their incremental ones as below.

$$\delta u = u - u_0, \ \delta \dot{u} = \dot{u} - \dot{u}_0, \ \ ...; \tilde{y} = y - y_0, \ \dot{\tilde{y}} = \dot{y} - \dot{y}_0, \ \ ... \tag{2.5.5}$$

We will use an example below to illustrate how to use the six-step process to linearize a nonlinear system model.

Example 2.16

For the pendulum system shown in Figure 2.41, derive the nonlinear system equation of motion, linearize it at $\theta_o = \pi$ (an inversed pendulum), and find the transfer function $G(s)$ from $\Delta T(s)$ to $\Delta\Theta(s)$.

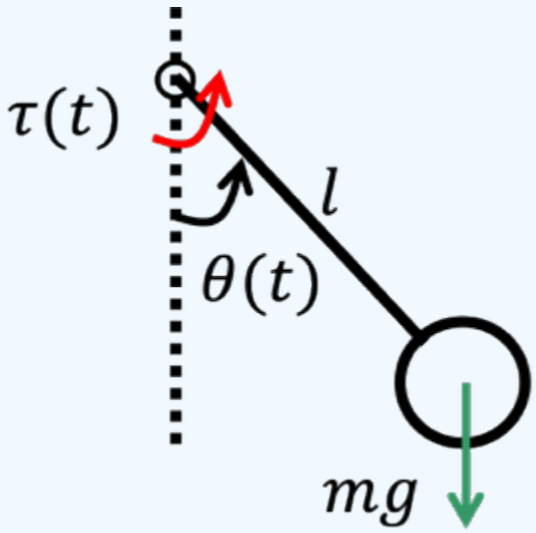


Figure 2.41: Pendulum system of Example 2.16.

Using Newton's law, we have the following equation of motion.

$$ml^2\ddot{\theta}(t) + mgl\sin\theta(t) \ = \tau(t) \tag{2.5.6}$$

Now, we will linearize equation 2.5.6 around $\theta_o = \pi$ using the six-step process.

1. Identify the system model input $\tau(t)$ and output $\theta(t)$.
2. Express the model in the form of $f\left(\tau, \ \dot{\tau}, ..., \theta, \dot{\theta}, ...\right) = 0$.

$$f\left(\theta,\ddot{\theta},\tau\right)=ml^2\ddot{\theta}\left(t\right)+mgl\sin\theta(t)\ -\tau\left(t\right)=0 \tag{2.5.7}$$

1. Find the equilibrium operation points θ_0 and τ_0 with $\theta_0=\pi$, $\ddot{\theta}_0=0$.

$$ml^2\ddot{\theta}_0+mgl\sin\theta_0\ -\tau_0=0\ \rightarrow\tau_0=0 \tag{2.5.8}$$

1. Perform Taylor expansion about $(\tau_0,\theta_0,\ddot{\theta}_0)$ with $\tau\left(t\right)=\tau_0+\delta\tau\left(t\right)$, $\theta\left(t\right)=\theta_0+\delta\theta\left(t\right)$, and $\ddot{\theta}\left(t\right)=\ddot{\theta}_0+\delta\ddot{\theta}\left(t\right)$, and retain only the first-order derivatives.

$$\begin{aligned} f\left(\theta,\ddot{\theta},\tau\right) &\cong \overbrace{f\left(\theta_0,\ddot{\theta}_0,\tau_0\right)}^{0}+\frac{\partial f}{\partial\theta}\Big|_{x_0}\delta\theta+\frac{\partial f}{\partial\ddot{\theta}}\Big|_{x_0}\delta\ddot{\theta}+\frac{\partial f}{\partial\tau}\Big|_{x_0}\delta\tau \\ &= mgl\cos\theta\ |_{\theta=\pi}\left(\theta-\pi\right)+ml^2|_{\ddot{\theta}=0}\left(\ddot{\theta}-0\right)-1|_{\tau=0}(\tau-0) \\ &= -mgl\left(\theta-\pi\right)+ml^2\left(\ddot{\theta}-0\right)-1\left(\tau-0\right) \end{aligned} \tag{2.5.9}$$

1. Change the system variables to their incremental ones for equation 2.5.9.

$$f\left(\theta,\ddot{\theta},\tau\right)=ml^2\delta\ddot{\theta}-mgl\delta\theta-\delta\tau=0 \tag{2.5.10}$$

1. Rewrite the function defined above into the ODE below and conduct the Laplace transform.

$$ml^2\delta\ddot{\theta}-mgl\delta\theta=\delta\tau\rightarrow\ ml^2s^2\Delta\Theta(s)-mgl\Delta\Theta(s)=\Delta T(s) \tag{2.5.11}$$

We have the following linearized transfer function.

$$G\left(s\right)=\frac{\Delta\Theta(s)}{\Delta T(s)}=\frac{1}{ml^2s^2-mgl} \tag{2.5.12}$$

With the linearized system represented by the transfer function $G(s)$, the relationship between the original nonlinear and linearized systems can be shown in Figure 2.42.

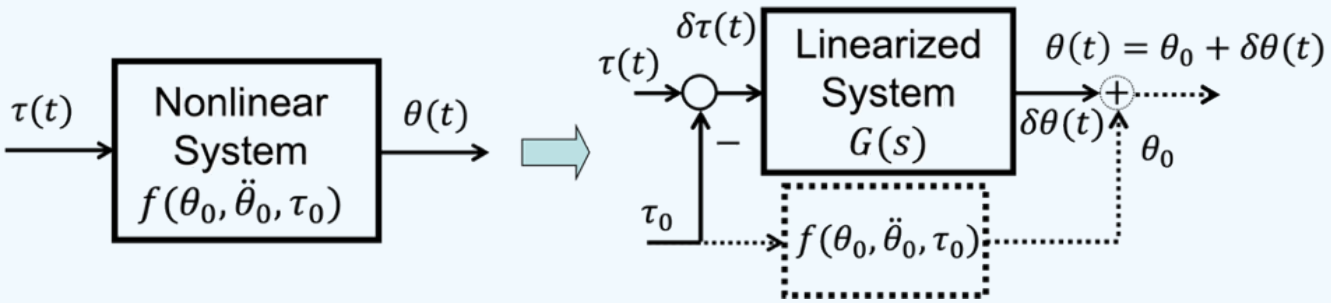


Figure 2.42: Relationship between nonlinear and linearized systems.

Note that the linearized model represented by the transfer function $G(s)$ describes only the system relationship from $\delta\tau(t)$ to $\delta\theta(t)$.

Example 2.17

Find the nonlinear differential equation described by the following circuit in Figure 2.43, linearize it at $v_i^0=10v$, and find the transfer function $G\left(s\right)$ from $\Delta V_i(s)$ to $\Delta V_o(s)$, where $R_1=1\Omega$, $R_{21}=1\Omega$, $R_{22}=200\Omega/A^2$, and $C=0.1F$.

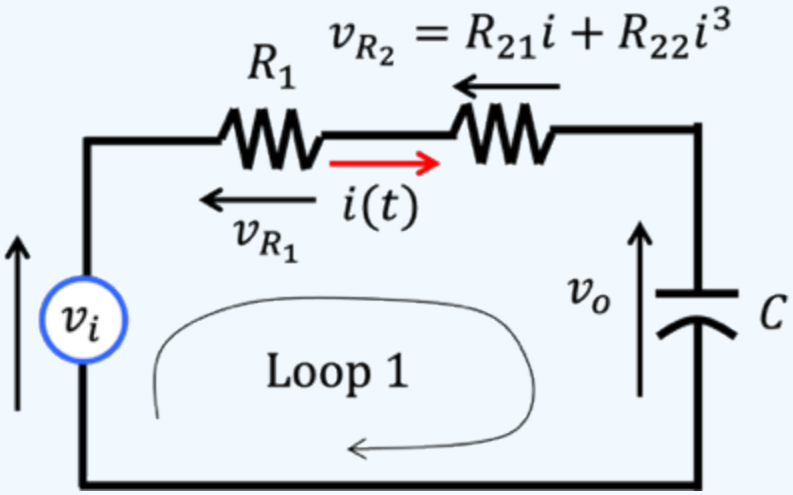


Figure 2.43: A nonlinear circuit for Example 2.17.

Using Kirchhoff's voltage law, we can calculate the output voltage $v_o(t)$ based on the capacitance formula by the following equation, assuming that $v_o(0) = 0$.

$$v_o(t) = \frac{1}{C}\int_0^t i(\tau)\,d\tau \rightarrow i(t) = C\dot{v}_o(t) \tag{2.5.13}$$

and $v_i(t)$ is

$$v_i(t) = (R_1 + R_{21})\,i(t) + R_{22}i^3(t) + v_o(t) \tag{2.5.14}$$

Substituting equation 2.5.13 into 2.5.14 yields the following.

$$v_i(t) = (R_1 + R_{21})\,C\dot{v}_o(t) + R_{22}C^3\dot{v}_o^3(t) + v_o(t) \tag{2.5.15}$$

In turn, substituting $R_1 = 1\Omega$, $R_{21} = 1\Omega$, $R_{22} = 200\Omega/A^2$, and $C = 0.1F$ into the above equation leads to the following nonlinear differential equation.

$$v_i(t) = 0.2\dot{v}_o(t) + 0.2\dot{v}_o^3(t) + v_o(t) \tag{2.5.16}$$

Now, we are ready to linearize the above nonlinear equation at $v_i^0 = 10V$ using the six-step process.

1. Identify the system model input $v_i(t)$ and output $v_o(t)$.
2. Express the model in the form of $f(v_i,\ \dot{v}_i, \ldots, v_o, \dot{v}_o, \ldots) = 0$.

$$f(v_i, v_o, \dot{v}_o) = 0.2\dot{v}_o(t) + 0.2\dot{v}_o^3(t) + v_o(t) - v_i(t) = 0 \tag{2.5.17}$$

1. Find equilibrium operation point v_o^0 with $v_i^0 = 10\text{V}$, $\dot{v}_o^0 = 0$.

$$0.2 \times 0 + 0.2 \times 0^3 + v_o(t) - 10 = 0 \ \rightarrow v_o^0 = 10V \tag{2.5.18}$$

1. Perform Taylor expansion about $x_0 = (v_i^0, v_o^0,\ \dot{v}_o^0)$ with $v_i(t) = v_i^0 + \delta v_i(t)$, $v_o(t) = v_o^0 + \delta v_o(t)$, and $\dot{v}_o(t) = \dot{v}_o^0 + \delta\dot{v}_o(t)$, and retain only the first-order derivatives.

$$\begin{aligned} f(v_i, v_o, \dot{v}_o) &\cong \overbrace{f(v_i^0, v_o^0, 0)}^{0} + \left.\frac{\partial f}{\partial v_i}\right|_{x_0}\delta v_i + \left.\frac{\partial f}{\partial v_o}\right|_{x_0}\delta v_o + \left.\frac{\partial f}{\partial \dot{v}_o}\right|_{x_0}\delta\dot{v}_o. \\ &= (-1)\left(v_i - v_i^0\right) + (1)\left(v_o - v_o^0\right) + \left.(0.2 + 0.6\dot{v}_o^2)\right|_{\dot{v}_o^0=0}(\dot{v}_o - \dot{v}_o^0) \\ &= -\left(v_i - v_i^0\right) + \left(v_o - v_o^0\right) + 0.2\left(\dot{v}_o - \dot{v}_o^0\right) \end{aligned} \tag{2.5.19}$$

1. Change the system variables to their incremental ones for equation 2.5.19

$$f\left(v_i, v_o, \dot{v}_o\right) = -\delta v_i + \delta v_o + 0.2\delta\dot{v}_o = 0 \tag{2.5.20}$$

1. Rewrite the function defined above into the ODE below and calculate the Laplace transform.

$$\delta v_o(t) + 0.2\delta\dot{v}_o(t) = \delta v_i(t) \;\rightarrow\; (1 + 0.2s)\Delta V_o(s) = \Delta V_i(s) \tag{2.5.21}$$

We have the following transfer function.

$$G\left(s\right) = \frac{\Delta V_o(s)}{\Delta V_i(s)} = \frac{1}{0.2s + 1} \tag{2.5.22}$$

Before ending this section, we will discuss how to linearize a time delay transfer function in the frequency domain. Consider the following time delay system.

$$y(t) = x\left(t - T_d\right) u(t - T_d) \tag{2.5.23}$$

where T_d is the time delay and $u(\,\cdot)$ is a unit step function. Applying the Laplace transform to the above equation, based upon Table 2.3, yields the following.

$$Y\left(s\right) = e^{-T_d s} X\left(s\right) \rightarrow \; \frac{Y(s)}{X(s)} = e^{-T_d s} \tag{2.5.24}$$

With the help of Taylor expansion in the frequency domain around $s = 0$, we have

$$e^{0.5T_d s} \;\approx\; e^{0.5T_d s}\big|_{s=0} + \left.\frac{\partial e^{0.5T_d s}}{\partial s}\right|_{s=0} (s - 0) = 1 + 0.5T_d s \tag{2.5.25}$$

and

$$e^{-0.5T_d s} \;\approx\; e^{-0.5T_d s}\big|_{s=0} + \left.\frac{\partial e^{-0.5T_d s}}{\partial s}\right|_{s=0} (s - 0) = 1 - 0.5T_d s \tag{2.5.26}$$

As a result,

$$e^{-T_d s} = \frac{e^{-0.5T_d s}}{e^{0.5T_d s}} \approx \frac{1 - 0.5T_d s}{1 + 0.5T_d s} \tag{2.5.27}$$

Applying the Taylor expansion to $e^{T_d s}$ directly at $s = 0$, we have the following linearized approximation.

$$e^{-T_d s} \approx \frac{1}{1 + T_d s} \tag{2.5.28}$$

Note that the approximation in equation 2.5.27 is preferred because the magnitude of the linearized model equals one at any frequency; this matches the original time delay transfer function. However, both approximated linear transfer functions 2.5.27 and 2.5.28 have the same phase characteristics, which are quite different from that of the original time delay transfer function, especially at high frequency.

Section Summary

This section introduces the concept of system model linearization. Note that all physical systems are nonlinear but are approximated or modeled as linear systems. When the system is modeled using a nonlinear model, it often needs to be linearized so that linear control theory can be applied. This section presents a six-step linearization process in time-domain, using Taylor expansion, to obtain the linearized model. Once the model has been linearized in time domain, the time-domain model can be converted into the frequency domain in the form of a transfer function by applying the Laplace transform.

Section Keywords

Linear system model: This model is a mathematical representation of a dynamic system with its output as a dynamic linear

function of its input. Thus, it obeys the principles of superposition (the response to a sum of inputs is the sum of the responses to each input) and homogeneity/scaling (scaling the input by a constant also scales the output by the same constant).

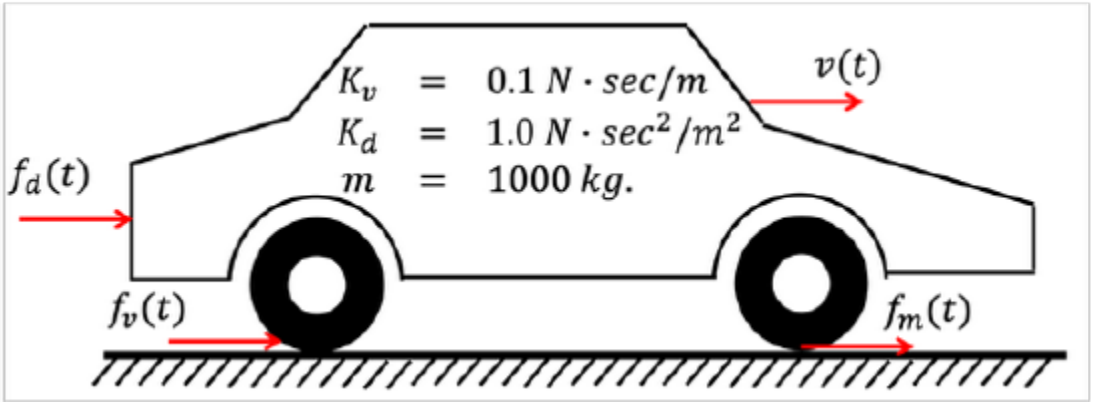
Nonlinear system model: This model describes a relationship with its dynamic output not directly proportional to the input. Thus, it does not follow the principles of superposition and homogeneity/scaling.

System model linearization: In mathematics, linearizing a nonlinear model is finding a linear model that approximates the nonlinear one at a given operational condition.

Taylor expansion: Also called Taylor series, this expansion mathematically expresses a given function as an infinite sum of terms relative to the function's derivatives at a given point (called the equilibrium point).

Practice Problems

1. Find the equation of motion for the vehicle system below, where f_m is the vehicle input traction force, $f_v(t) = K_v v(t)$ is the vehicle viscous (or rolling resistance) force, $v(t)$ is vehicle speed, and $f_d(t) = K_d v^2(t)$ is the vehicle aerodynamic drag force. Then linearize the nonlinear equation of motion at $v_0 = 30\ m/sec$, and find the linearized transfer $G(s)$ function from $\Delta F_m(s)$ to $\Delta V(s)$.



2. For the equation of motion shown below (obtained by solving 49P2.5),
   a. Solve for the operating point x_0 when $f_{a0} = 10N$ (assuming that $x > 0,\ \dot{x}_0 = \ddot{x}_0 = 0$).
   b. Derive the linearized model as a function of $\delta x = x - x_0$ and $\delta f_a = f_a - f_{a0}$, and associated transfer function $G(s) = \Delta X(s)/\Delta F_a(s)$.

$$2\ddot{x} + 6\dot{x} + 3x + x^2 = f_a(t)$$

This page titled 2.5: System Linearization is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning

Learning Objectives

After completing this section, you will be able to:

1. Define gray- and black-box modeling approaches.
2. Describe the machine learning modeling method for developing a gray-box model.
3. Use system identification to develop a block-box model.

Sections 2.2 to 2.4 introduced physical based (or physics-based) modeling techniques. There are two additional data-based modeling approaches, called gray- and black-box methods, that construct the system model based on only system input and output data. Because most of the data is collected digitally with a fixed sample rate, data-based modeling is often in the discrete-time domain described by difference equations. For instance, the following general form 2.6.1 with $n \geq m > 0$ can be used to model a single-input and single-output (SISO) system that is also called a regression model.

$$y(k) + a_1 y(k-1) + \cdots + a_n y(k-n) = b_0 u(k) + b_1 u(k-1) + \cdots + b_m u(k-m) \tag{2.6.1}$$

Note that a linear continuous-time system described by a continuous-time ODE can always be discretized using sample-and-hold devices such as A/D (analog-to-digital) and D/A (digital-to analog) converters (see Chapter 4). For a simple system, a time-derivative in an ODE can be replaced by a difference function as follows when the sample period ΔT is small enough.

$$\frac{df(t)}{dt} \approx \frac{f(k\Delta T) - f((k-1)\Delta T)}{\Delta T} \tag{2.6.2}$$

In this case, for a small enough sample period ΔT, the time derivatives can be replaced by the above difference formula to convert a continuous-time ODE into a discrete-time difference equation. Assuming that the model structure is known (that is, n and m are known), there are multiple methods in this case that can be used to find the difference equation's coefficients ($a_1, a_2, \ldots, a_n$ and $b_0, b_1, \ldots, b_m$). One commonly used method is called the Least-Squares (LS) optimization method. We will provide a simple example to demonstrate it below in Example 2.18.

Example 2.18

Let $n = 2$ and $m = 1$. Equation 2.6.2 can then be reorganized into the following.

$$y(k) = b_0 u(k) + b_1 u(k-1) - a_1 y(k-1) - a_2 y(k-2) \tag{2.6.3}$$

Using information at steps $k, k+1, \ldots, k+l$ leads to $l+1$ equations. We can organize these equations into the following matrix equation

$$\underbrace{\begin{bmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+l) \end{bmatrix}}_{Y(k)} = \underbrace{\begin{bmatrix} u(k) & u(k-1) & -y(k-1) & -y(k-2) \\ u(k+1) & u(k) & -y(k) & -y(k-1) \\ \vdots & \vdots & \vdots & \vdots \\ u(k+l) & u(k+l-1) & -y(k+l-1) & -y(k+l-2) \end{bmatrix}}_{D(k)} \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ -a_1 \\ -a_2 \end{bmatrix}}_{B} \tag{2.6.4}$$

where vector $Y(k)$ and matrix $D(k)$ can be formed using the collected experimental control input $u(k+j)$ ($j = 0, 1, \ldots, l$) and correspondent system output $y(k+j)$ ($j = 0, 1, \ldots, l$). Normally, parameter l is selected to be much greater than $n + m$ or, in other words, $l \gg m + n$. With this assumption, matrix $D(k)$ is a tall matrix with the number of rows much higher than the number of columns. Due to measurement noises in experimental data and system nonlinearity, linear equation 2.6.4 will generally not be satisfied. As a result, there is no feasible solution for B. Only an optimal approximated solution can be obtained that minimizes the error defined below

$$\left\| \begin{bmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+l) \end{bmatrix} - \begin{bmatrix} u(k) & u(k-1) & -y(k-1) & -y(k-2) \\ u(k+1) & u(k) & -y(k) & -y(k-1) \\ \vdots & \vdots & \vdots & \vdots \\ u(k+l) & u(k+l-1) & -y(k+l-1) & -y(k+l-2) \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ -a_1 \\ -a_2 \end{bmatrix} \right\|_2 \tag{2.6.5}$$

where $\|\cdot\|_2$ is a 2-norm in the Euclidian space. That is, if $x = [x_1,\ x_2, \ldots, x_n]^T$, then

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{2.6.6}$$

With the assumption that $l \gg m + n$ and matrix $D(k)$ is full column rank, the optimal solution minimizing the 2-norm in equation 2.6.5, called the Least-Squares (LS) solution, is shown below

$$B = \left(D^T(k)\, D(k)\right)^{-1} D^T(k)\, Y(k) \tag{2.6.7}$$

where the obtained parameter vector B represents the identified model defined in equation 2.6.3.

Remark 2.1

In order to have a high-quality identified model, the control input $u(k)$ should contain rich information so that the dynamic system is fully excited. One indication of a good excitation is that the condition number of matrix $D^T(k)\, D(k)$ is small; in this case, the condition number is defined as the maximum eigenvalue of $D^T(k)\, D(k)$ divided by the minimum one. Note that matrix $D^T(k)\, D(k)$ is positive definite, assuming that it is invertible. Here, the condition number of matrix $D^T(k)\, D(k)$ can be used to justify the quality of the identified gray-box system model.

Remark 2.2

The formulation in equation 2.6.4 can also be applied to nonlinear difference equations for gray-box system identification. That is, the method described in Example 2.18 is not limited to linear system identification.

Example 2.19 uses the LS method to identify a nonlinear system model defined by a nonlinear difference equation.

Example 2.19

Consider a nonlinear system described by the following nonlinear difference equation with its $n = 2$ and $m = 1$.

$$y(k) + a_1 y^3(k-1) + a_2 y(k-2) = b_0 u(k) + b_1 u^2(k-1) \tag{2.6.8}$$

Letting $l \gg m + n$ and using a similar method to that in Example 2.18, we can reorganize equation 2.6.8 into a matrix equation as follows with nonlinear terms in $D(k)$.

$$\underbrace{\begin{bmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+l) \end{bmatrix}}_{Y(k)} = \underbrace{\begin{bmatrix} u(k) & u^2(k-1) & -y^3(k-1) & -y(k-2) \\ u(k+1) & u^2(k) & -y^3(k) & -y(k-1) \\ \vdots & \vdots & \vdots & \vdots \\ u(k+l) & u^2(k+l-1) & -y^3(k+l-1) & -y(k+l-2) \end{bmatrix}}_{D(k)} \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ -a_1 \\ -a_2 \end{bmatrix}}_{B} \tag{2.6.9}$$

With the assumption that $l \gg m + n$ and matrix $D(k)$ is full column rank, we can obtain the LS optimal solution minimizing the 2-norm of $\|Y(k) - D(k)\, B\|_2$. In this case, the entries in columns 2 and 3 of matrix $D(k)$ are nonlinear.

There is a real-time system identification scheme using the iterative LS method with a forgetting factor so that the identified model will adapt to current physical system status *[1]* and *[9]*. This approach can also be used for predicting system future response (see *[19]* and *[20]* for its application to vehicle control systems).

Machine learning is now widely used for optimization based on the model obtained from the learning process. To obtain a high-quality model, the excitation needs to be designed to fully excite the physical system. The machine learning approach can fill this gap by automatically selecting the new test points to improve the current model accuracy [21] and [22]. In this section, we will discuss one type of surrogate model—the Kriging model—that is widely used in machine learning applications.

A surrogate model (static gray-box model) is cheap to evaluate because it does not include the physical dynamics that maps from inputs to outputs. Reference [23] reviews how to train a gray-box surrogate model, based on a collection of input-output data, from either simulating a detailed physical model or conducting experiments on a physical system. Among these methods, the Kriging (Gaussian process) model is now widely used. It predicts the mean value; it also predicts variance that includes both the intrinsic uncertainty inherent in a stochastic process and the extrinsic uncertainty of the unknown response surface [24].

The standard Kriging model is deterministic as it includes only model uncertainty, also called extrinsic uncertainty; it does not consider system intrinsic noises. Much literature has discussed Kriging modeling in detail [25] with a few modified for stochastic processes [24], [25], [26]. In a stochastic Kriging model, by allowing the Kriging model to regress the data instead of interpolating it, the original deterministic approach was extended to a broad one. This was accomplished by adding a regression constant, also called a regularization constant, to the leading diagonal of the Kriging correlation matrix, which will be defined in equation 2.6.10 . This section briefly introduces both deterministic and stochastic Kriging models. They can be modeled based on the extension of the Design and Analysis of Computer Experiments (DACE) MATLAB toolbox by Nielsen [27] and the extended SK code package developed by Nelson's group [24]. The reasons for using both deterministic and stochastic surrogate models will be explained in the next paragraph.

As seen in equation 2.6.10, a scalar stochastic Kriging model essentially includes three terms. The first term, $\boldsymbol{f}^T(\boldsymbol{x})\beta$, is a function of input vector $\boldsymbol{x}$ to capture the general trend. The second, $z(\boldsymbol{x})$, is a realization of a zero mean random field so that the response of unknown design sites is modeled within a statistical framework. The third, $\varepsilon(\boldsymbol{x})$, is introduced for system and measurement noise consideration to model the nature of stochastic processes. The probabilistic format of a scalar response surface model is

$$y(\boldsymbol{x}) = \boldsymbol{f}^T(\boldsymbol{x})\beta + z(\boldsymbol{x}) + \varepsilon(\boldsymbol{x}) \tag{2.6.10}$$

where $\boldsymbol{x}$ is a p dimensional vector of design variables; $\boldsymbol{f}(\boldsymbol{x})$ is a q dimensional vector of known (given) functions; and β is a constant parameter vector to be determined with compatible dimension. In addition, $z(\boldsymbol{x})$ is a statistical model of random process with zero mean and covariance based on spatial correlation functions between two design points $\boldsymbol{x}^i$and $\boldsymbol{x}^j$, where $i,\ j \in [1,\ n]$ and n is the total number of evaluation points Furthermore, $\varepsilon(\boldsymbol{x})$ is an intrinsic noise that is assumed to be Gaussian with its variance $\lambda^2(\boldsymbol{x}) = Ver(\varepsilon(\boldsymbol{x}))$ called the regularization constant.

Without noise $\varepsilon(\boldsymbol{x})$, the expression for elements of n sample covariance matrix R_n can be expressed in equation 2.6.11

$$[R_n(i,j)] = \left[Cov\left(z\left(\boldsymbol{x}^i\right), z\left(\boldsymbol{x}^j\right)\right)\right] = \sigma^2 R_d\left(\theta, \boldsymbol{x}^i, \boldsymbol{x}^j\right) \tag{2.6.11}$$

where σ^2 is the variance of the process to be determined, and R_d is an $n \times n$ correlation function matrix based on the spatial distance for all pairs $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ $(i, j \in [1, n])$. For the Kriging model, R_d is a Gaussian correlation function and can be calculated using equation 2.6.12 based on the spatial distance. Note that the correlation between points decreases as their distance increases.

$$[R_d(i,j)] = \left[Cov\left(z\left(\boldsymbol{x}^i\right), z\left(\boldsymbol{x}^j\right)\right)\right] = \left[\exp\left(-\sum_{l=1}^{p}\theta_l\left|x_l^i - x_l^j\right|^2\right)\right] \tag{2.6.12}$$

where θ_l $(1 \le l \le p)$ is the lth element of parameter vector $\theta = [\theta_1,\ \theta_2, \cdots, \theta_p]^T$ associated with the width of the Gaussian function and x_l^i $(1 \le l \le p)$ is the lth element of design variable vector $\boldsymbol{x}^i = \left[x_1^i,\ x_2^i, \cdots, x_p^i\right]^T$.

The effect of parameter vector θ can be thought of as determining how quickly the function changes as $\boldsymbol{x}^j$ moves away from $\boldsymbol{x}^i$. A large θ_l implies a significant activity, or low correlation, in dimension l. From this, with the known input-output data set D formed by n input points, an $n \times n$ covariance matrix $R_n = \sigma^2 R_d$ can be constructed.

For the stochastic case with independent noises, the corresponding covariance matrix R_s is obtained by adding regulation constant λ^2 to the diagonal components of $\sigma^2 R_d$ such that $R_s = \sigma^2 R_d + \lambda^2 I$. With all these, the model has three unknowns to be solved for the Kriging model. They are scalar σ, and parameter vectors β and θ, determined by maximizing the likelihood function in equation 2.6.13 over the observed points. To simplify the likelihood maximization, the natural logarithm will normally be taken to have the following,

$$\ln L\left(\sigma,\beta,\ \theta\right)\ =-\frac{1}{2}\left[n\ln\left(2\pi\sigma^2\right)+\ln\left(\det\left(R_s\right)\ \right)\ \right]+\frac{1}{\sigma^2}(Y-F\beta)^TR_s^{-1}\left(Y-F\beta\right) \tag{2.6.13}$$

where R_s is a covariance matrix for these observed points, $Y\ =\left[y\left(\boldsymbol{x}^1\right),y\left(\boldsymbol{x}^2\right),\ldots,y(\boldsymbol{x}^N)\right]^T$. In addition, $F\ =\left[\boldsymbol{f}^T\left(\boldsymbol{x}^1\right),\boldsymbol{f}^T\left(\boldsymbol{x}^2\right),\ldots,\boldsymbol{f}^T(\boldsymbol{x}^N)\right]^T$ is a vector formed by n observed function responses and a function matrix evaluated at n design vectors.

After the likelihood function is maximized, the stochastic Kriging model can be obtained. Accordingly, the predicted mean μ_y and mean square error s_y^2 for $y(\boldsymbol{x})$ can be calculated as

$$\begin{aligned}\mu_y\left(\boldsymbol{x}\right)&=\boldsymbol{f}^T\left(\boldsymbol{x}\right)\beta+\boldsymbol{r}^T\left(\boldsymbol{x}\right)R_s^{-1}\left(Y-F\beta\right)\\ s_y^2\left(\boldsymbol{x}\right)&=\sigma^2+\boldsymbol{u}^T\left(\boldsymbol{x}\right)\left(\boldsymbol{F}^TR_s^{-1}\boldsymbol{F}\right)^{-1}\boldsymbol{u}\left(\boldsymbol{x}\right)-\boldsymbol{r}^T\left(\boldsymbol{x}\right)R_s^{-1}\boldsymbol{r}\left(\boldsymbol{x}\right)\end{aligned} \tag{2.6.14}$$

where $\boldsymbol{r}^T\left(\boldsymbol{x}\right)=\sigma^2\left[R_d\left(\boldsymbol{x},\boldsymbol{x}^1\right),R_d\left(\boldsymbol{x},\boldsymbol{x}^2\right),\ldots,R_d\left(\boldsymbol{x},\boldsymbol{x}^n\right)\right]$ and $\boldsymbol{u}\left(\boldsymbol{x}\right)=\boldsymbol{f}\left(\boldsymbol{x}\right)-\boldsymbol{F}^TR_s^{-1}r(\boldsymbol{x})$.

Note that the exact stochastic property of $z\left(\boldsymbol{x}\right)$ is not modeled in this process. Instead, the stochastic property of $y\left(\boldsymbol{x}\right)$, required for optimization, is modeled. The general structure of deterministic and stochastic Kriging models is summarized in Figure 2.44 (solid line path). For the deterministic case, variance λ^2 is set to zero to obtain the predicted mean and mean square errors. In this case, each point provides an exact correlation with itself, forcing the predictor to interpolate evaluated points. Detailed derivation is presented in reference [25].

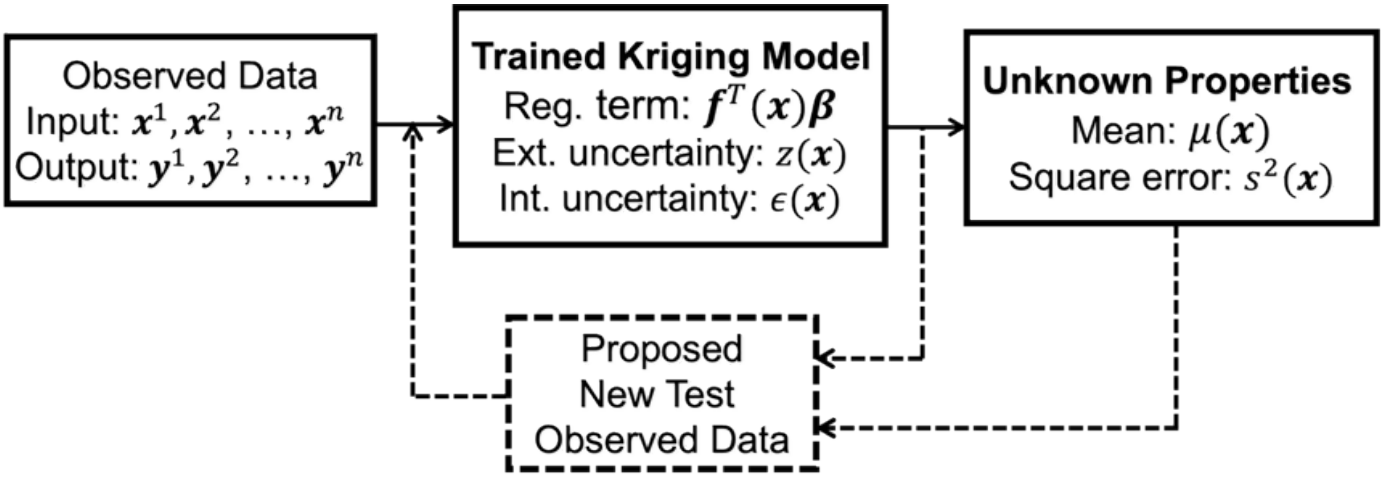


Figure 2.44: Structure of deterministic and stochastic Kriging models.

Existing research has shown that the regression function used in the Kriging model has less impact on the model performance [28]. However, the zero-order function in the ordinary Kriging model has proven to be erroneous for certain nonlinear applications [29]. The universal Kriging model adopts polynomial functions, mostly linear or quadratic, as the regression part to capture the simple spatial trend [29], [30]. A polynomial function with order higher than two should be avoided because complicated data trends are better described by the stochastic model component [30] (Gaussian process). With all these details, a surrogate Kriging model can be developed.

In order to solve the optimization problem efficiently, a surrogate-model-assisted optimization (SMAO) algorithm can be adopted to improve the model accuracy in the region near the optimum using proposed new test observation data points; see the dotted-line path in Figure 2.44. For detailed application information, see references [21] and [22].

For the black-box model, there is no assumption on the model structure. Usually, the detailed model information such as model order is determined during the system identification process based purely on system input and output data. As discussed earlier; in order to have a high-quality system identified model, the input excitation needs to contain signals over a wide bandwidth so that all system modes can be excited. If a system mode were not excited, the associated mode would not be contained in the identified model. Therefore, white noise, which has a constant frequency spectrum over the entire frequency range, is ideal for system identification (black-box modeling). However, every practical system has its physical limitations such as limited actuator magnitude. In practice, ideal white noise may not be feasible to be used because saturations on the input signals (actuators) have to be applied to avoid violating actuator constraints.

An impulse signal, such as the impulse function δ(t) introduced earlier in Section 2.1 about the Laplace transform, is also a choice for system excitation. However, its large magnitude makes it infeasible for accurate implementation. One of the input signals widely used in practice for system identification is PRBS (pseudo-random-binary-signal); its absolute magnitude remaining constant makes it feasible to be accurately implemented in practice. In the remainder of this section, we will introduce a system identification method using PRBS as the excitation to obtain a black-box model.

The most commonly used PRBS signals are based on maximum length sequences (called m-sequences) [31] for which the length of the PRBS signals is $m = 2^n - 1$, where n is an integer called the order of PRBS. Let z^{-1} represent a delay operator and define $\hat{p}\left(z^{-1}\right)$ and $p\left(z^{-1}\right)$ to be the polynomials shown below.

$$\(p\left(z^{-1}\right) = a_n z^{-n+1} \bigoplus \cdots \bigoplus a_2 z^{-1} \bigoplus a_1 = \hat{p}\left(z^{-1}\right) z^{-1} \bigoplus 1 \tag{2.6.15}$$

where a_i is either 1 or 0, and $\bigoplus$ obeys binary addition, i.e.,

$$1 \bigoplus 1 = 0 = 0 \bigoplus 0 \quad \& \; 1 \bigoplus 0 = 1 = 0 \bigoplus 1 \tag{2.6.16}$$

and the non-zero coefficients a_i of the polynomial are defined in Table 2.5 and in [31].

Table 2.5: Nonzero coefficients of PRBS polynomial.

Polynomialorder (n)	Period ofsequence (m)	Non-zeroCoefficient
2	63	a_5, a_6
3	127	a_4, a_7
4	255	a_2, a_3, a_4, a_8
5	511	a_5, a_9
6	1023	a_7, a_{10}
7	2047	a_9, a_{11}

The PRBS can then be generated by the following formula

$$\hat{u}(k+1) = \hat{p}\left(z^{-1}\right)\hat{u}(k) \tag{2.6.17}$$

where $\hat{u}\left(0\right) = 1$ and $\hat{u}\left(-1\right) = \hat{u}\left(-2\right) = \cdots = \hat{u}\left(-n\right) = 0$. Define the following sequence.

$$s(k) = \begin{cases} a; & \text{if } k \text{ is even} \\ -a; & \text{if } k \text{ is odd} \end{cases} \tag{2.6.18}$$

The signal shown in the following equation

$$u(k) = s(k) \bigotimes \left[-a + 2a\hat{u}(k)\right] \tag{2.6.19}$$

is called the *inverse* PRBS, where $\bigotimes$ obeys the following rules.

$$a \bigotimes a = -a = (-a) \bigotimes (-a)\,, a \bigotimes (-a) = a = (-a) \bigotimes a \tag{2.6.20}$$

It is clear after some analysis that $u(k)$ has a period of $2m$ and that $u\left(k\right) = -u(k+m)$. The mean of the inverse PRBS is shown below.

$$m_u = E_{2m}\left[u(k)\right] \triangleq \frac{1}{2m}\sum_{k=0}^{2m-1} u\left(k\right) = 0 \tag{2.6.21}$$

The autocorrelation of $u(k)$, which is $R_{uu}\left(\tau\right) = E_{2m}\left[u(k+\tau)u(k)\right]$, is shown below.

$$R_{uu}\left(\tau\right) = \frac{1}{2m}\sum_{k=0}^{2m-1} u\left(k+\tau\right)u\left(k\right) = \begin{cases} a^2, & \tau = 0 \quad -a^2, \quad \tau = m \\ -a^2/m, & \tau \text{ even} \\ a^2/m, & \tau \text{ odd} \end{cases} \tag{2.6.22}$$

Note that the first and second order information of the inverse PRBS signal are very close to those of the white noise for a large enough m. The inverse PRBS is used in the q-Markov Cover (q-Markov COVariance Equivalent Realization) identification algorithm, discussed in the next subsection [10]. For convenience, in the rest of this section, the term "PRBS" is used to represent the inverse PRBS.

A discrete-time asymptotically stable system can be represented in a standard discrete-time state-space model format.

$$x(k+1) \quad = \quad Ax\,(k) + Dw(k)\; y(k) \quad = \quad Cx\,(k) + Hw(k) \tag{2.6.23}$$

where $x \in \mathbb{R}^{n_x}$, $w \in \mathbb{R}^{n_w}$, and $y \in \mathbb{R}^{n_y}$ are state, input, and output vectors, respectively. Let X be the positive definite solution of the following Lyapunov equation

$$X = AXA^T + DWD^T \tag{2.6.24}$$

where W is the covariance matrix of white noise input $w(k)$. The system Markov parameter matrices ($H_i,\ i = 0,\ 1,\ \ldots,\ q-1$) and the associated covariance matrices ($R_i,\ i = 0,\ 1,\ \ldots,\ q-1$) are defined below.

$$\begin{aligned} H_0 = H,\ H_i = CA^{i-1}D,\ i = 1,\ 2,\ \ldots,\ q-1 \\ R_i = CA^iXC^T + H_iWH_i^T,\ i = 0,\ 1,\ \ldots,\ q-1 \end{aligned} \tag{2.6.25}$$

Note that these parameter matrices are dependent on system matrices A, D, C, and H. They can be calculated based on the input-output data and its autocorrelations. Thus inversely, the results will be used further to generate the system state-space model.

Because PRBS signal $u(k)$ is a periodic signal with period $2m$, its system steady-state response is also periodic with the same period as defined in equations 2.6.26 and 2.6.27.

$$x_s^j\,(k+i) = A^{k+i}x_s^j\,(0) + \sum_{n=0}^{k+i-1} A^{k+i-1-n}De_ju(n) \tag{2.6.26}$$

where $e_j = [0,\ \ldots,\ 1,\ 0,\ \ldots,0]^T$is a unit base vector with its jth element equal to 1 and the rest of elements equal to zero. In addition, $x_s^j\,(0)$ is the steady-state periodic response at the beginning of the period. The associated system output responses are shown below.

$$y_s^j\,(k+i) = Cx_s^j\,(k+i) + HDe_ju(k+i) \tag{2.6.27}$$

for $i = 0.\ 1,\ \ldots,\ q-1$, where e_j selects the jth input channel of $w(k)$ to be excited by PRBS. Now with all equations from 2.6.23 to 2.6.27, we have the following.

$$Q_qy_q^j\,(k) = Q_qO_qx_s^j\,(k) + Q_qH_qw_q^j(k) \tag{2.6.28}$$

where $Q_q = block\ diag\,[Q,\ Q,\ldots,\ Q]$ is a block diagonal matrix with q blocks of positive definite output weighting matrix Q; vectors $y_q^j\,(k)$ and $w_q^j\,(k)$ are defined in equation 2.6.29; and matrices O_q and H_q are defined in equation 2.6.30.

$$y_q^j\,(k) = \begin{bmatrix} y_s^j\,(k) \\ y_s^j\,(k+1) \\ \vdots \\ y_s^j\,(k+q-1) \end{bmatrix}, w_q^j\,(k) = \begin{bmatrix} e_ju\,(k) \\ e_ju\,(k+1) \\ \vdots \\ e_ju\,(k+q-1) \end{bmatrix} \tag{2.6.29}$$

$$O_q = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{q-1} \end{bmatrix}, H_q = \begin{bmatrix} H_0 & 0 & \cdots & 0 \\ H_1 & H_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ H_{q-1} & H_{q-2} & \cdots & H_0 \end{bmatrix} \tag{2.6.30}$$

Taking the expectation over one PRBS period from both sides of equation 2.6.28 yields the following equations from 2.6.31 to 2.6.36 for any $q > 0$.

$$Q_qR_qQ_q^T = Q_qO_qXO_q^TQ_q^T + Q_qH_qWH_q^TQ_q^T + Q_q\Delta_qQ_q^T \tag{2.6.31}$$

where Δ_q is the error term caused by finite period number $2m$ of the PRBS and based upon Theorem 3.1 in reference [10]. In addition, the Frobenius norm of Δ_q goes to zero as m goes to infinity. The rest of the matrices are defined below.

$$R_q = \begin{bmatrix} R_{yy}(0) & R_{yy}(???) & \cdots & R_{yy}(q-1) \\ R_{yy}(???) & R_{yy}(0) & \cdots & R_{yy}(q-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_{yy}(q-1) & R_{yy}(q-2) & \cdots & R_{yy}(0) \end{bmatrix} \tag{2.6.32}$$

$$W_q = block\ diag\left[W,\ W, \ldots, W\right] \tag{2.6.33}$$

where

$$R_{yy}(i) = \sum_{j=1}^{n_w} \left(\frac{1}{2m} \sum_{k=0}^{2m-1} y_s^j\,(k+i)\left[y_s^j\,(k)\right]^T \right) \tag{2.6.34}$$

$$X = \sum_{j=1}^{n_w} \left(\frac{1}{2m} \sum_{k=0}^{2m-1} x_s^j\,(k)\left[x_s^j\,(k)\right]^T \right) \tag{2.6.35}$$

$$W = \sum_{j=1}^{n_w} \left[\frac{1}{2m} \sum_{k=0}^{2m-1} e_j u^2(k) e_j^T \right] \tag{2.6.36}$$

For a sufficiently large m, we have the following.

$$Q_q R_q Q_q^T = Q_q O_q X O_q^T Q_q^T + Q_q H_q W H_q^T Q_q^T \tag{2.6.37}$$

This equation allows the use of PRBS to develop a q-Markov Cover for system identification of these system matrices defined in equation 2.6.23. As shown in Figure 2.45, we use the five steps of system identification to complete the entire identification process. The detailed q-Markov Cover system identification algorithm is provided in the next subsection.

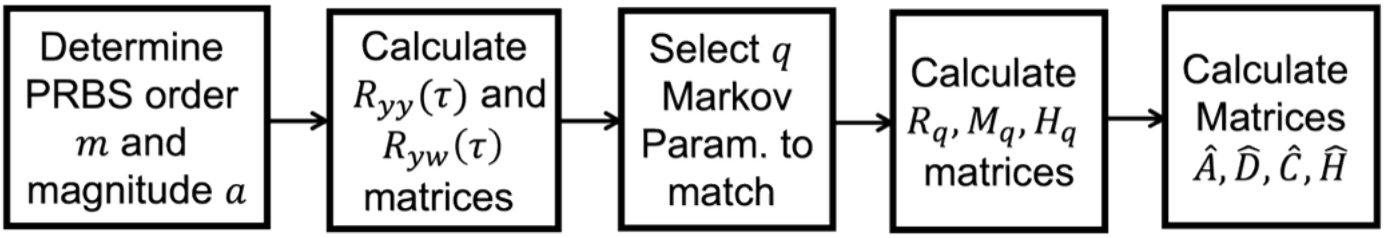


Figure 2.45: Five-steps for q-Markov Cover system identification.

The q-Markov Cover Algorithm Using PRBS

The following list provides a step-by-step process of q-Markov Cover algorithm with PRBS as input excitation signal for system identification.

1. Select parameters q (the number of Markov parameters to be matched) and m (PRBS order), and conduct experiments to obtain steady-state output response $y_s^j(k)$ $(k = 0,\ 1,\ 2,\ \ldots,\ 2m-1;\ j = 1,\ 2,\ \ldots,\ n_w)$.
2. Compute auto-correlation matrices $R_{yy}(\tau)$ and cross-correlation matrices $R_{yw}(i)$ $(i = 0,\ 1,\ 2,\ \ldots,\ q-1)$ using the following formulae.

$$R_{yy}(i) = \sum_{j=1}^{n_w} \left(\frac{1}{2m} \sum_{k=0}^{2m-1} y_s^j\,(k+i)\left[y_s^j\,(k)\right]^T \right) \backslash\backslash (R_{yw}(i) = \sum_{j=1}^{n_w} \left(\frac{1}{2m} \sum_{k=0}^{2m-1} y_s^j\,(k+i)\left[u\,(k)\,e_j\right]^T \right) \tag{2.6.38}$$

   Then calculate the Markov parameters using the following equation with W defined in 2.6.36.

$$H_i = R_{yw}\,(i)\,W^{-1}; W = a^2 I \tag{2.6.39}$$

3. Form matrices $R_q,\ M_q$, and H_q.

$$R_q = \begin{bmatrix} R_{yy}(0) & R_{yy}(???) & \cdots & R_{yy}(q-1) \\ R_{yy}(???) & R_{yy}(0) & \cdots & R_{yy}(q-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_{yy}(q-1) & R_{yy}(q-2) & \cdots & R_{yy}(0) \end{bmatrix} \tag{2.6.40}$$

$$M_q = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_{q-1} \end{bmatrix} \backslash), \backslash(H_q = \begin{bmatrix} H_0 & 0 & \cdots & 0 \\ H_1 & H_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ H_{q-1} & H_{q-2} & \cdots & H_0 \end{bmatrix} \tag{2.6.41}$$

4. Form matrix P_q by computing $D_q = R_q - H_q W_q H_q^T$, where $W_q = block\ diag\,[W,\ W,\ \ldots,\ W]$ and $W = a^2 I$. Then conduct the Schur decomposition to obtain

$$D_q = \begin{bmatrix} \hat{P}_q & \tilde{P}_q \end{bmatrix} \begin{bmatrix} \Lambda & 0 \\ 0 & \tilde{\Lambda} \end{bmatrix} \begin{bmatrix} \hat{P}_q^T \\ \tilde{P}_q^T \end{bmatrix} \tag{2.6.42}$$

such that the diagonal elements of $block\ diag\ \left[\Lambda,\ \tilde{\Lambda}\right]$ are in a decreasing order, where $\Lambda \in \mathbb{R}^{r\times r}$ and $\tilde{\Lambda}$ are diagonal matrices. Λ is chosen so that $\Lambda(\mathrm{r},\mathrm{r}) \gg \tilde{\Lambda}(1,1)$, Note that integer r is the resulting model order of the identified system. Let

$$P_q = \hat{P}_q \Lambda^{1/2} \tag{2.6.43}$$

5. Construct matrices $P,\ \overline{P}$ and $V_p U V_p^T$ by letting $P_i \in \mathbb{R}^{n_y \times r}$ and defining the following.

$$P_q^T = \left[P_0^T,\ P_1^T, \ldots, P_{q-1}^T\right], P^T = \left[P_0^T,\ P_1^T, \ldots, P_{q-2}^T\right], \overline{P}^T = \left[P_1^T,\ P_2^T, \ldots, P_{q-1}^T\right] \tag{2.6.44}$$

Conducting the singular value decomposition of P and $\left[\overline{P},\ M_q\right]$ leads to the following.

$$P = \begin{bmatrix} U_a & U_b \end{bmatrix} \begin{bmatrix} \Sigma_a & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_a^T \\ V_b^T \end{bmatrix}, \left[\overline{P},\ M_q\right] = \begin{bmatrix} U_c & U_d \end{bmatrix} \begin{bmatrix} \Sigma_c & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_c^T \\ V_d^T \end{bmatrix} \tag{2.6.45}$$

If P is full column rank, let $V_p U V_{\overline{p}}^T = 0$, and otherwise, $V_p U V_{\overline{p}}^T = V_b U V_d^T$, where U is an arbitrary unitary matrix.

6. Compute identified system matrices $\hat{A}, \hat{D}, \hat{C}$ and $\hat{H}$ by the following formulae.

$$\begin{aligned} \hat{A} &= P^+ \overline{P} + V_p U V_{\overline{p}}^T I_A;\ I_A^T = [I_r,\ 0] \\ \hat{D} &= P^+ M_q + V_p U V_{\overline{p}}^T I_D;\ I_D^T = [0,\ I_{n_w}] \\ \hat{C} &= P_0 \\ \hat{H} &= H_0 \end{aligned} \tag{2.6.46}$$

where I_r and I_{n_w} are identity matrices with dimensions of $r \times r$ and $n_w \times n_w$, respectively, and P^+ is the pseudo-inverse of matrix P. Based upon on the singular value decomposition of P in 2.6.45, P^+ can be calculated using the following formula.

$$P^+ = V_a \Sigma_a^{-1} U_a^T. \tag{2.6.47}$$

Remark 2.3

Selection of the q-Markov Cover system identification parameter r (order of identified linear system model) is important for making sure that $\Lambda(\mathrm{r},\mathrm{r}) \gg \tilde{\Lambda}(1,1)$ so that the identification error is small. Λ and $\tilde{\Lambda}$ are defined in equation 2.6.42.

Remark 2.4

Selection of the PRBS order for q-Markov Cover system identification is very important to make sure that the PRBS covers the system frequency range. Normally, the system sample rate $(1/T)$ should be ten times greater than the system's highest frequency mode. In addition, the PRBS order should be selected so that $1/(mT)$ is at least five times smaller than the system's lowest frequency mode.

Remark 2.5

Reference [10] shows that if the system responses are from a nonlinear system, the identified system model, using q-Markov Cover, represents a linearized model of that nonlinear system. That is, nonlinear system linearization can also be completed using q-Markov Cover system identification.

Section Summary

This section introduces the concepts of the gray- and black-box model identification process. The gray-box model identification generates the model coefficients with a given (derived) model structure whereas the black-box model identification generates a linear system model with its order determined during the identification process. The Least-Squares minimization approach is used as a sample approach for gray-box modeling, and the *q*-Markov Cover is used as a sample approach for black-box modeling. Note that the contents of this section can be used for graduate level class.

Section Keywords

Black-box model: This system model is constructed based upon the system input and output data, assuming no system model architecture (structure).

Gray-box model: This system model is constructed based upon the system input and output data with a given system model architecture (structure).

Least Squares (LS): This statistical and mathematical method is used to find the best-fit model (often a line or curve) for a given set of data by minimizing the sum of the squared differences (residuals) between the observed data points and the values predicted by the model.

q-Markov Cover: Short for q-Markov covariance equivalent realization, this system identification method is based upon system responses associated with either white noise, impulse, or PRBS inputs.

Practice Problems

1. Define a linear system described by the difference equation 2.6.3 with $a_1 = 0.8$, $a_2 = 0.6$, $b_0 = 1.0$, and $b_1 = 0.2$.
   a. Let $l = 20$ and $u\,(k) = \cos(0.1\pi k)\ \ (k = 0, 1,\ ..,\ l)$. Calculate $y\,(k)$ $(k = 0, 1,\ ..,\ l)$ assuming that $y\,(-1) = y\,(-2) = 0$ (zero initial condition).
   b. Use equation 2.6.4 to find data vector $Y(0)$ and matrix $D(0)$.
   c. Solve for the LS solution using equation 2.6.7 for coefficient vector $B = [b_0,\ b_1,\ \ -a_1, -a_2\ ]^T$. Then compare them with the known ones ($a_1 = 0.8$, $a_2 = 0.6$, $b_0 = 1.0$, and $b_1 = 0.2$).
   d. Add a white noise to $u(k)$ with variance of 0.01. Repeat a), b), and c), and compare with the results without noise.
2. Read references [21] and [22] to understand the Kriging gray-box modeling process and optimization processes. Then summarize them into a simple report.
3. Read references [10] and [11] to understand q-Markov Cover system identification (black-box modeling). Then program the algorithm, using a simple example to conduct *q*-Markov Cover system identification, and summarize it into a simple report.

This page titled 2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

2.7: Chapter References

References

1. R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 12th ed., Englewood Cliffs, NJ, USA: Prentice Hall, 2011, ISBN-10: 0-13-602458-0.
2. T. J. R. Hughes, *Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, 1st ed., Englewood Cliffs, NJ, USA: Prentice Hall, 1987, ISBN-13: 978-0-1331-7025-2.
3. X. Chen, G. Zhu, X. Yang, D. Hung, and X, Tan, "Model-based estimation of flow characteristics using an ionic polymer-metal composite beam," *IEEE/ASME Trans. on Mechatronics*, vol. 18, no., pp. 932–943, Jun. 2013, doi: 10.1109/TMECH.2012.2194300.
4. G. Zhu, K. M. Grigoriadis, and R. E. Skelton, "Covariance control design for the Hubble Space Telescope," *AIAA J. of Guid., Control, and Dyn.,* vol. 18, no.2, pp. 230–236, May 1995.
5. G. Rizzoni and J. Kearns, *Principles and Applications of Electrical Engineering*, 6th ed., New York, NY, USA: McGraw-Hill, 2015, ISBN-13: 978-0-0735-2959-2.
6. A. M. H. Al-Hajjar, S. S-M. Swei, and G. Zhu, "Novel LPV modeling and flutter suppression control of a smart airfoil," *IMechE J. of Syst. Control Eng.*, vol. 233, no. 6, pp. 609–624, 2019, doi: 10.1177/0959651818804377.
7. A. K. Al-Jiboory and G. Zhu, "Mode shape matching for LPV modeling to handle mode veering phenomena," *Int. J. of Dyn. And Control*, vol. 7, no. 2, pp. 469–475, 2019, doi: 10.1007/s40435-018-0469-0.
8. S. Qu, G. Zhu, W. Su, and S. S. Swei, "LPV-based transition flight control design for a tilt-rotor aircraft," *Proc. of IMechE, Part G: J. of Aerosp. Eng.*, vol. 236, no. 16, pp. 3354–3369, Dec. 2022, doi: 10.1177/09544100221083713.
9. L. Hua, J. Tang, H. Dourra, and G. Zhu, "A grey-box surrogate vehicle energy consumption model capable of real-time updating," *Fourth Edition of Focused Sect. on TMECH/AIM Emerg. Topics, IEEE/ASME Trans. on Mechatronics*, vol. 24, no. 4, pp. 2092–2100, Aug. 2023, doi: 10.1109/TMECH.2023.3276743.
10. G. Zhu, R. E. Skelton, and P. Li, "Q-Markov cover identification using pseudo-random binary signals," *Int. J. of Control,* vol. 62, no.6, pp. 1273–1290, 1995.
11. G. Zhu, "Weighted multi-rate q-Markov cover identification using PRBS — an application to engine systems," *Math. Problems in Eng.*, vol. 6, pp. 201–224, 2000.
12. L. Ljung: *System Identification — Theory for the User*, 2nd ed., Upper Saddle River, NJ, USA: PTR Prentice Hall, 1999, ISBN-10: 0-13-656695-2.
13. D. Mitrinović and J. Kečkić, (1984). *The Cauchy Method of Residues: Theory and Applications*, Amsterdam, The Netherlands: D. Reidel Publishing Company, 1984, ISBN-10: 9-02-771623-4.
14. Z. Chen, S. Shatara, and X. Tan, "Modeling of biomimetic robotic fish propelled by an ionic polymer-metal composite caudal fin," *IEEE/ASME Trans. on Mechatronics*, vol. 15, no. 3, pp. 448–459, 2010.
15. J. Sader, "Frequency response of cantilever beams immersed in viscous fluids with applications to the atomic force microscope," *J. of Appl. Phys.*, vol. 84, no. 1, pp. 64–76, Jul. 1998.
16. D. Williamson, R. E. Skelton, and G. Zhu, "Moment matching model reduction for multi-rate linear systems," *Int. J. of Control,* vol. 52, no. 6, pp.1279–1294, 1990.
17. J. V. Candy, "Accelerometer modeling in the state-space," *LLNL report*, Feb. 10, 2021. [Online] Available: https://www.osti.gov/servlets/purl/1777338.
18. P. Regtien and E. Dertien, *Sensors for Mechatronics*, 2nd. ed., Amsterdam, The Netherlands: Elsevier, 2018, IBSN 978-0-12-813810-6.
19. J. Yang and G. Zhu, "Adaptive recursive prediction of the desired torque for a hybrid powertrain," *IEEE Trans. on Veh. Tech.*, vol. 64, no. 8, pp. 3402–3413, Aug. 2015, doi: 10.1109/TVT.2014.2357395.
20. J. Yang and G. Zhu, "Stochastic predictive boundary management of a hybrid powertrain," *IEEE Trans. on Veh. Tech.*, vol. 65, no. 6, pp. 4700–4713, Jun. 2016, dol: 10.1109/TVT.2015.2504325.
21. A. Pal, L. Zhu, Y. Wang, and G. Zhu, "Constrained surrogate-based engine calibration using lower confidence bound," *IEEE/ASME Trans. on Mechatronics*, vol. 26, no. 6, pp. 3116–3127, Dec. 2021, doi: 10.1109/TMECH.2021.3053246.
22. J. Tang, A. Pal* D. Wen, C. Archer, J. Yi, and G. Zhu, "Stochastic Bayesian optimization for predicting borderline knock," *Int. J. of Engine Res.*, vol. 24, no. 3, pp.793–807, Mar. 2023, doi: 10.1177/14680874211065237.
23. M. J. Asher, B. F. Croke, A. J. Jakeman, et al. "A review of surrogate models and their application to groundwater modeling," *Water Resour. Res.*, vol. 51, no. 8, pp. 5957–5973, 2015.
24. B. Ankenman, B. L. Nelson, and J. Staum, "Stochastic kriging for simulation metamodeling," in *Proc. IEEE 2008 Winter Simul. Conf.*, 2008, pp. 362–370.
25. A. Forrester, A. Sobester, and A. Keane, "*Engineering Design via Surrogate Modelling: A Practical Guide*," Hoboken, NJ, USA:

Wiley, 2008, ISBN-13: 978-0-4700-0606-8.
26. A. I. Forrester, A. J. Keane, and N. W. Bressloff, "Design and analysis of noisy computer experiments," *AIAA J.*, vol. 44, no. 10, pp. 2331–2339, 2006.
27. S. N. Lophaven, H. B. Nielsen, and J. Søndergaard, "*DACE: a Matlab kriging toolbox*, *Version 2 (2002),*" Accessed: Jan. 2026. [Online] Available: http://www2.compute.dtu.dk/pubdb/edoc/imm3213.ps.
28. J. Sacks, W. J. Welch, T. J. Mitchell, et al., "Design and analysis of computer experiments," *Statistical Sci.,* vol. 4, no. 4, pp. 409–423, 1989.
29. T. Mukhopadhyay, S. Chakraborty, S. Dey, et al., "A critical assessment of kriging model variants for high-fidelity uncertainty quantification in dynamics of composite shells," *Arch. of Comput. Methods in Eng.*, vol. 24, no. 3, pp. 495–518, 2017.
30. P. J. Diggle, J. A. Tawn, and R. A. Moyeed, "Model-based geo-statistics," *J. of the Roy. Statist. Soc.: Series C (Appl. Statist.)*, vol. 47, no. 3, pp. 299–350, 1998.
31. W. W. Peterson, *Error Correcting Coding*, Cambridge, MA, USA: MIT Tech. Press, 1961, ISBN-13: 978-0-2621-6039-1.

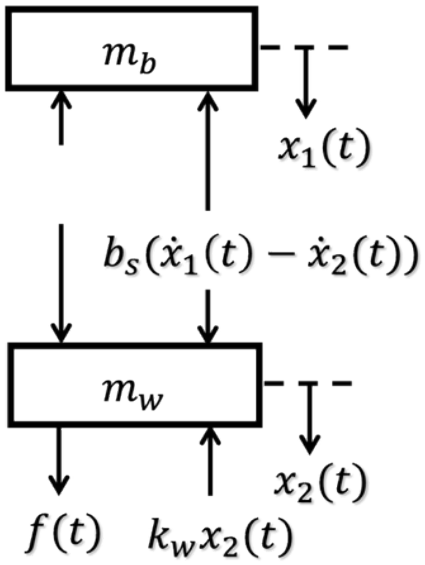
Solutions to Practice Problems

1. The solutions for P2.1 are
    1. $F(s) = \dfrac{3s+8}{(s+2)(s+3)}$
    2. $G(s) = \dfrac{2}{s^2}e^{-3s}$
    3. $F(s) = \dfrac{2}{s+2}e^{-2(s+2)}$
    4. $F(s) = 10\dfrac{(s+5)e^{-\pi(s+5)}}{(s+5)^2+1}$

1. The solutions for P2.2 are
    1. $y(t) = 8(1-e^{-2t})$
    2. $y(t) = e^{-t}\cos 2t$
    3. $f(t) = te^{-2t} + \dfrac{t^2}{2}e^{-2t}$
    4. $f(t) = \cos 2(t-3)\,u(t-3)$

1. Three-step solution process in Section 2.2:

Step 1) The quarter car free-body diagrams.



Step 2) Equations of motion

$$m_b\ddot{x}_1(t) + b_s(\dot{x}_1(t) - \dot{x}_2(t)) + k_s(x_1(t) - x_2(t)) = 0$$

$$m_w\ddot{x}_2(t) + b_s(\dot{x}_2(t) - \dot{x}_1(t)) + k_s(x_2(t) - x_1(t)) + k_w x_2(t) = f(t)$$

Step 3) Transfer functions

$$X_1(s) = \frac{b_s s + k_s}{m_b s^2 + b_s s + k_s} X_2(s)$$

$$X_2(s) = \frac{1}{m_w s^2 + b_s s + (k_s + k_w)} F(s) + \frac{b_s s + k_s}{m_w s^2 + b_s s + (k_s + k_w)} X_1(s)$$

1.

$$\frac{(J + mr^2)}{r}\ddot{x}(t) + \frac{b}{r}\dot{x}(t) = -\tau(t)$$

$$G_1(s) = \frac{\Theta(s)}{T(s)} = \frac{1}{(J + mr^2)s^2 + bs} = \frac{1}{s[(J + mr^2)s + b]}$$

$$G_2(s) = \frac{X(s)}{T(s)} = \frac{-r}{(J + mr^2)s^2 + bs} = \frac{-r}{s[(J + mr^2)s + b]}$$

1. The coefficient matrices for the five-element beam model are

$$J = J_e \begin{bmatrix} 52 & 42 & 30 & 18 & 6 \\ 42 & 40 & 30 & 18 & 6 \\ 30 & 30 & 28 & 18 & 6 \\ 18 & 18 & 18 & 16 & 6 \\ 6 & 6 & 6 & 6 & 4 \end{bmatrix}$$

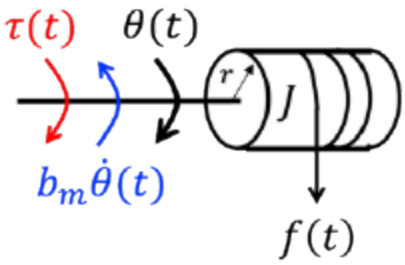
$$B = b_e \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

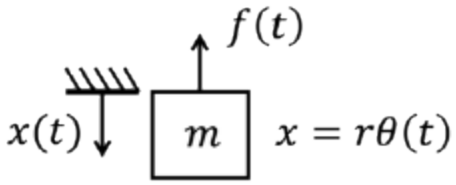
$$K = k_e \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

The fluid flow-induced torque vector Γ is expressed as

$$\Gamma(\mathrm{t}) = \frac{l}{2} \begin{bmatrix} f_1^f(t) + 2\sum_{j=2}^{5} f_j^f(t) \\ f_2^f(t) + 2\sum_{j=3}^{5} f_j^f(t) \\ f_3^f(t) + 2\sum_{j=4}^{5} f_j^f(t) \\ f_4^f(t) + 2f_5^f(t) \\ f_5^f(t) \end{bmatrix}$$

a. The following are the free body diagrams for J and m.





The equations of motion for J and m are as follows.

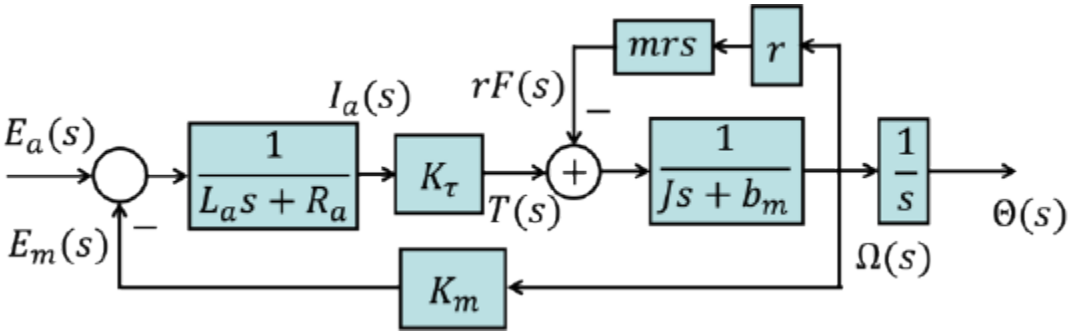
$$j\ddot{\theta}(t) + b_m\dot{\theta}(t) = \tau(t) + rf(t)$$

$$m\ddot{x}(t) = -rf(t), x(t) = r\theta(t)$$

b) The Laplace transforms of all of the equations are

$$\begin{aligned} I_a(s) &= \frac{1}{L_a s + R_a}(E_a(s) + E_m(s)) \\ T(s) &= K_\tau I_a(s), E_m(s) = K_m\Omega(s) \\ \Omega(s) &= \frac{1}{Js + b_m}(T(s) - T_l(s)), X(s) = \frac{-1}{ms^2}F(s) \\ \Omega(s) &= s\Theta(s), \; X(s) = r\Theta(s), sX(s) = r\Omega(s) \end{aligned}$$ &

c) The system block diagram is



d) The equation of motion from $E_a(s)$ to $\Theta(s)$ is

$$G(s) = \frac{\Theta(s)}{E_a(s)} = \frac{K_\tau}{s\{(L_a s + R_a)((J + ms^2)s + b_m) + K_\tau K_m\}}$$

The linearized model at $v_0 = 30\ m/sec$ is

$$1000\delta\dot{v}(t) + 60.1\delta v(t) = \delta f_m(t) \rightarrow G(s) = \frac{\Delta V(s)}{\Delta F_m(s)} = \frac{1}{1000s + 60.1}$$

a. Solution for steady state $x_0 = 2 > 0$ with $\dot{x}_0 = \ddot{x}_0 = 0$
b. The linearized differential equation and transfer function are shown below.

$$2\delta\ddot{x}(t) + 6\delta\dot{x}(t) + 7\delta x(0) = \delta f_a(t) \rightarrow \frac{\Delta X(s)}{\Delta F_a(s)} = \frac{1}{2s^2 + 6s + 7}$$

1. Complete the problem based on Example 2.19.

1. Complete the reading assignment and write a summary by following these steps below.
    1. Reading references [21] and [22] and summarizing the Kriging gray-box modeling method into a step-by-step process.
    2. Creating a MATLAB example of a physical system in your research area and generating the associated Kriging gray-box

model following the process developed in Step 1).
3. Summarizing the Kriging gray-box modeling process and your example into a technical report.

1. Complete the reading assignment and write a summary.
   1. Reading references [10] and [11] and summarizing the q-Markov Cover system identification method into a step-by-step process.
   2. Creating a MATLAB example of a physical system in your research area and generating the q-Markov Cover system model following the process developed in Step 1).
   3. Summarizing the q-Markov Cover modeling process and your identification example into a technical report.

This page titled 2.7: Chapter References is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

3: SYSTEM RESPONSES AND CONTROLS

Learning Objectives

1. Compute system time responses with impulse, step, and sinusoidal inputs using inverse Laplace transforms.
2. Define system bounded-input and bounded-output (BIBO) and asymptotical stability measures for linear time invariant (LTI) systems. Justify if the system is marginally or asymptotically stable. Understand definitions of system response measures such as overshoot, settling time, and rise time.
3. Understand response characteristics of first and second order systems and the relationship between system pole locations and performance criteria. Learn to analytically calculate response performance criteria such as overshoot and settling time.
4. Describe the principle of PID (proportional-integral-derivative) control. Conduct its gain tuning utilizing Ziegler-Nichols and direct-tuning approaches.
5. Describe the model-based control design method and the associated gain-tuning method using the output covariance control (OCC) design method.
6. Understand the model-based gain scheduled control design method based upon the LPV (linear parameter-varying) control model.

This chapter provides an introduction to control-related techniques used in mechatronic system analysis and controls, including system excitations (impulse, step, sinusoid), the first and second order system responses, system stability (bounded-input/bounded-output stability, asymptotical stability), proportional-integral-derivative (PID) control, and gain tuning. In addition, this chapter provides a brief review of model-based control design and its associated gain tuning method along with machine-learning-based control design methodology.

3.0: Prelude

3.1: Impulse, Step, and Sinusoidal Responses

3.2: System Stability and Its Performance Measures

3.3: The First and Second Order System Responses and Associated Pole Locations

3.4: PID Control and Its Gain Tuning Methods

3.5: Review of Model-Based Control Design and Gain Tuning Methods

3.6: Machine Learning-Based Control

3.7: Chapter References

This page titled 3: System Responses and Controls is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

3.0: Prelude

In Chapter 2, we discussed how to model a mechatronic system based on its physics. We also looked at both gray- and black-box approaches, where gray- and black-box approaches develop system models based on system input excitations and corresponding output responses. In this chapter, we will discuss the mechatronic system behaviors, including its stability and response performance indices. We will also address closed-loop control design and tuning of a mechatronic control system, including both model-free and model-based approaches.

Section 3.1 introduces these input-excitations used to study the performance of a mechatronic system, including impulse, step, and sinusoidal inputs. The system impulse response, in fact, is the inverse Laplace transform of the system transfer function. The step response of a mechatronic system can be associated with many performance indices such as system response time, settling time, system overshoot, and system steady state error. On the other hand, the system sinusoidal response provides its frequency response performance information [1], such as system magnitude and phase, as a function of frequency. This function provides key information for system characteristics such as phase and gain margins, system stability, and phase delay as a function of frequency.

Mechatronic system stability is a very important performance measure. Note that stability is an essential requirement for mechatronic system; if the system is not stable, the mechatronic system cannot respond to the command properly. In Section 3.2, we will define two kinds of mechatronic system stability characteristics for both linear and nonlinear mechatronic systems. These are asymptotical stability and bounded-input bounded-output (BIBO) stability. The asymptotical stability is associated with the system convergence property with any initial condition whereas BIBO stability is the characteristics of bounded system response with bounded input.

Many methods can be used to test system stability, including Lyapunov stability criteria for nonlinear and linear systems [2], closed-loop system pole locations, and system gain and phase margins for linear systems. For linear systems, the system transfer function pole locations can be used for not only stability analysis, but also for system performance study.

In Section 3.3, we analyze system response characteristics for the first and second order linear systems with associated system response time, settling time, and overshoot. For the first order systems, there is no response overshoot for the step input; we can calculate its settling time analytically using the system time constant. For the second order systems, we can calculate the system response characteristic analytically. The main difference between the first and second order systems is that there exists response overshoot for underdamped second order systems. We can approximate many high order system characteristics by those for either first or second order systems, which makes it important to become familiar with the system response characteristics of the first and second order systems. We will also analyze the relationship between system pole locations and response characteristics. These results can also be used to guide control design and tuning.

In practical applications, PID (proportional-integral-derivative) controllers are widely used due to their simplicity and easy gain tuning. In Section 3.4, we will discuss these three control parameters that make controller tuning easy. We will then address the Ziegler-Nichols PID gain tuning rule [3] in detail; normally the Ziegler-Nichols PID gains can be used as initial control gains for control gain tuning. We will also present an ad hoc (direct) PID control gain tuning method. In industrial applications, PID gains are often scheduled as a function of response error to improve closed-loop system performance. Examples will demonstrate how the gain-scheduled PID control can improve system performance.

Besides the PID control, model-based control techniques are often used for mechatronic systems. In Section 3.5, we will review the LQ (linear quadratic) and LQG (linear quadratic Gaussian) optimal control techniques [4], [5] for linear mechatronic systems. The main challenge of utilizing the LQ or LQG control for mechatronic systems is how to tune the model-based controller. Note that PID control is easy to tune because it has only three gains. We will also discuss the OCC (output covariance constraint) control technique [6], [7], [8] to address the issue of model-based control gain tuning because a family of OCC controllers can be generated with gains from low to high. To deal with nonlinear mechatronic systems, we can use the LPV (linear parameter-varying) control [9], [10], [11], [12], [13] to design gain scheduling controller for handling the mechatronic system nonlinearity.

In Section 3.6, we will review the machine-learning based scheduled PID control gain tuning. In Chapter 2, we discussed how to develop gray- and black-box models. Here, in Chapter 3, we discuss how to use machine learning to tune the gain-scheduled PID controller [15]. The gray- and black-box models can also be used for feedforward control [14]; however, they will not be addressed in this book.

Orienting Questions

1. What are the typical system excitations studied in mechatronic systems?
2. What are the characteristics of first and second order system responses and their performance criteria?
3. What are definitions of asymptotical and bounded-input/bounded-output stability?
4. What is a PID (proportional-integral-derivative) controller and how can its gains be tuned?
5. What is the process of tuning the model-based control gains?
6. How is the gain scheduling controller designed?

This page titled 3.0: Prelude is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

3.1: Impulse, Step, and Sinusoidal Responses

Learning Objectives

After completing this section, you will be able to:

1. Understand impulse, step, and sinusoidal excitations and associated responses.
2. Identify system transient and steady-state responses.
3. Use the Laplace transform to calculate system transient and steady-state responses.

This section addresses the system response characteristics due to certain excitation to mechatronic systems. Figure 3.1 shows the system input excitation and associated response relationship, where $G(s)$ is the system transfer function. Note that the system input is $u(t)$ and the output is $y(t)$. The associated Laplace transforms are $U(s)$ and $Y(s)$, respectively.

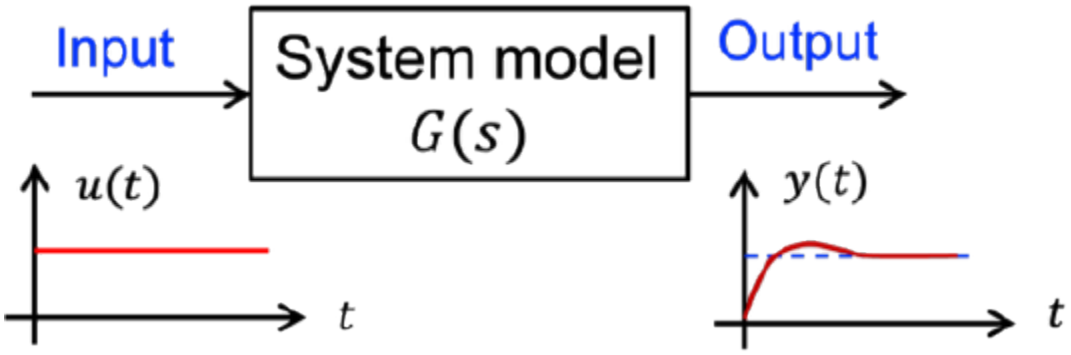


Figure 3.1: System excitation and associated responses.

Table 3.1 lists the typical input excitations and their Laplace transforms. For this section, we consider mainly three kinds of inputs: impulse, step, and sinusoid.

Table 3.1: Input excitation functions and their Laplace transforms.

Description	$f(t), \quad t \geq 0$	$F(s) \triangleq L[f(t)]$
Unit impulse	$\delta(t)$	1
Unit step	$u(t)$	$\frac{1}{s}$
Unit ramp	t	$\frac{1}{s^2}$
Unit acceleration	t^2	$\frac{2}{s^3}$
Sinusoid	$\sin\omega t$	$\frac{\omega}{s^2+\omega^2}$

First, we will consider the unit impulse input $\delta(t)$. Its Laplace transform is 1, and the system output Laplace transform is shown below.

$$Y(s) = G(s)\, L[\delta(t)] = G(s) \tag{3.1.1}$$

The time domain system response can be obtained by the inverse Laplace transform

$$y(t) = L^{-1}[G(s)] = g(t) \tag{3.1.2}$$

where $g(t)$ is often called impulse response. In this case, if $G(s)$ is stable (stability will be addressed in Section 3.2), the system output $g(t)$ will converge to zero.

Example 3.1

Let the system transfer function be defined below.

$$G(s) = \frac{2}{s+2} \tag{3.1.3}$$

Its inverse Laplace transform is then

$$g(t) = L^{-1}[G(s)] = 2e^{-2t} \tag{3.1.4}$$

Therefore, the system impulse response is $y(t) = g(t) = 2e^{-2t}$.

Now consider the case for unit step input $u(t) = 1$ with its Laplace transform

$$U(s) = L[u(t)] = \frac{1}{s} \tag{3.1.5}$$

The system output in the Laplace transform is

$$Y(s) = G(s)U(s) = \frac{G(s)}{s} \tag{3.1.6}$$

In turn, the associated time response is the inverse Laplace transform of $Y(s)$.

Example 3.2

Assuming the same system transfer function as defined in equation 3.1.3, the Laplace transform of system output $Y(s)$ is shown below and can be expanded using a partial fraction expansion technique.

$$Y(s) = \frac{2}{s(s+2)} = \frac{1}{s} - \frac{1}{s+2} \tag{3.1.7}$$

The step time response is the inverse Laplace transform of $Y(s)$ defined in equation 3.1.7.

$$y(t) = L^{-1}\left[\frac{1}{s} - \frac{1}{s+2}\right] = L^{-1}\left[\frac{1}{s}\right] - L^{-1}\left[\frac{1}{s+2}\right] = 1 - e^{-2t} \tag{3.1.8}$$

Remark 3.1

The system response $y(t)$ can be divided into two parts, transient and steady state responses, $y_t(t)$ and $y_{ss}(t)$, respectively, where

$$\lim_{t\to\infty} y_t(t) = 0, \qquad y_{ss}(t) = y(t) - y_t(t) \tag{3.1.9}$$

If the system is stable, the following final value theorem provides a way to calculate steady-state response when time goes to infinity.

Theorem 3.1[1]

The final value theorem states that if system $G(s)$ is stable, its steady-state response of the unit step input is

$$y_{ss} = \lim_{t\to\infty} y(t) = \lim_{s\to 0} sY(s) = \lim_{s\to 0} sG(s)\frac{1}{s} = \lim_{s\to 0} G(s) \tag{3.1.10}$$

Example 3.3

Consider system transfer function $G(s)$ in Example 3.2 and the associated time response below.

$$y(t) = 1 - e^{-2t} \tag{3.1.11}$$

In this case

$$y_{ss} = \lim_{t\to\infty} y(t) = \lim_{t\to\infty}(1 - e^{-2t}) = 1 \tag{3.1.12}$$

On the other hand, using the final value Theorem 3.1, we have the following.

$$y_{ss} = \lim_{s\to 0} G(s) = \lim_{s\to 0} \frac{2}{s+2} = 1 \tag{3.1.13}$$

Thus, the final value theorem provides the same final value as that by definition.

Now, let's investigate the system response with sinusoidal input. Let the Laplace transform of system sinusoidal input ($\sin\omega t$) be

$$U(s) = \frac{\omega}{s^2 + \omega^2} \tag{3.1.14}$$

where ω is the angular frequency of the sinusoidal signal. Then, the system output response is

$$y(t) = L^{-1}[G(s)U(s)] = L^{-1}\left[G(s)\frac{\omega}{s^2+\omega^2}\right] \tag{3.1.15}$$

In general, calculating the response analytically by inverse Laplace transform needs to be handled case-by-case. The following example provides a demonstration.

Example 3.4

Let the system transfer function be

$$G(s) = \frac{1}{s+1} \tag{3.1.16}$$

Assuming that the input is a sinusoidal signal with frequency $\omega = 1$, the Laplace transform of system output $y(t)$ is

$$Y(s) = G(s)U(s) = \frac{1}{s+1}\cdot\frac{1}{s^2+1} = \frac{1}{2}\left(\frac{1}{s+1} + \frac{1-s}{s^2+1}\right) \tag{3.1.17}$$

As a result, the system time response is

$$y(t) = L^{-1}[G(s)U(s)] = \frac{1}{2}L^{-1}\left[\frac{1}{s+1} + \frac{1-s}{s^2+1}\right] = \frac{1}{2}(e^{-t} + \sin t - \cos t) \tag{3.1.18}$$

and its steady-state sinusoidal response is

$$y_{ss}(t) = \frac{1}{2}(\sin t - \cos t) = \frac{1}{\sqrt{2}}\sin(t - 45^{\circ}) \tag{3.1.19}$$

From equation 3.1.19, we can see that, with sinusoidal input, the steady-state system output is also sinusoidal. It has the same frequency, but different magnitude and phase, which leads to the following theorem.

Theorem 3.2 [1]

Theorem For a sinusoidal input $u(t) = A\sin\omega t$, the steady-state output of a stable system $G(s)$ is in the following form.

$$y_{ss}(t) = A\,|G(j\omega)|\sin(\omega t + \angle G(j\omega)) \tag{3.1.20}$$

where $|G(j\omega)|$ is called the magnitude of the system transfer function and $\angle G(j\omega)$ is the phase of the transfer function.

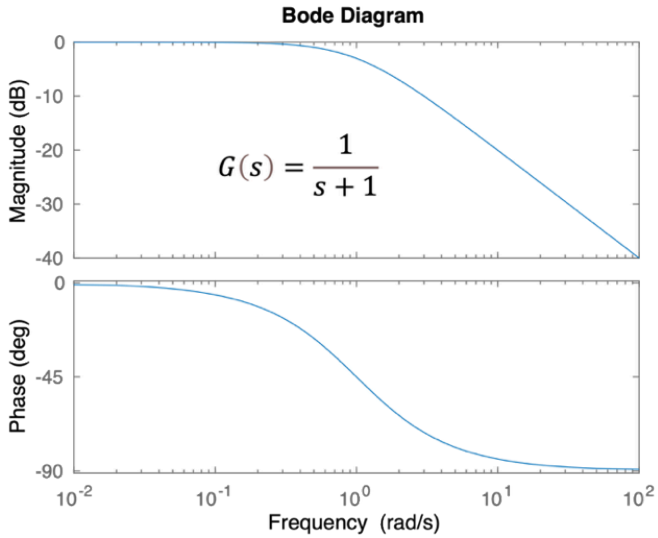


Figure 3.2: A sample system Bode plot.

The system transfer function magnitude $|G(j\omega)|$ and phase $\angle G(j\omega)$ provide important information regarding the system performance, such as relative stability. The magnitude and phase are often used to make the system Bode plot [1]. In this case, for the magnitude Bode plot, $20log_{10}|G(j\omega)|$—measured in decibel (dB)—is used for the vertical axis while $log_{10}(\omega)$ is used for the horizontal axis. For the phase Bode plot, angle (in degree) is directly used for the vertical axis while $log_{10}(\omega)$ is used for the horizontal axis. Figure 3.2 provides a typical Bode plot for the system transfer function defined in equation 3.1.16, where $G(s) = 1/(s+1)$. Note that the MATLAB command bode can be used to make the Bode plot. Also note that for the given system transfer function in equation 3.1.16, the transfer function magnitude and phase can be calculated using the following equations.

$$\begin{aligned} |G(j\omega)| &= \left(\omega^2+1\right)^{-0.5} \\ \angle G(j\omega) &= -\tan^{-1}\left(\frac{\omega}{1}\right) = -\tan^{-1}(\omega) \end{aligned} \tag{3.1.21}$$

The system Bode plot provides information such as bandwidth ω_c (also called crossover frequency) and phase delay. The Bode plot in Figure 3.2 indicates a system bandwidth of around $\omega_c = 1$ (rad/s) with a phase delay of -45°. In addition, For the frequency beyond the crossover frequency ω_c, the magnitude decays 20 dB every ten-time frequency, and the maximum phase delay is -90°. We will discuss the performance-related analysis of the Bode plot in Section 3.2.

Section Summary

This section addresses those input-excitations used to study the performance of a mechatronic system, including impulse, step, and sinusoidal inputs. Note that the impulse response is, in fact, the inverse Laplace transform of the system transfer function. The step response of a mechatronic system can be associated with many performance indices such as system response time, settling time, system overshoot, and system steady state error. The system sinusoidal response provides its frequency response performance information [1], such as system magnitude and phase, as a function of frequency These provide key information for system characteristics such as phase and gain margins, system stability, and phase delay as a function frequency.

Section Keywords

Bode plot: It provides two frequency response graphs of a system transfer function: a combination of a Bode magnitude plot with magnitude in decibels and a Bode phase plot with phase in degrees. For both plots, the horizontal axis of angular frequency is in logarithm of base 10.

Impulse response: This response is the system output with an impulse function as input.

Sinusoidal response: This response is the system output with a sinusoidal function as input.

Step response: This response is the system output with a step function as input.

Step response performance indices: These performance indices include system response time, settling time, system overshoot, and system steady state error.

Practice Problems

1. The sinusoidal system response provides system performance information in which of the following? Select the correct answer.
   a. Time domain responses.
   b. Frequency domain responses.
   c. Time domain responses such as settling time and overshoot.

This page titled 3.1: Impulse, Step, and Sinusoidal Responses is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

3.2: System Stability and Its Performance Measures

Learning Objectives

After completing this section, you will be able to:

1. Provide definitions of bounded-input/bounded-output and asymptotical stability.
2. Use Routh-Hurwitz criterion to determine system stability.
3. Describe step response performance criteria, including settling time, percentage overshoot, and rise time.

System stability is essential for mechatronic systems. It determines whether the system response will be bounded or not, which is the basic requirement for a mechatronic system. In this section, we will introduce two stability criteria: bounded-input and bounded-output (BIBO) stability and asymptotically stability. In addition, we will provide the condition making these two stability criteria equivalent. We will start with BIBO stability.

Definition 3.1

A dynamic system is bounded-input and bounded-output (BIBO) stable if for any bounded control input $\max_{t\geq 0}|u(t)|<\gamma$, where $\gamma>0$ is a positive number, the generated system output $y(t)$ is also bounded such that $\max_{t\geq 0}|y(t)|<\mu$ with zero initial condition, where $\mu>0$ is a positive number.

Figure 3.3 illustrates BIBO stability

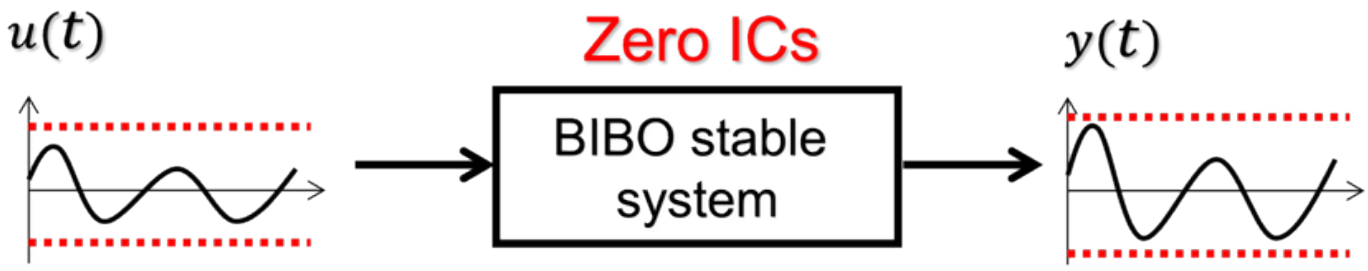


Figure 3.3: BIBO stability diagram.

Definition 3.2

A dynamic system is asymptotically stable if for any given system initial condition with zero control input, that is, $u(t)=0$), the system output $y(t)$ converges to zero as time goes to infinity.

Figure 3.4 demonstrates asymptotical stability.

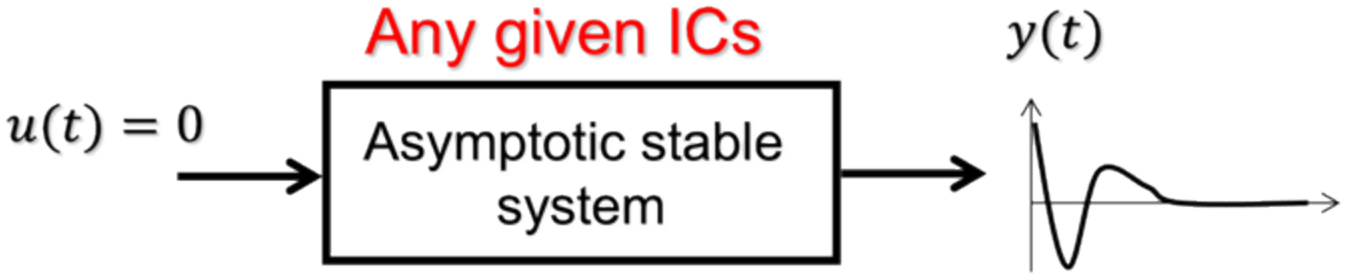


Figure 3.4: Asymptotical stability diagram.

Remark 3.2 [1]

For linear time-invariant (LTI) systems defined by a transfer function $G(s)$, the BIBO stability is equivalent to the asymptotical stability.

Now, consider an LTI system represented by the transfer function defined below.

$$G(s) = \frac{m(s)}{d(s)} \tag{3.2.1}$$

where both $m(s)$ and $d(s)$ are polynomials with order of $d(s)$ greater than or equal to that of $m(s)$. That is, $G(s)$ is proper. Let s_i ($i = 1,\ 2,\ \ldots,\ n$) be roots of the denominator's polynomial $d(s)$ defined in equation 3.2.1, also called system poles of $G(s)$; here n is the polynomial order of $d(s)$. Then $G(s)$ is BIBO and asymptotically stable if $Re(s_i) < 0$ for $i = 1,\ 2,\ \ldots,\ n$. Theorem 3.3 defines whether system $G(s)$ is asymptotically (or BIBO) stable, marginally stable, or unstable.

Theorem 3.3 [1]

For a linear time-invariant (LTI) system defined by a transfer function $G(s) = m(s)/d(s)$, let s_i ($i = 1,\ 2,\ \ldots,\ n$) be roots of the denominator's polynomial $d(s)$, where n is the order of $d(s)$. Then $G(s)$ is asymptotically (or BIBO) stable if $Re(s_i) < 0$ for $i = 1,\ 2,\ \ldots,\ n$; marginally stable if $Re(s_i) \leq 0$ for $i = 1,\ 2,\ \ldots,\ n$ with simple roots (no repeated ones) for these with $Re(s_i) = 0$; and unstable if it is neither stable nor marginally stable.

Figure 3.5 shows the system stable and unstable region of pole locations on the complex plane (s-plane). Here, if all system poles are located at the open left half plane (OLHP), the system is asymptotically (or BIBO) stable. If all system poles are located at the closed left half plane (CLHP) without any repeated poles on the on the imaginary ($j\omega$) axis, the system is marginally stable. Finally, if there is any system pole located at the open right half plane (ORHP) or repeated pole on the imaginary ($j\omega$) axis, the system is unstable. In other words, if the system is neither stable nor marginally stable, the system is unstable.

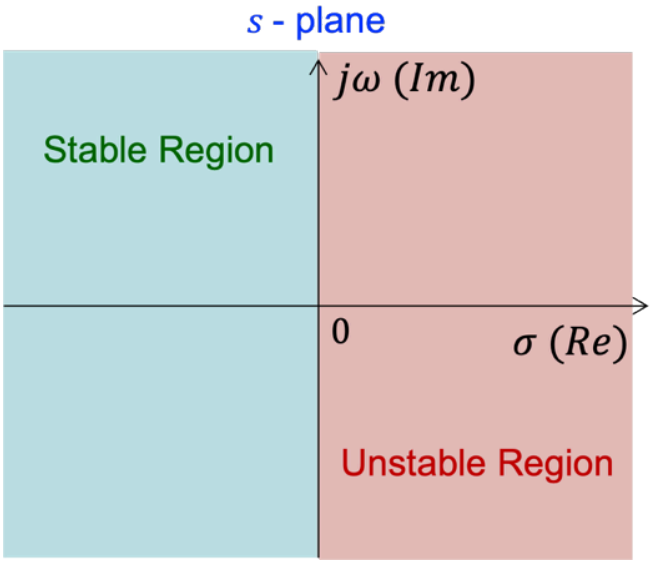


Figure 3.5: Asymptotically stable and unstable regions.

Example 3.5

Show if system transfer functions listed below are stable, marginally stable, or unstable.

$$G_1(s)=\frac{1}{s^2+1}, G_2(s)=\frac{1}{s^2+s+1},\ G_3(s)=\frac{1}{(s^2+1)^2} \tag{3.2.2}$$

For $G_1(s)$, there are two poles, $s_{1,2}=\pm j$. Because there is a pair of (nonrepeating) poles on the imaginary axis, system $G_1(s)$ is marginally stable. For $G_2(s)$, there is a pair of poles located at the open left half plane ($s_{1,2}=-0.5\pm j\sqrt{3}$), and system $G_2(s)$ is asymptotically stable. For $G_3(s)$, there are two pairs of repeated poles ($s_{1,2}=\pm j$) on the imaginary axis, and system $G_3(s)$ is unstable.

To investigate the stability of a mechatronic system described by a rational proper transfer function, the system poles of $G(s)$ can be calculated manually or using a MATLAB command [16], especially for a high-order system. Manually calculating system poles for high-order systems is challenging. Decades ago, the Routh-Hurwitz criterion [1] was developed to determine whether the roots of a polynomial are in the open left-half plane without calculating such roots. Even though nowadays software such as MATLAB can easily be used to calculate the roots of a polynomial with a fairly high degree, it turns out that the Routh-Hurwitz criterion is still useful in the analysis of feedback control systems. It can be used to determine the range of system parameters with guaranteed system stability. The Routh-Hurwitz criterion is established based on the Routh table defined below. Consider the denominator's polynomial $d(s)$ of a transfer function $G(s)$.

$$d(s)=a_n s^n+a_{n-1}s^{n-1}+\cdots+a_2 s^2+a_1 s+a_0 \tag{3.2.3}$$

Assume that $a_0\neq 0$. Note that if $a_0=0$, the polynomial can be reorganized into the following equation, with $\hat{a}_0\neq 0$.

$$d(s)=s^m\left(\hat{a}_{n-m}s^{n-m}+\hat{a}_{n-m-1}s^{n-m-1}+\cdots+\hat{a}_1 s+\hat{a}_0\right)=s^m\hat{d}(s) \tag{3.2.4}$$

Here, $\hat{d}(s)$ can be used for constructing the Routh table. Note that a Routh table can be formed for a given $d(s)$ following the rules defined in Figure 3.6, where coefficients a_i ($i=0,1,\ 2,\ \ldots,\ n$) are from the polynomial $d(s)$ defined in 3.2.3.

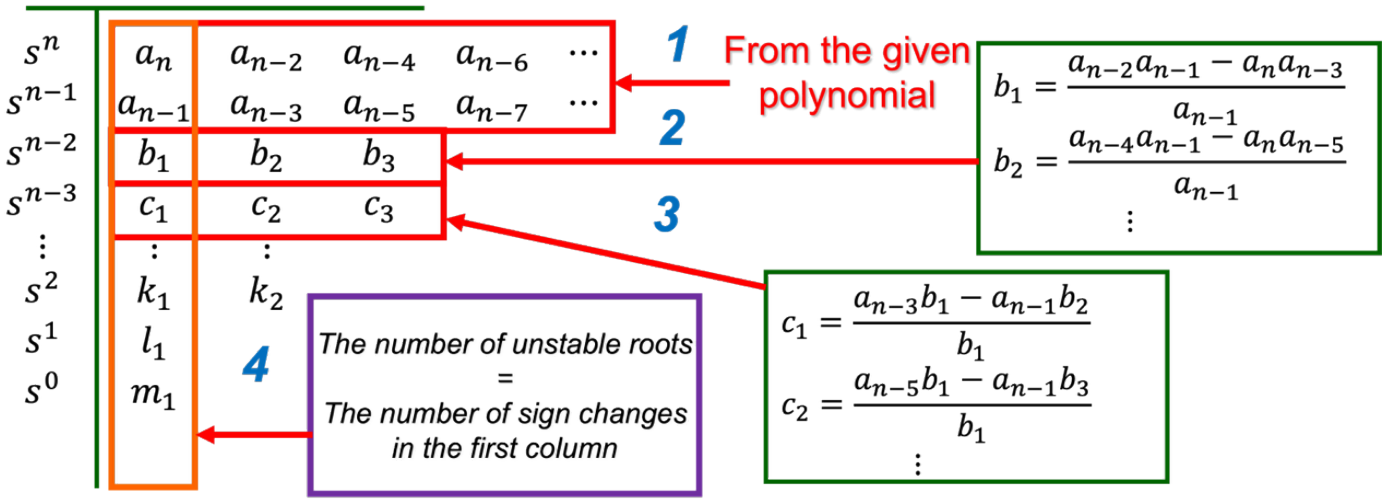


Figure 3.6: How to form a Routh table.

Theorem 3.4 [1]

The Routh-Hurwitz criterion states that the number of roots in the open right half-plane (ORHP) is equal to the number of sign changes in the first column of the Routh table.

Therefore, based on the Routh-Hurwitz criterion, the dynamic system $G(s)$ is stable if there is no sign change in the first column of the Routh table, where coefficients of 0 in the first column of Routh table need to be handled differently.

Example 3.6

Find the range of control gain K that stabilizes the closed-loop system shown in Figure 3.7.

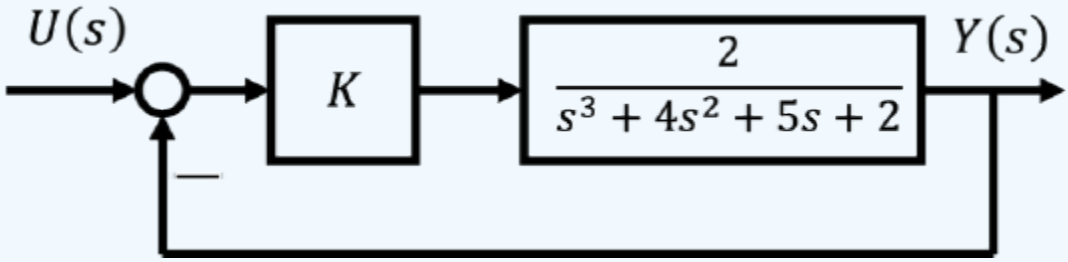


Figure 3.7: Closed-loop system for Example 3.6.

First, let's calculate the closed-loop transfer function $G(s)$ as below.

$$G(s) = \frac{n(s)}{d(s)} = \frac{2K}{s^3 + 4s^2 + 5s + 2(K+1)} \tag{3.2.5}$$

Based upon the denominator's polynomial $d(s)$ of the closed-loop transfer function, we can form the Routh table shown in Figure 3.8. To have no sign change in the first column of the Routh table, control gain K needs to be greater than -1 and less than 9.

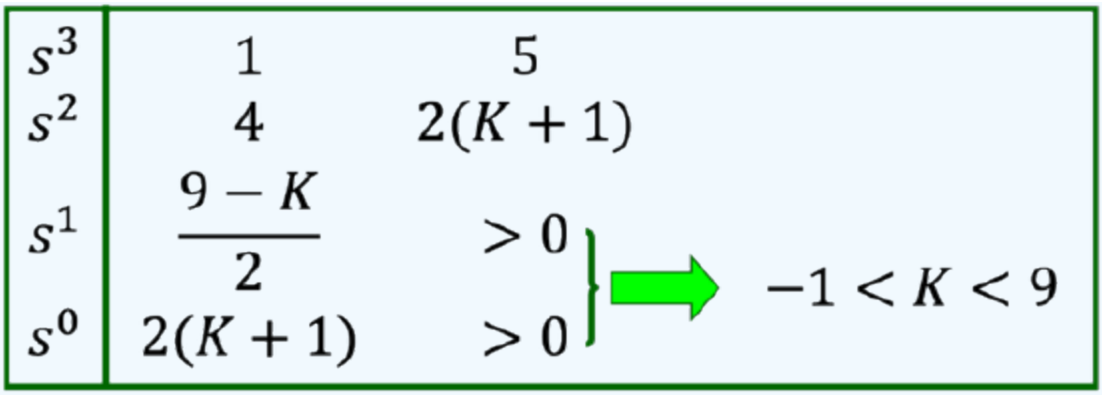


Figure 3.8: Resulting Routh table for Example 3.6.

Remark 3.3

Even though *Figure 3.6* provides ways to calculate coefficients for the Routh table, there are two special cases that need to be handled. First, if 0 appears in the first column of a nonzero row in the Routh table, replace it with a small positive number $\varepsilon > 0$. Then check for very small ε to see if there is any sign change. If so, $d(s)$ has some roots in the right half plan (RHP) (same rule in *Theorem 3.4 [1]* applies).

Second, if a zero row appears in the Routh table, $d(s)$ has roots either on the imaginary axis or in RHP. In this case, the zero row in the Routh table should be replaced by coefficients obtained by taking the derivative of the auxiliary polynomial defined by the row above; see Example 3.7. The pole locations on the imaginary or RHP can be calculated by finding roots of the auxiliary polynomial.

Examples 3.7 and 3.8 demonstrate how to deal with these two cases—with zero in the first column of a nonzero row and when a zero row appears in the Routh table.

Example 3.7

Consider the system with the following $d(s)$.

$$d(s) = s^5 + 2s^4 + 2s^3 + 4s^2 + 11s + 10 \tag{3.2.6}$$

The Routh table in Figure 3.9 can be formed based on the rules addressed in Figure 3.6 and in Remark 3.3.

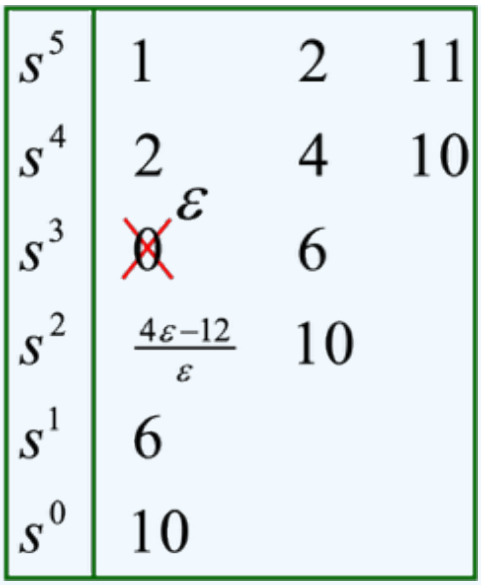


Figure 3.9: Resulting Routh table for Example 3.7.

For the s^3 row in Figure 3.9, the first entry is 0. A small positive number ε is used to replace the 0 and the calculation continues, resulting where the first entry of s^2 row is $c_1 = 4 - 12/\varepsilon$. For a very small $\varepsilon > 0$, note that $c_1 = 4 - 12/\varepsilon < 0$. As a result, there are two sign changes, leading to two poles in the open right half plane. Using MATLAB, we can obtain the following five pole locations

$$s_{1,2} = 0.8950 \pm 1.4561j, s_{3,4} = -1.2407 \pm 1.0375j,\ s_5 = -1.3087 \tag{3.2.7}$$

which confirms that there are two poles at ORLP.

Example 3.8

Consider the system with the following $d(s)$.

$$d(s) = s^5 + s^4 + 2s^3 + 2s^2 + s + 1 = (s+1)(s^2+1)^2 \tag{3.2.8}$$

The Routh table in Figure 3.10 can be formed based on the rules addressed in Figure 3.6 and in Remark 3.3.

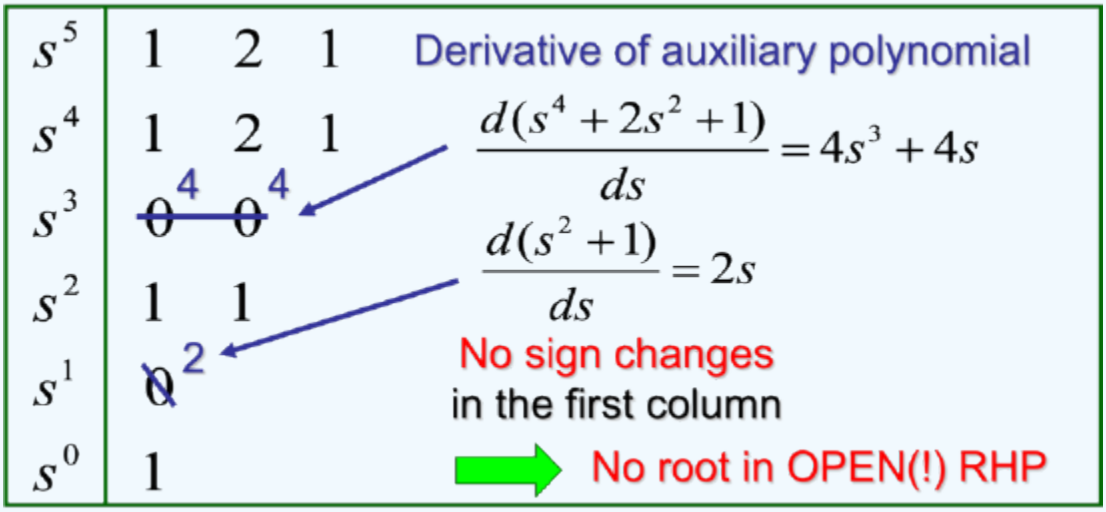


Figure 3.10: Resulting Routh table for Example 3.8.

For the zero s^3 row in Figure 3.10, we replace the two zeros by coefficients of derivative auxiliary polynomial in the line above (row of s^4). The associated auxiliary polynomial is $s^4 + 2s^2 + 1$ with its derivative of $4s^3 + 4s$. Therefore, the zero coefficients in s^3 are replaced by $[44]$ as shown. For the zero s^1 row (even though there is only one zero) in the figure, we replace the zero in row s^1 by coefficients of the derivative auxiliary polynomial in the line above (row of s^2). The associated auxiliary polynomial is $s^2 + 1$ with its derivative of $2s$. Therefore, the zero coefficients in s^1 are replaced by 2 as shown in Figure 3.10. Note that the auxiliary polynomial, $s^4 + 2s^2 + 1 = (s^2 + 1)^2$, leads to two repeated pairs of complex poles $s_{1,2} = s_{3,4} = \pm j$. From 3.2.8 there are the following five poles for $d(s)$

$$s_{1,2} = s_{3,4} = \pm j,\ s_5 = -1.0 \tag{3.2.9}$$

which confirms that there are two pairs of poles at imaginary axis. Hence, this system is unstable.

Remark 3.4

There are a few direct methods to justify whether the denominator's polynomial of a closed-loop transfer function is stable or not. For the first and second order transfer functions where $d(s) = a_1 s + a_0$ and d$(s) = a_2 s^2 + a_1 s + a_0$, all system poles are located at open left half plane (OLHP)— if and only if all coefficients ($a_1,\ a_0$ for the first order system and $a_2, a_{1,} a_0$ for the second order) have the same sign and are not equal to zero, and, as a result, the associated system is asymptotically stable. For the $d(s)$ defined in equation 3.2.3, the necessary condition for the system to be asymptotically stable is that all coefficients $(a_n, \ldots, a_{1,} a_0)$ of $d(s)$ have the same sign. These conditions can be used to find out if the system is stable or unstable.

There is also a stability measure for nonlinear systems described by the following nonlinear differential state equation; see reference [2].

$$\dot{x} = f(x, u) \tag{3.2.10}$$

where x is the system state vector, u is the system input vector, and $f(\cdot, \cdot)$ is a nonlinear function of state and control. Lyapunov stability criterion can be used in this case; see details in [2].

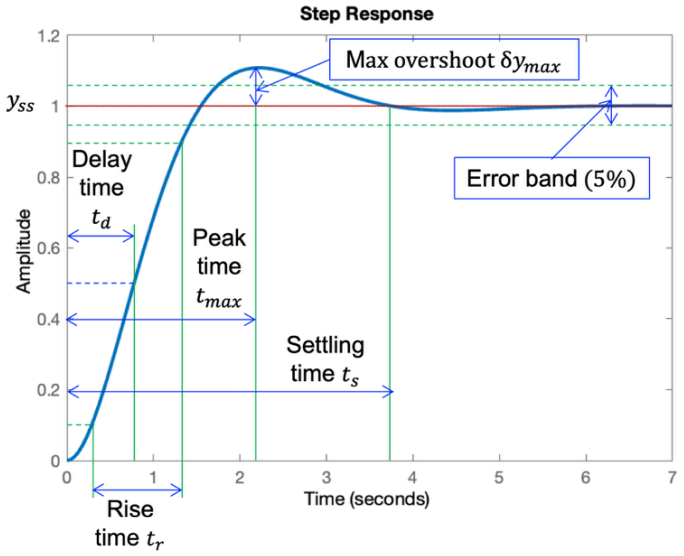


Figure 3.11: Time response performance measures.

The rest of this section discusses system performance criteria in both time and frequency domains. First, we will address the performance measures for a stable system excited by a step input. Table 3.2 shows the detailed specification of a step response.

Table 3.2: Performance measure definitions for a time-domain step response.

Performance Measures	Definition
Delay time t_d	Time to reach 50% of y_{ss} (steady state value)
Peak time t_{max}	Time to reach the peak response value
Percentage overshoot (PO)	Ratio of max overshoot $\frac{\delta y_{max}}{y_{ss}} 100\%$ (see Figure 3.11 for δy_{max})
Rise time t_r	Time to rise from 10% y_{ss} to 90% y_{ss}
Settling time (5%)	Time to reach 5% error band and remain with 5% error band
Settling time (2%)	Time to reach 2% error band and remain with 2% error band
Steady state error SS_{err}	$SS_{err} = 1 - y_{ss}$

Both Figure 3.11 and Table 3.2 define seven performance measures for a stable system excited by a unit step input, where the response speed is measured by delay time (t_d), rise time (t_r), and settling (t_s); the system's relative stability is measured by system percentage overshoot (PO). In general, fast system response leads to large overshoot. A large percentage overshoot provides small stability margin. In the control design, one needs to take a tradeoff to balance response speed and relative stability. That is, it is not practical to have extremely fast response without reducing system stability margin. To have a good stability margin, we need to keep the system percent overshoot (PO) below 15%. In Section 3.3, we will address the relationship between stability margin and closed-loop pole locations.

Next we will discuss the frequency domain stability margin measures in terms of system gain and phase margins. Before defining the system gain and phase margins, consider the concept of loop transfer function for a typical closed-loop system shown in Figure 3.12, where $G(s)$ is the system plant model, $H(s)$ the system sensor dynamics, and $C(s)$ system closed-loop controller.

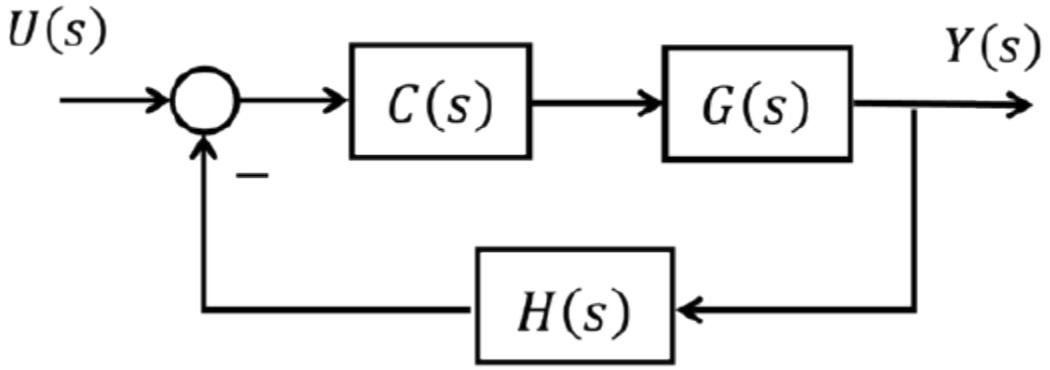


Figure 3.12: Typical closed-loop system architecture.

Define the system loop transfer function $L(s)$ as the following.

$$L(s) = C(s)\,G(s)\,H(s) \tag{3.2.11}$$

That is, the system loop transfer function is the product of all transfer functions over the feedback loop.

In this case, the system's relative stability can be measured by its loop transfer function gain and phase margins. In order to demonstrate and define these margins, consider the following loop transfer function.

$$L(s) = C(s)\,G(s)\,H(s) = \frac{10}{s} \cdot \frac{50}{s+50} \cdot \frac{5}{s+5} = \frac{2500}{s^3 + 55s^2 + 250s} \tag{3.2.12}$$

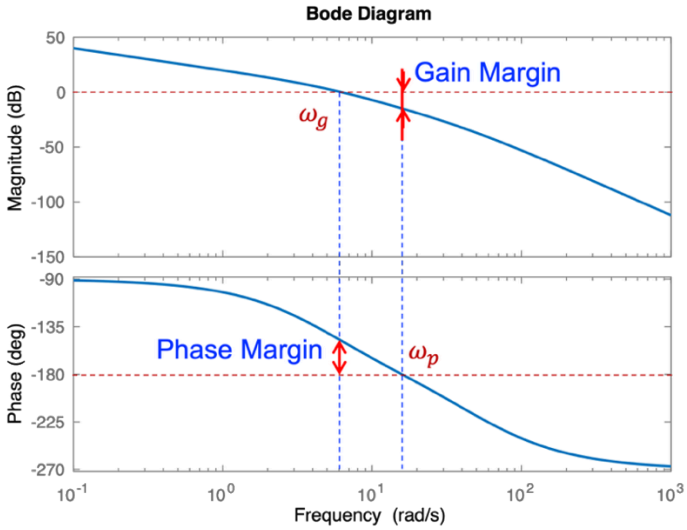


Figure 3.13: Bode plot of $L(s)$ defined in equation 3.2.12.

We can plot the Bode diagram based on the definition provided before Figure 3.2. Here, Figure 3.13 is the Bode diagram of $L(s)$ defined in equation 3.2.12. It is plotted using MATLAB with system gain and phase margins marked on the plot as "Gain Margin" and "Phase Margin" (see Definition 3.3).

Definition 3.3

A dynamic system gain margin and phase margin are defined by its Bode plot of the loop transfer function $L(s)$. The gain margin is the negative dB value of the magnitude plot at ω_p (i.e., $20log_{10}|L(j\omega_p)|$), where ω_p is the frequency when the phase plot crosses the -180° line. The phase margin is 180° plus the phase angle value at ω_g (i.e., $180^{\circ}+\angle L(j\omega_p)$), where ω_g is the frequency when the gain plot crosses the 0 dB line.

Figure 3.13 displays the Bode plot for the loop transfer function defined by equation 3.2.12. The gain and phase margins are marked on both gain and phase plots. The phase crossover frequency ω_p is the frequency when the phase plot crosses the -180° line whereas the gain crossover frequency ω_g is the frequency when the gain plot crosses the 0 dB line.

From reference [1], the closed-loop system is stable if both gain and phase margin are greater than zero. That is, as the gain margin (or phase margin) reduces down to zero, the system stability margin also goes to zero. To have a good stability margin for a closed-loop system, the system gain margin should be greater than 6 dB and the phase margin should be greater than 45°. From Figure 3.13, we can find that the gain margin for $L(s)$ defined in equation 3.2.12 is 15.4 dB and the phase margin is 45°.

Note that the controller $C(s)=10/s$ defined in equation 3.2.12 is an integral controller; it is a PID control with nonzero integral gain of only 10. We can add a proportional gain to improve the system stability margin. For example, let $C(s)=(5s+10)/s$. Then $L(s)$ becomes the following.

$$L(s)=C(s)G(s)H(s)=\frac{5s+10}{s}\cdot\frac{50}{s+50}\cdot\frac{5}{s+5}=\frac{1225s+2500}{s^3+55s^2+250s} \tag{3.2.13}$$

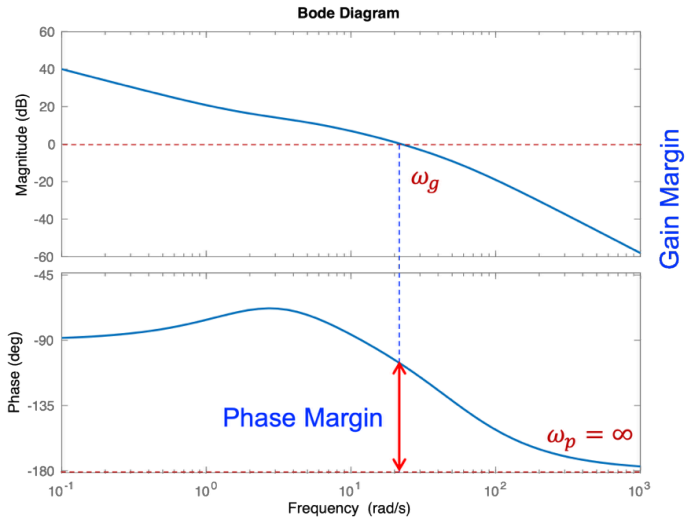


Figure 3.14: Bode plot of $L(s)$ defined in equation 3.2.13.

Figure 3.14 shows the associated Bode plot. In this case, with $\omega_p = \infty$, the gain margin is also infinity from the Bode plot shown in Figure 3.14, and the phase margin is about 105°. Using the PI (proportional-integral) control can significantly improve closed-loop system stability margin. Note that Bode plot analysis can help tune the PID gains to achieve improved stability margin.

Section Summary

Mechatronic system stability is an important performance measure because stability is an essential requirement for mechatronic systems. Without stability, the mechatronic system cannot respond to the command properly. Two kinds of mechatronic system stability characteristics are defined for both linear and nonlinear mechatronic systems. These are asymptotical stability and bounded-input bounded-output (BIBO) stability. Asymptotical stability is associated with the system convergence property with any initial condition. BIBO stability is the characteristics of bounded system response with bounded input. Many methods can be used to test system stability including Lyapunov stability criteria for nonlinear and linear systems [2]; closed-loop system pole locations and Routh-Hurwitz criterion; and system gain and phase margins for linear systems. For linear systems, the system transfer function pole locations can be used not only for stability analysis but also for system performance study.

Section Keywords

Asymptotical stability: Asymptotic stability refers to a property of a dynamical system where after being disturbed, the system will return to its equilibrium state as time approaches infinity.

Bode plot: This plot provides two frequency response graphs of a system transfer function. These are a combination of a Bode magnitude plot (with magnitude in decibels) and a Bode phase plot (with phase in degrees). For both plots, the horizontal axis of angular frequency is in logarithm of base 10.

Bounded-input and bounded-output (BIBO) stability: In this system property, any finite and limited input to a dynamic system will result in a finite and limited output.

Gain margin: This margin measures the absolute stability of a control system, indicating how much extra gain can be added before the system becomes unstable. It can be determined at the frequency where the phase shift reaches -180° A higher gain margin (expressed in dB) signifies a more stable system.

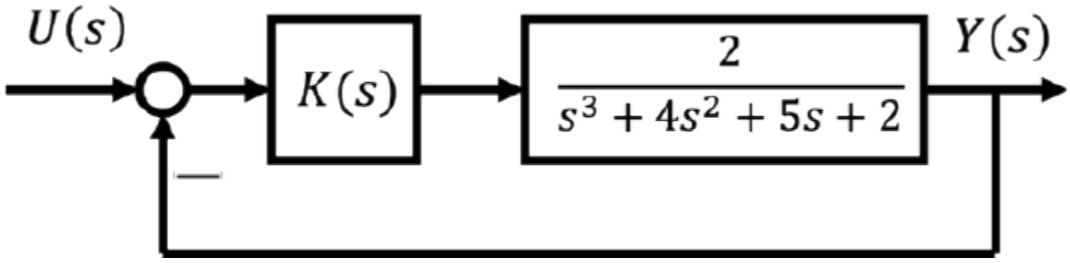
Percentage overshoot (PO): For a step input, the percentage overshoot measures how much a system response exceeds its final steady-state value, expressed as a percentage of the final value. It is calculated as the difference between the maximum peak value and the final value, divided by the final value, then multiplied by 100%.

Routh-Hurwitz criterion: This mathematical test of a control system determines if a linear time-invariant (LTI) system is stable by examining the coefficients of its characteristic equation without solving for the roots.

Settling time: For a unit step input, the system settling time (t_s) is the time for the system to reach its given error band of 2% (or 5%), where the error band is defined as the range between $1+2\%$ (or 5%) and $1-2\%$ (or 5%). Note that if 2% (or 5%) is used, this measure is called 2% (or 5%) settling time.

Practice Questions

1. With any given initial condition, which type of stability leads the system response to converge to zero? Select the correct answer.
   a. BIBO stable.
   b. Asymptotically stable.
   c. Exponentially stable.
2. With any input bounded in magnitude, which type of stability is present when the system output is also bounded in magnitude? Select the correct answer.
   a. BIBO stable,
   b. Asymptotically stable,
   c. Exponentially stable,
3. For the following system transfer functions, determine if the system is stable, marginally stable, or unstable.
   a. $G(s)=\frac{8}{s^2+1}$
   b. $G(s)=\frac{4}{s^2+2s+4}$
   c. $G(s)=\frac{4}{s(s^2+s+2)}$
   d. $G(s)=\frac{4}{s^2(s^2+4s+9)}$
   e. $G(s)=\frac{1}{s^2-1}$
4. Consider the following closed-loop system with $K(s)=3+\frac{K_I}{s}$ to be a PI (proportional-integral) controller. Find the closed-loop transfer function $(s)=\frac{n(s)}{d(s)}$, form the Routh table, and find the range of $K_I>0$ such that the closed-loop system is asymptotically stable.



5. Without calculating roots of the following $d(s)$, nor using Routh table, find out if the system is stable or unstable based on Remark 3.4.
   a. $d(s)=s^2+2s+4$
   b. $d(s)=s^2-2s+1$
   c. $d(s)=-s^2-s-2$
   d. $G(s)=s^4+s^3-3s^2+2s+2$

This page titled 3.2: System Stability and Its Performance Measures is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

3.3: The First and Second Order System Responses and Associated Pole Locations

Learning Objectives

After completing this section, you will be able to:

1. Calculate the first and second order system responses.
2. Describe the relationship between system pole locations and response performances.
3. Calculate system settling time and percentage overshoot for the first and second order systems based upon the model coefficients.

Obtaining system performance measures in general can be challenging. For first and second order systems, most system performance criteria can be found analytically. Fortunately, many physical systems can be simplified into first or second order systems for analyzing and studying system characteristics. Thus, we can study the time responses of first and second order systems and their analytical performance measures in both time and frequency domains.

First, we will address the time response and its performance measures for first order systems. Consider the following standard form of the first order transfer function.

$$G(s) = \frac{K}{Ts+1} \tag{3.3.1}$$

Here, K is the system DC gain (steady-state response y_{ss} with unit step input) and T is the system time constant. The unit step input is shown below.

$$u(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{3.3.2}$$

The Laplace transform of system output with unit step input $U(s) = L[u(t)]$ is

$$Y(s) = G(s)\,U(s) = \frac{K}{Ts+1} \cdot \frac{1}{s} \tag{3.3.3}$$

Using partial fractional expansion, we have the following.

$$Y(s) = \frac{K/T}{s(s+1/T)} = \frac{K}{s} - \frac{K}{s+1/T} \tag{3.3.4}$$

The inverse Laplace transform of the above transfer function leads to the following time response.

$$y(t) = K\left(1 - e^{-t/T}\right), t \geq 0 \tag{3.3.5}$$

When $T > 0$, the system is stable. We can calculate the steady-state response as follows.

$$y_{ss} = \lim_{t\to\infty} y(t) \ = K = \lim_{s\to 0} sY(s) \tag{3.3.6}$$

Figure 3.15 shows the plot of a typical first order system response with $K = T = 1$.

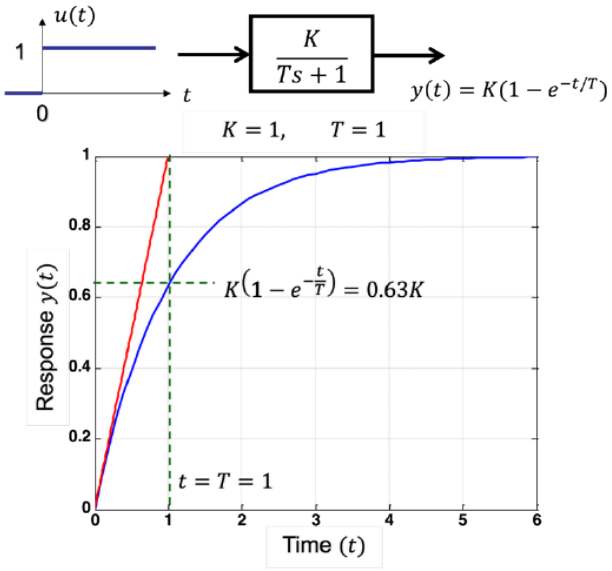


Figure 3.15: First order system time response with a unit step input.

From Figure 3.15, we can see that the system response does not have any overshoot. The maximum response will be achieved as time goes to infinity. As a result, the system does not have any overshoot (or OP = 0%) and peak time $t_{max} = \infty$; see Figure 3.11 for their definitions. As marked on Figure 3.15, at $t = T$, the system reaches about 63% of its steady state response y_{ss}. Table 3.3 provides the first order system response with an increment of one time constant.

Table 3.3: First order system response per time constant T.

Time (second)	$Ke^{-t/T}$	$K(1 - e^{-t/T})$
0	$1.0K$	0
T	$0.3679K$	$0.6321K$
$2T$	$0.1353K$	$0.8647K$
$3T$ $(t_s\,(5\%))$	$0.0498K$	$0.9502K$
$4T$ $(t_s\,(2\%))$	$0.0183K$	$0.9817K$
$5T$	$0.0067K$	$0.9933K$

Table 3.3 shows that it takes about $3T$ time to achieve 95% of the steady-state response, indicating that the 5% settling time $t_s(5\%)$ is $3T$ (three times of time constant). Similarly, the 2% settling time $t_s(2\%)$ is $4T$ (four times of time constant). The time constant is the time for the step response to reach 63% of the steady state response. As a result, the smaller the time constant T, the faster the system response.

Example 3.9

Consider the following first order system with the transfer function below. Calculate the y_{ss} and 2% settling time.

$$G\,(s) = \frac{10}{4s + 2} \tag{3.3.7}$$

First, we will convert the system transfer function into the standard form so that the time constant and DC gain can be calculated as below.

$$G\,(s) = \frac{10}{4s + 2} = \frac{5}{2s + 1} \tag{3.3.8}$$

As a result, the system DC gain is $K = 5$ and time constant $T = 2$. Therefore, we have the following.

$$y_{ss} = K = 5, t_s\,(2\%) = 4T = 8\ (\text{second}) \tag{3.3.9}$$

Now, let's study the second order system. The standard form of the second order system can be expressed as follows.

$$G\,(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.3.10}$$

where K is the system DC gain, ζ is the system damping ratio, and ω_n is the system natural frequency. Now let's consider the unit step response of the second order system. The Laplace transform of system output is shown below, assuming the underdamped damping ratio ($0 < \zeta < 1$).

$$Y\,(s) = G\,(s)\,U\,(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \cdot \frac{1}{s} = \frac{K\omega_n^2}{(s + \zeta\omega_n)^2 + (1 - \zeta^2)\omega_n^2} \cdot \frac{1}{s} \tag{3.3.11}$$

Using fractional expansion leads to

$$Y\,(s) = \frac{K}{s} - \frac{K\,(s + \zeta\omega_n)}{(s + \zeta\omega_n)^2 + \omega_d^2} - \frac{\zeta}{\sqrt{1 - \zeta^2}} \cdot \frac{K\omega_d}{(s + \zeta\omega_n)^2 + \omega_d^2} \tag{3.3.12}$$

where $\omega_d = \omega_n\sqrt{1 - \zeta^2}$ is called the **damped natural frequency**. The inverse Laplace transform yields the following.

$$y\,(t) = K\left\{1 - e^{-\zeta\omega_n t}\cos\omega_d t\ - \frac{\zeta}{\sqrt{1 - \zeta^2}} e^{-\zeta\omega_n t}\sin\omega_d t\ \right\} \tag{3.3.13}$$

We can reorganize equation 3.3.13 into the following.

$$y\,(t) = K\left\{1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1 - \zeta^2}}\left(\sqrt{1 - \zeta^2}\cos\omega_d t\ + \zeta\sin\omega_d t\ \right)\right\} \tag{3.3.14}$$

Because $0 < \zeta < 1$, letting $\sin\varphi\ = \sqrt{1 - \zeta^2}$ leads to $\cos\varphi\ = \zeta$. Substituting both $\sin\varphi$ and $\cos\varphi$ into equation 3.3.14 and using the trigonometric sum identity, $\sin\varphi\,\cos\omega_d t\ + \cos\varphi\sin\omega_d t\ = \sin\left(\omega_d t + \varphi\right),$ leads to the following simplified second order time response with unit step input.

$$y\,(t) = K\left\{1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1 - \zeta^2}}\sin\left(\omega_d t + \varphi\right)\ \right\} \tag{3.3.15}$$

where $\varphi = \cos^{-1}\zeta$. Figure 3.16 plots the system responses for $K = 1$ under the different damping ratio $0 \le \zeta \le 2$ with horizontal axis $\omega_n t$. We can see that when $\zeta = 0$, the response will not converge to its steady-state. Instead, it keeps oscillating between 0 and 2 because the system is marginally stable and the system response will not converge to its target steady state $y_{ss} = 1$. When $0 < \zeta < 1$ after a few oscillations, the response will converge to its steady state $y_{ss} = 1$. When $\zeta \ge 1$, the response will monotonically converge to its steady-state $y_{ss} = 1$ without any oscillation.

Note that equation 3.3.15 can be used for calculating the response for $0 \le \zeta < 1$, but not for $\zeta \ge 1$. In the case of $\zeta \ge 1$, the response needs to be solved following the same process, but it will be different from equation 3.3.15. Note that when $\zeta = 1$, the system has two repeated poles $s_{1,2} = -\omega_n$. Furthermore, when $\zeta > 1$, the system has two distinguished poles $s_{1,2} = \omega_n\left(-\zeta \pm \sqrt{\zeta^2 - 1}\right)$.

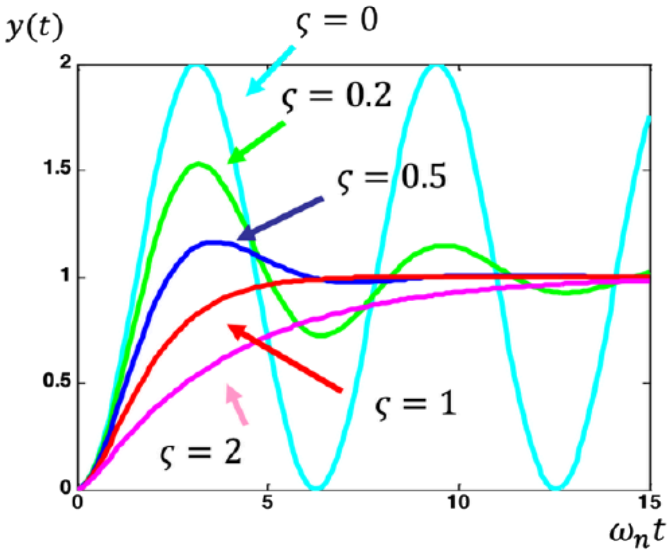


Figure 3.16: Second order system time responses with a unit step input.

From equation 3.3.15, we can find that the response of the second order system with $0 < \zeta < 1$ is bounded by the following equation with plus-sign "+" for the upper bound and negative-sign "−" for the lower bound. That is, the oscillated response with $0 < \zeta < 1$ falls between these two boundary curves.

$$y(t) = K\left\{1 \pm \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}}\right\} \tag{3.3.16}$$

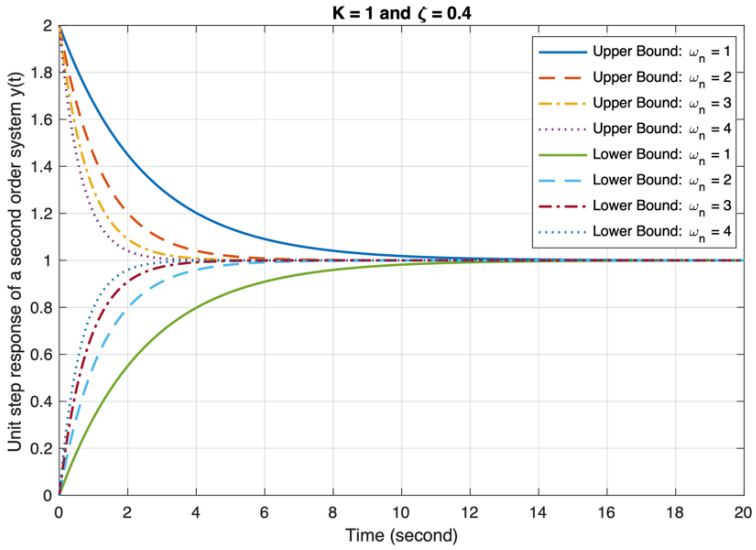


Figure 3.17: Second order system response bounds with a unit step input.

Figure 3.17 plots the boundary curves with different ω_n values for $K = 1$ and $\zeta = 0.4$. The rate of convergence increases as $\zeta\omega_n$ goes up from $\zeta\omega_n = 0.4$ to $\zeta\omega_n = 1.6$. That is, the system settling time decreases as $\zeta\omega_n$ increases. Notice the case of $\zeta\omega_n = 0.4$ ($\omega_n = 1$). Here, the calculated 2% settling time $t_s\,(2\%)$ is $4/\zeta\omega_n = 10$ seconds. From Figure 3.17 we can see that the green curve reaches about 0.98 at the 10th second.

From equation 3.3.15, the oscillation frequency of the unit step response for the second order system is damped natural frequency $\omega_d = \omega_n\sqrt{1-\zeta^2}$ instead of its natural frequency ω_n. The response convergence bounds are $\pm e^{-\zeta\omega_n t}$. Comparing these to the convergence response $e^{-t/T}$ of the first order system, where T is the time constant, we can see that letting coefficient $T = 1/\zeta\omega_n$ makes the convergence rates for the first order and second order systems comparable. Recall that for the first order system, the

system response settling times, $t_s(2\%)$ and $t_s(5\%)$, are $4T$ and $3T$, respectively. As a result, for the second order systems, system response settling times, $t_s(2\%)$ and $t_s(5\%)$, are $4/\zeta\omega_n$ and $3/\zeta\omega_n$, respectively.

Next, let's investigate the system peak time and percentage overshoot, starting with the peak time. Note that peak time candidates can be found by taking the time derivative of $y(t)$ defined in equation 3.3.15 and setting the derivative to zero as below.

$$\dot{y}(t) = K\left\{\zeta\omega_n \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}}\sin(\omega_d t+\varphi) \; - \omega_d \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}}\cos(\omega_d t+\varphi)\right\} = 0 \tag{3.3.17}$$

This equation leads to the following.

$$\zeta\sin(\omega_d t+\varphi) \; - \sqrt{1-\zeta^2}\cos(\omega_d t+\varphi) \; = 0 \tag{3.3.18}$$

Using the same definitions of $\sin\varphi = \sqrt{1-\zeta^2}$ and $\cos\varphi = \zeta$ as before, and utilizing the trigonometric identity, $\sin\varphi\cos(\omega_d t+\varphi) + \cos\varphi\sin(\omega_d t+\varphi) = \sin(\omega_d t+\varphi)$, we have the following.

$$\cos\varphi\sin(\omega_d t+\varphi) \; - \sin\varphi\cos(\omega_d t+\varphi) \; = \sin(\omega_d t+\varphi-\varphi) \; = \sin\omega_d t \; = 0 \tag{3.3.19}$$

It is clear that when $\dot{y}(t) = 0$, $t = n\pi/\omega_d$ $(n = 0, 1, 2, \ldots)$. From Figure 3.16 and Figure 3.17, the peak time can only occur at $t_{max} = \pi/\omega_d$. Substituting the peak time into equation 3.3.15 yields the following.

$$y_{max} = K\left\{1 - \frac{e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}}{\sqrt{1-\zeta^2}}\sin(\pi+\varphi)\right\} = K\left\{1 + \frac{e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}}{\sqrt{1-\zeta^2}}\sin\varphi\right\} = K\left\{1 + e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}\right\} \tag{3.3.20}$$

Therefore, the percentage overshoot (PO) can be calculated by the following equation.

$$PO(\%) = \frac{y_{max}-y_{ss}}{y_{ss}}100 = \frac{K\left\{1 + e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}\right\} - K}{K}100 = 100e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}} \tag{3.3.21}$$

Note that the derived percentage (PO) overshoot is a function of the second order system damping ratio ζ. That is, the system damping ratio is the only factor that affects the system PO. Now, we can summarize performance indices of the first and second order systems into Table 3.4.

Example 3.10

Consider the closed-loop system shown in Figure 3.18. Calculate y_{ss}, y_{max}, t_{max}, the percentage overshoot (PO), and 2% settling time for $K = 10,\ 25$.

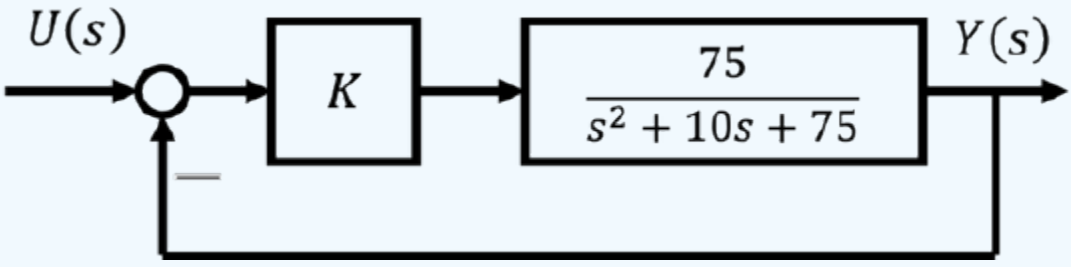


Figure 3.18: Second order system for Example 3.10.

Table 3.4: Summary of the first and second order system performance measures.

Performance indices	1st Order	2nd Order
Peak time	∞	$\frac{\pi}{\omega_d} = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}}$
Peak value	K	$K + Ke^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}$
Percent Overshoot	0	$100e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}$
Settling time (5%)	$3T$	$\frac{3}{\zeta\omega_n}$
Settling time (2%)	$4T$	$\frac{4}{\zeta\omega_n}$

First, we will calculate the closed-loop system transfer function $G(s)$ from $U(s)$ to $Y(s)$.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{75K}{s^2 + 10s + 75(K+1)} \tag{3.3.22}$$

For $K = 10$,

$$\omega_n = \sqrt{75 \times 11} = 28.72, \zeta = \frac{5}{\omega_n} = 0.174, \ \sqrt{1-\zeta^2} = 0.9847 \tag{3.3.23}$$

Now, we are ready to calculate these performance indices as below.

$$y_{ss} = G(0) = \frac{750}{75 \times 11} = 0.909, t_{max} = \frac{\pi}{28.72 \times 0.9847} = 0.111 \text{ (second)} \tag{3.3.24}$$

$$y_{max} = 0.909\left(1 + e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}\right) = 1.4316, PO(\%) = 100e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} = 57.4\% \tag{3.3.25}$$

$$t_s(2\%) = \frac{4}{\zeta\omega_n} = \frac{4}{0.174 \times 28.72} = 0.8 \text{ (second)}$$

For $K = 25$,

$$\omega_n = \sqrt{75 \times 26} = 44.16, \zeta = \frac{5}{\omega_n} = 0.113, \sqrt{1-\zeta^2} = 0.9936 \tag{3.3.26}$$

Then, we will calculate these performance indices as below.

$$y_{ss} = G(0) = \frac{75 \times 25}{75 \times 26} = 0.9615, t_{max} = \frac{\pi}{44.16 \times 0.9936} = 0.0716 \text{ (second)} \tag{3.3.27}$$

$$y_{max} = 0.9615\left(1 + e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}\right) = 1.634, PO(\%) = 100e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} = 69.96\% \tag{3.3.28}$$

$$(t_s(2\%) = \frac{4}{\zeta\omega_n} = \frac{4}{0.113 \times 44.16} = 0.8016 \text{ (second)}$$

Note that the increased settling time is due mainly to the reduced damping ratio.

Now, we are going to discuss how the closed-loop system pole locations affect system performance indices such as percentage overshoot (PO) and settling time t_s. Consider the second order system defined in equation 3.3.10. We can calculate its closed-loop poles by finding the pair of complex conjugate roots of the denominator's polynomial as follows.

$$s_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \tag{3.3.29}$$

Figure 3.19 shows the pole location geometry on the complex plane. Pole s_1 forms a right triangle with three sides of $\omega_n\sqrt{1-\zeta^2}$, $\zeta\omega_n$, and ω_n, where angle $\theta = \cos^{-1}\zeta$.

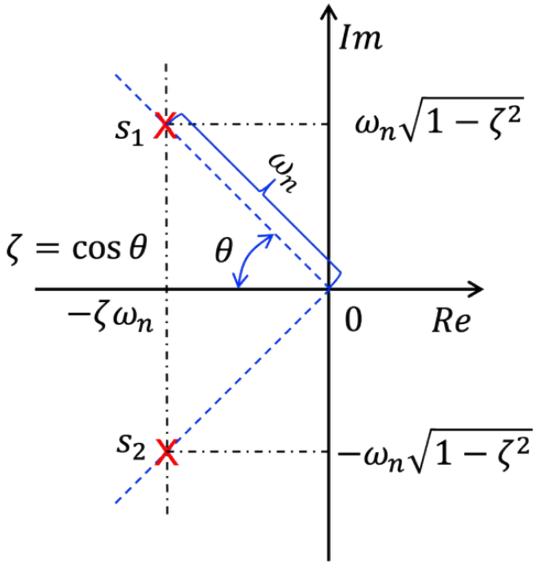


Figure 3.19: Pole location geometry on the complex plane for the second order system.

As discussed earlier in Table 3.4, the system percentage overshoot (PO) is only a function of damping ratio ζ (or angle θ defined in Figure 3.19). The system settling time is inversely proportional to $\zeta\omega_n$. Therefore, we could place (or assign) the closed-loop system pole locations, if possible, to guarantee system performances such as PO and settling time using a certain closed-loop controller. In Example 3.11, we demonstrate how to relate the performance indices to pole locations.

Example 3.11

Find the region of feasible pole locations for a closed-loop second order system with 5% settling time less than $t_{sm} = 3$ (second) and percentage overshoot (PO) less than $PO_{max} = 4.3\%$. Note that $t_s\,(5\%) = \dfrac{3}{\zeta\omega_n} \le t_{sm}$ and $PO = 100e^{-\pi\zeta/\sqrt{1-\zeta^2}}$ $= 100e^{-\pi/\tan\theta} \le PO_{max}$. In this case, we can have the following relationship to meet the performance requirements.

$$\zeta\omega_n \ge \frac{3}{t_{sm}} = 1, \tag{3.3.30}$$

$$\theta \le \tan^{-1}\left(\frac{-\pi}{\ln(0.01PO_{max})}\right) = \theta_{max} = 45^{\circ}$$

For both settling time and overshoot requirements of $t_s(5\%) \le 3$ and $PO \le 4.3\%$, we can generate feasible pole location maps for settling time and percentage overshoot (PO), as seen in Figure 3.20. The shaded areas are feasible regions of pole locations that satisfy settling time (left) and percentage overshoot (right) requirements, respectively.

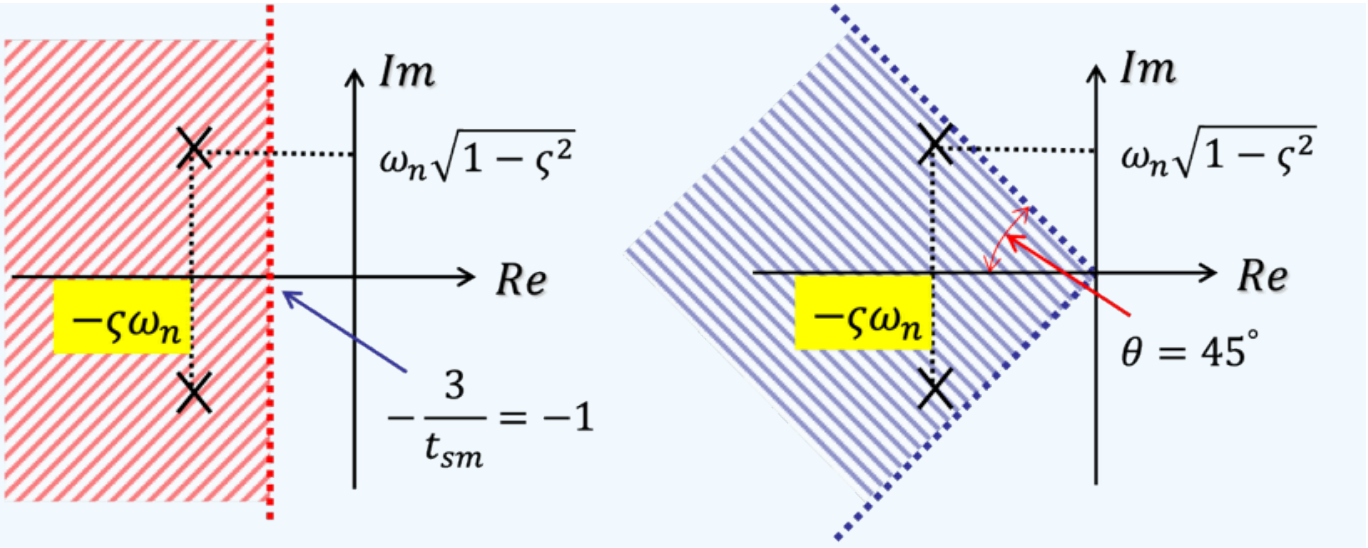


Figure 3.20: Feasible regions for given desired settling time and percentage overshoot.

Taking the intersection of the two shaded areas in Figure 3.20 leads to the combined feasible region for the closed-loop pole locations, as shown in Figure 3.21. That is, if the closed-loop system poles are located within the shaded area, the second order system response will have a 5% settling time less than 3 seconds and a percentage overshoot less than 4.3%.

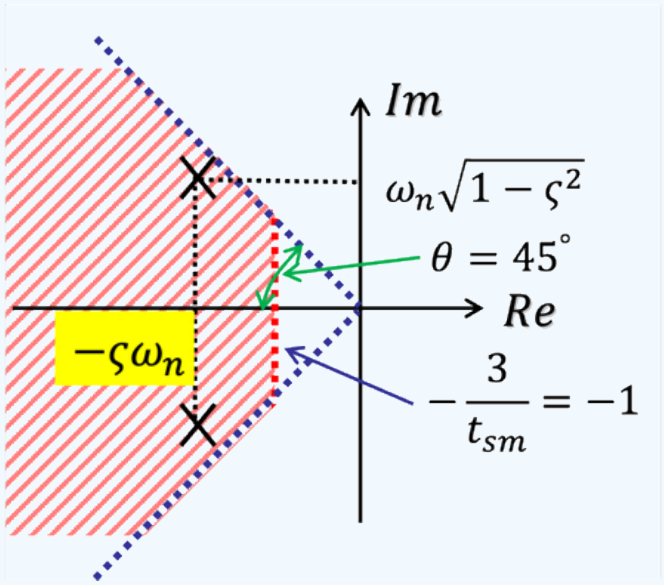


Figure 3.21: Combined feasible region for given desired settling time and percentage overshoot.

In short, we can see that the transient response of the second order system can be characterized by its damping ratio ζ and undamped natural frequency ω_n. In other words, pole locations are associated with roots of $s^2 + 2\zeta\omega_n s + \omega_n^2$. Delay time and rise time are not as easy to characterize and can be obtained only through simulation investigation. For transient responses of high order systems, normally the only option is computer simulation study. After we find the desired (feasible) pole location region, we can use pole assignment control to locate the closed-loop system poles inside the desired region (see Section 3.5).

Section Summary

This section analyzed system response characteristics for the first and second order linear systems with associated system response time, settling time, overshoot, etc. For first order systems, there is no response overshoot for the step input. Its settling time can be calculated analytically using the system time constant. For second order systems, the system response characteristic can also be calculated analytically. The main difference between first and second order systems is that a response overshoot exists for underdamped second order systems. Many high order system characteristics can be approximated by those of either

first or second order system. Thus, it is important to be familiar with the system response characteristic of these systems. We also examined the relationship between system pole locations and response characteristics; these results can also be used to guide control design and tuning.

Glossary

Practice Problems

1. To have good stability margin, you need to keep the closed-loop system percent overshoot (PO) below which of the following? Select the correct answer.
   a. 20%
   b. 30%
   c. 40%
   d. 50%
2. Plot the Bode diagram using the MATLAB command `bode` for the following loop transfer function. Find the gain crossover and phase crossover frequencies ω_g and ω_p, and then find the system gain margin and phase margin.

$$L(s) = C(s)G(s)H(s) = \frac{10(s+1)}{s} \cdot \frac{10(s+100)}{(s+10)^2} \cdot \frac{100}{s+1000}$$

3. For the following first order transfer function, find a) percentage overshoot (PO), b) steady-state response with unit step input, and c) 2% and 5% settling times.

$$G(s) = \frac{10}{20s+5}$$

4. The unit step response of a second order system is described by its transfer function below.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{a_2 s^2 + a_1 s + a_0}$$

   The system has the following properties:

$$y_{ss} = 2, \quad PO(\%) = 25\%, \quad t_s(2\%) = 25\%$$

   Determine the values of coefficients a_0, a_1, and a_2.
5. Consider the second order system transfer function shown below.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{10}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

   Find the feasible closed-loop pole locations such that the system 2% settling time $t_s(2\%)$ is less than 2 seconds and the percentage overshoot (PO) is less than 20%. Plot the feasible region on the complex plane.

This page titled 3.3: The First and Second Order System Responses and Associated Pole Locations is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

3.4: PID Control and Its Gain Tuning Methods

Learning Objectives

After completing this section, you will be able to:

1. Describe the structure of the PID (proportional-integral-derivative) controller.
2. Understand the system performance variation as a function of individual PID gain (proportional, integral, and derivative).
3. Tune these PID gains using the Ziegler-Nichols methods.
4. Tune these PID gains using the direct-tuning method.

This section presents PID (proportional-integral-derivative) control and its gain tuning approach. Closed-loop control strategies can be divided into two groups: model-free and model-based ones, where the PID control is model-free. As a result, its gains need to be tuned to optimize closed-loop system performance such as settling time and percentage overshoot. It is well-known that PID control gain tuning heavily relies on experience; however, only a few gain-tuning methods will be introduced in this section. PID controllers are widely used in many industries (such as chemical process, robotics, and automotives) due to their simplicity and relatively good closed-loop performance. We will introduce the structure of a PID controller first. See Figure 3.22 for its block diagram and equation 3.4.1 for its mathematical representation in the time domain, where the proportional, integral, and derivative gains are denoted by K_p, K_i, and K_d.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{3.4.1}$$

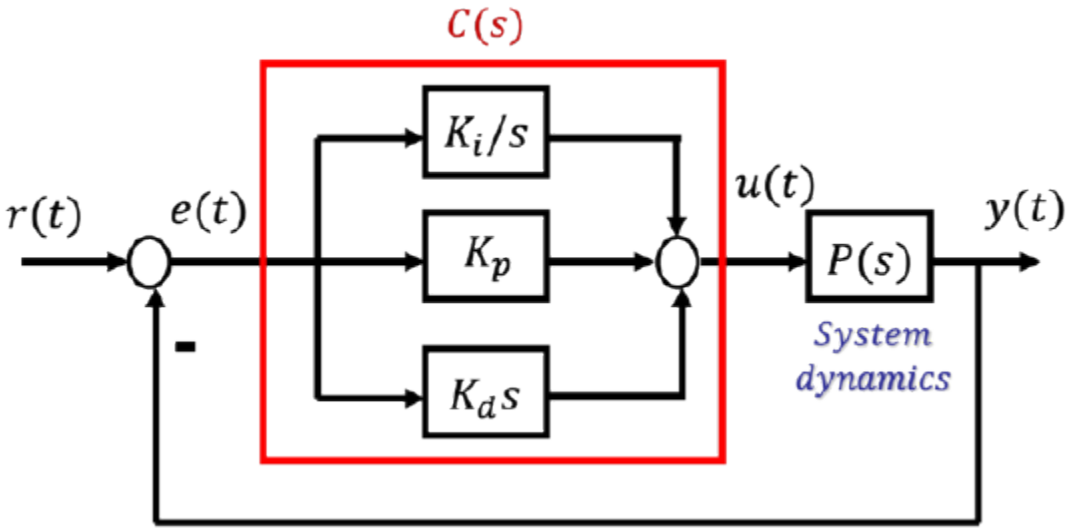


Figure 3.22: Continuous-time PID controller structure.

The associated PID controller transfer function is shown below.

$$C(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s = K_p \left(1 + \frac{K_i}{K_p s} + \frac{K_d s}{K_p}\right) \tag{3.4.2}$$

Normally, for any PID controller, the proportional gain will not be set to zero. In practical applications, to avoid amplification of the high frequency noise in $e(t)$ due to the differentiation operation, the derivative from equation 3.4.2 is often implemented as low-pass filtered differentiation, as below.

$$K_d s \approx \frac{K_d s}{\tau_d s + 1} \tag{3.4.3}$$

where τ_d is the time constant of the first order lowpass filter. Normally the value of τ_d can be selected below the system

sensor high frequency noise corner frequency $(1/\tau_d < \omega_{sensor})$ and above the bandwidth of system plane dynamics $(1/\tau_d > \omega_{system})$. Industrial applications have two other forms of PID controllers, called PI and PD controllers. The PI control strategy is formed by setting derivative gain to zero as below.

$$C(s) = K_p + \frac{K_i}{s} = K_p\left(1 + \frac{K_i}{K_p s}\right) \tag{3.4.4}$$

Note that the PI controller is often used for certain systems not requiring fast response but with zero steady-state error. These systems are very popular for chemical reaction process applications, where eliminating steady-state error is important. The PD controller is shown below with integral gain set to zero.

$$C(s) = K_p + K_d s = K_p\left(1 + \frac{K_d s}{K_p}\right) \tag{3.4.5}$$

Note that a PD controller is often used for systems requiring fast response but without a steady-state error requirement. These systems are very important for many mechatronic and robotic systems, where fast response is extremely important. Example 3.12 shows the effect of proportional, integral, and derivative gains.

Example 3.12

Consider the system diagram shown in Figure 3.22 with its plant transfer function shown below.

$$P(s) = \frac{1}{(s+1)^3} \tag{3.4.6}$$

Let's start by considering a proportional controller, $P(s) = K_p$, with reference signal $r(t) = u(t)$, a unit step input. From the simulated responses shown in Figure 3.23, we can observe two main characteristics. First, there exists steady-state error with steady-state response y_{ss} below the reference $r(t) = 1$. The steady state error reduces as the proportional control gain increases. Second, high proportional gain leads to fast response with reduced stability (increased overshoot and oscillations).

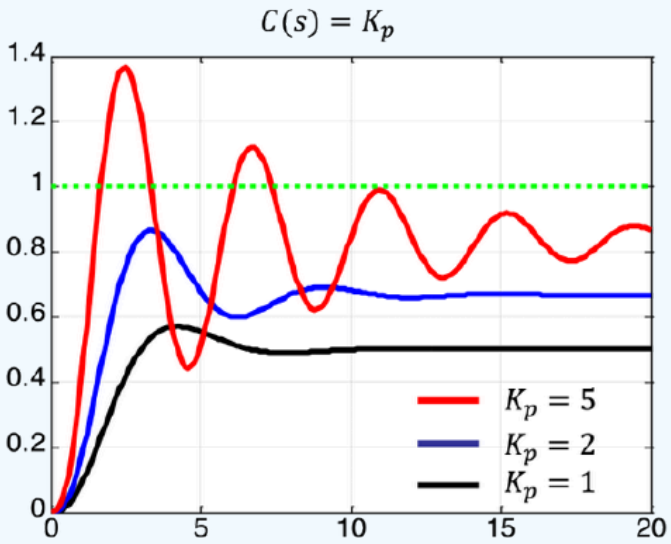


Figure 3.23: Step responses of proportional controllers.

Next, we consider a proportional-integral controller, $P(s) = K_p + K_i/s$, with unit step reference signal $r(t) = u(t)$ and $K_p = 1$. From the simulated responses shown in Figure 3.24, we can make two observations. First, the steady-state error is eliminated by nonzero integral gain when the system is stable. Second, high integral gain reduces response time with increased oscillations and overshoot (or reduced relative stability). Therefore, it is a tradeoff relationship: integral gain increases with improved response time, but with reduced relative stability (increased overshoot).

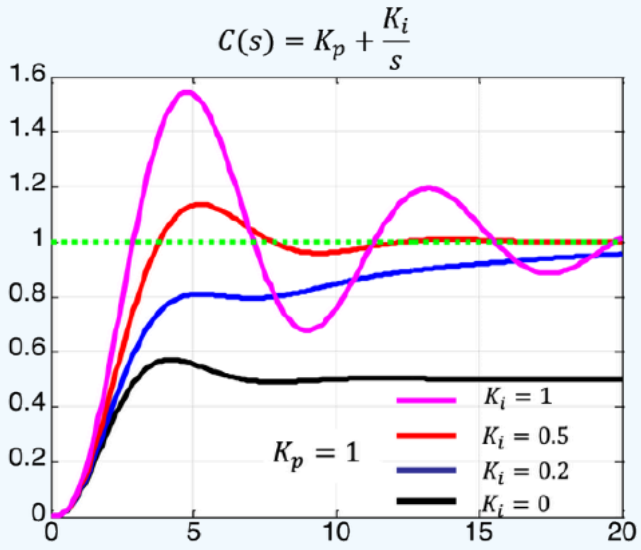


Figure 3.24: Step responses of proportional-integral (PI) controllers.

Lastly, we consider the proportional-integral-derivative (PID) controller, $P(s) = K_p + K_i/s + K_d s$, with step reference signal $r(t) = u(t)$, $K_p = 2.5$, and $K_i = 1.5$. From the simulated responses shown in Figure 3.25, we can make three observations. First, the steady state error is eliminated due to nonzero integral gain. Second, high derivative gain increases system damping. Third, however, extremely large derivative gain could make response performance worsen (or with reduced relative stability). As a result, increasing derivative gain improves response time, but large derivative gain could reduce relative stability (slow response).

From the study above, we can see that when $K_p = 2.5$, $K_i = 1.5$, and $K_d = 2.0$, the system response performance is reasonably good with the settling time about 2 seconds and percentage overshoot (PO) about 15%.

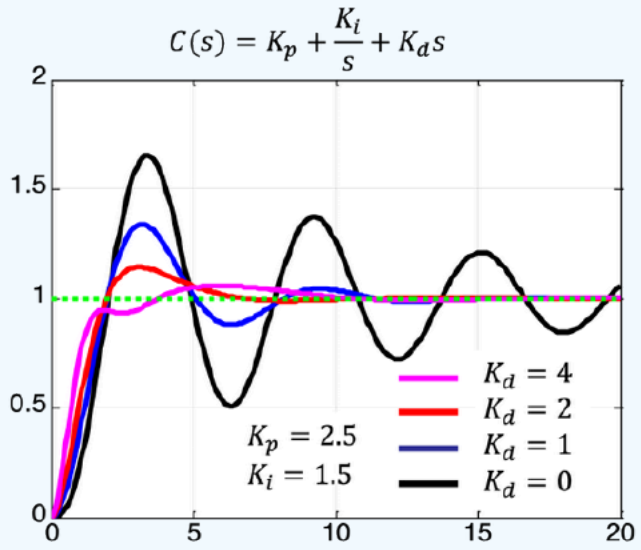


Figure 3.25: Step responses of proportional-integral-derivative (PID) controllers.

PID control gains can be tuned in two ways: model-based and model-free. One of the typical model-based tuning method is based on root locus analysis [1]. Note that a root locus plots the system's closed-loop pole locations as a function of selected gain from zero to infinity. As a result, PID gains can be tuned one-by-one [1]. On the other hand, PID gains can also be designed by model-based approaches [17]. In this book, we will mainly address the model-free PID gain tuning methods.

We start with the Ziegler-Nichols PID gain tuning method [3], introduced in 1942. In general, this method is suitable for stable

systems with any level of complexity when trial-and-error gain tuning is allowed. There are two approaches for Ziegler-Nichols PID gain tuning. The first approach is called “step response method” based on parameters, τ and α, defined in Figure 3.26, where the steepest tangent (red line) can be drawn based on the stable (converged) system step response. As a result, the “step response method” method is limited to stable systems because the step response will diverge for unstable systems.

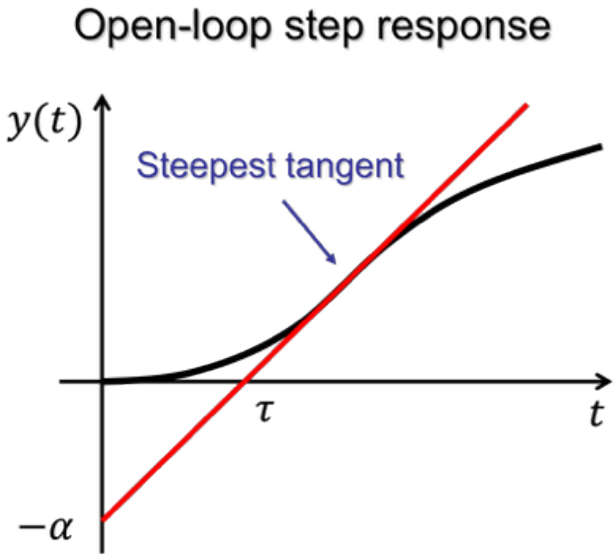


Figure 3.26: Zeigler-Nichols’ step response method.

Reorganize the PID controller into the following form, where $T_i = K_p/K_i$ and $T_d = K_d/K_p$.

$$C(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{3.4.7}$$

To tune the PID controller gains, find the open-loop system step response. Draw the steepest tangent line as shown in Figure 3.26, and find the two parameters, τ and α, marked on Figure 3.26. The Ziegler-Nichols step response method defines gains for proportional (P), proportional-integral (PI), and proportional-integral-derivative (PID) controllers, as seen in Table 3.5. When the controller migrates from proportional (P) controller to proportional-integral (PI) controller, the proportional gain K_p is reduced from $1/\alpha$ down to $0.9/\alpha$. This change indicates that adding integral control will reduce system relative stability and that proportional gain needs to be reduced. Also, when the controller migrates from a proportional-integral (PI) controller to a proportional-integral-derivative (PID) controller, the proportional gain K_p is increased from $0.9/\alpha$ up to $1.2/\alpha$, indicating that adding derivative control increases system relative stability.

Table 3.5: Ziegler-Nichols step response gain table.

Controller type	K_p	K_i	K_d
P	$1/\alpha$		
PI	$0.9/\alpha$	3τ	
PID	$1.2/\alpha$	2τ	0.5τ

The second approach, called the “ultimate sensitivity method” is based on parameters, K_c and T_c. It is defined in Figure 3.27. For the closed-loop system (shown in Figure 3.22) with a proportional controller, the ultimate sensitivity time response is obtained by gradually increasing the proportional gain until steady oscillation response is occurred. The two coefficients K_c and T_c are the proportional gain leading to steady oscillations and the oscillation period, respectively. This also means that applying the ultimate sensitivity method does not require the open-loop system to be stable. The only requirement is that the system can be stabilized by a proportional controller. Therefore, the ultimate sensitivity method has a broader application than the step response method.

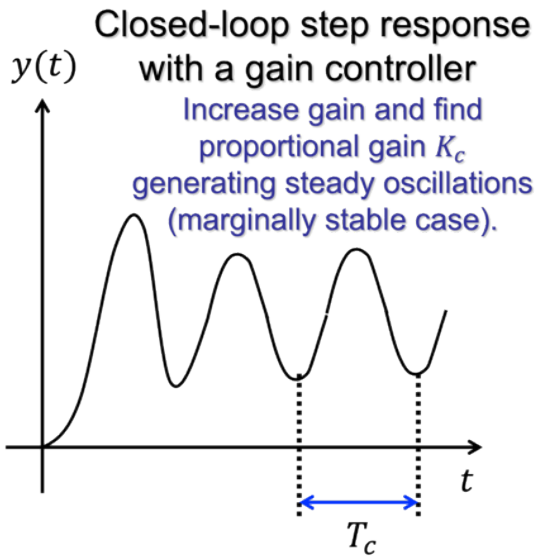


Figure 3.27: Ziegler-Nichols ultimate sensitivity method.

After the two parameters, K_c and T_c, are found, the Ziegler-Nichols ultimate sensitivity method defines gains for proportional (P), proportional-integral (PI), and proportional-integral-derivative (PID) controllers in Table 3.6. As with the step response method, when the controller migrates from proportional (P) controller to proportional-integral (PI) controller, the proportional gain K_p is reduced from $0.5K_c$ down to $0.4K_c$; this indicates that adding integral control will reduce system relative stability and that proportional gain needs to be reduced. Also, when the controller migrates from proportional-integral (PI) controller to proportional-integral-derivative (PID) controller, the proportional gain K_p is increased from $0.4K_c$ up to $0.6K_c$; this indicates that adding derivative control increases system relative stability. In addition, the integral gain is reduced from $0.8T_c$ down to $0.5T_c$. Below, we will present an example of applying the Ziegler-Nichols methods.

Table 3.6: Ziegler-Nichols ultimate sensitivity gain table.

Controller type	K_p	K_i	K_d
P	$0.5K_c$		
PI	$0.4K_c$	$0.8T_c$	
PID	$0.6K_c$	$0.5T_c$	$0.125T_c$

Example 3.13

Next, we will use the same plant model as that in Example 3.12. Figure 3.28 plots the step response and the steepest tangent line. The corresponding parameters are obtained as $\tau = 0.79$ and $\alpha = 0.28$. As a result, proportional, proportional-integral, and proportional-integral-derivative gains can be calculated using Table 3.5; they are listed in Table 3.7 along with well-tuned PID gains.

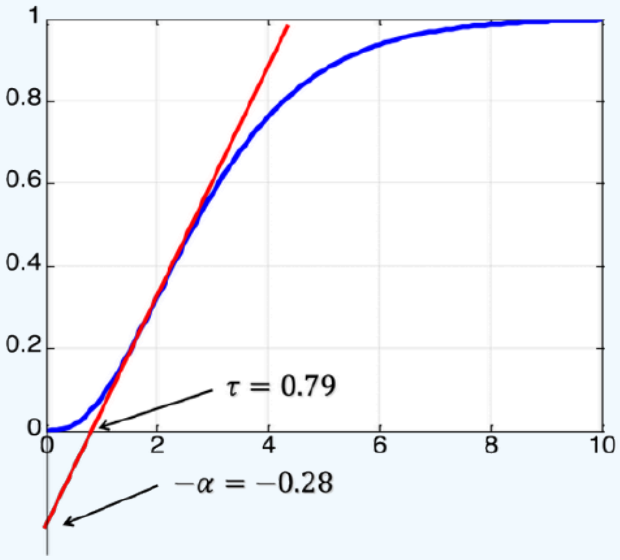


Figure 3.28: Ziegler-Nichols step response method for Example 3.13.

Table 3.7: Ziegler-Nichols step response gains for Example 3.13.

Controller type	K_p	K_i	K_d
P	3.57		
PI	3.21	2.37	
PID	4.29	1.58	0.395
Well-tuned	2.5	1.5	2.0

With the proportional, proportional-integral, and proportional-integral-derivative control gains calculated, we can calculate simulation studies. The associated results, which can be found in Figure 3.29, indicate that the time responses for P, PI, and PID controllers have a lot of oscillations. To be more specific, the overshoot of PI control is about 60% and that of PID control is about 48%.

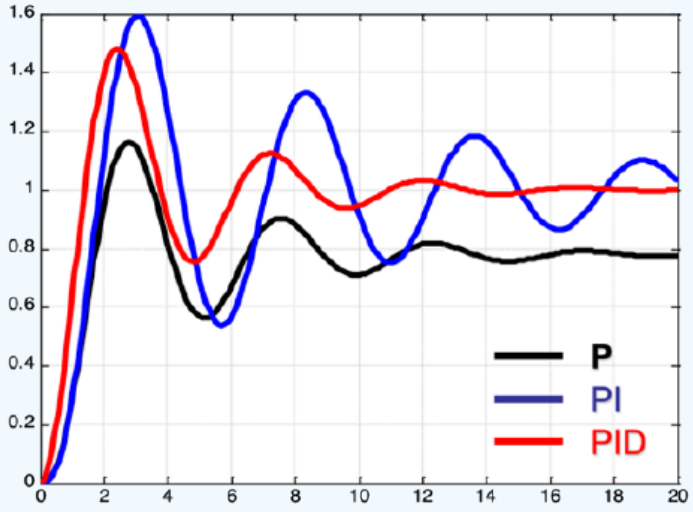


Figure 3.29: Step responses with gains from Ziegler-Nichols step response method.

Normally we would like to keep the percentage overshoot of closed-loop systems below 15%. Therefore, the relative stability of PI and PID controllers with gains from the step response method of Ziegler-Nichols is poor. We may need to use these gains as the initial gains and detune these gains to get satisfactory performance. The "well-tuned" gains in Table 3.7 are from Example 3.12, with its step response presented in Figure 3.25. Comparing these gains with thee gains from the step response method, the proportional gain is reduced, and the derivative gain is increased.

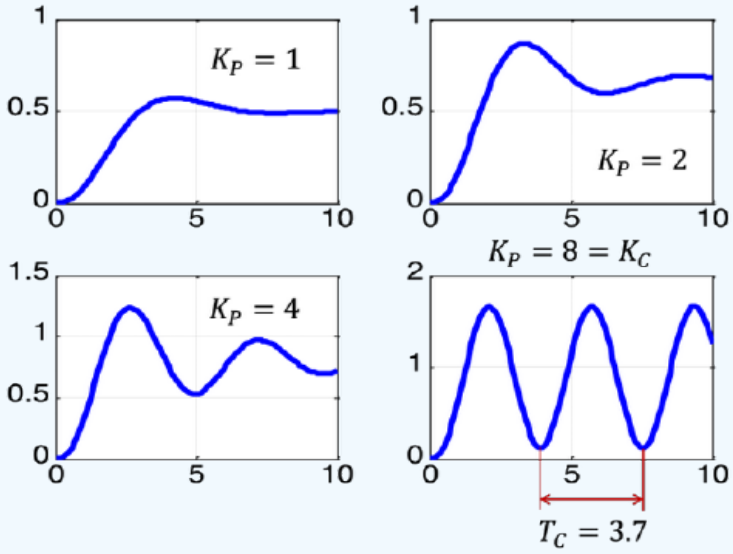


Figure 3.30: Ziegler-Nichols ultimate sensitivity method for Example 3.13.

Now let's look into the ultimate sensitivity method. Figure 3.30 plots the ultimate sensitivity step response with proportional gain increased from $K_p = 1$ to $K_p = 8$. When $K_p = 8$, the step response becomes marginally stable, oscillating with a constant magnitude and period. As a result, we can determine that $K_c = 8$ and $T_c = 3.7$. Proportional, proportional-integral, and proportional-integral-derivative gains can then be calculated. These gains are listed in Table 3.8 along with well-tuned PID gains.

Table 3.8: Ziegler-Nichols ultimate sensitivity gains for Example 3.13.

Controller type	K_p	K_i	K_d
P	4.0		
PI	3.2	2.96	
PID	4.8	1.85	0.463
Well-tuned	2.5	1.5	2.0

Simulation studies were conducted with the proportional (P), proportional-integral (PI), and proportional-integral-derivative (PID) control gains calculated in Table 3.8. Figure 3.31 shows the associated results. The time responses for P, PI, and PID have a lot of oscillations, but less than those from the step response method. To be more specific, the overshoot of PI control is about 52% whereas that of PID control is about 38%. Note that we would like to keep the percentage overshoot of closed-loop systems below 15%. Therefore, the relative stability of PI and PID controllers with gains from the ultimate sensitivity method of Ziegler-Nichols is poor. Again, we may use these gains as the initial gains, then decrease them to get satisfactory performance. In Table 3.8, the "well-tuned" gains are from Example 3.12 with its step response presented in Figure 3.25. Comparing these gains with the gains from the step response method, the proportional gain is reduced, and the derivative gain is increased significantly.

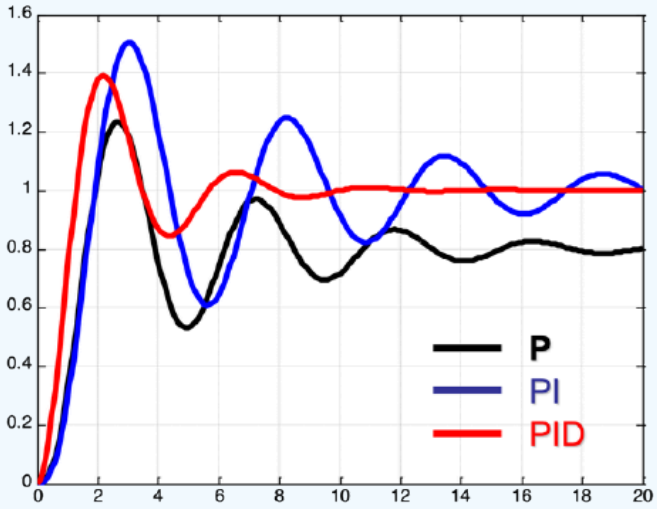


Figure 3.31: Step responses with gains from the Ziegler-Nichols ultimate sensitivity method.

Before finishing this section, let's look at a direct-tuning method for PID controllers, based on the author's industrial experience. The direct-tuning method is somewhat similar to the ultimate sensitivity method. It uses the following steps, and this method is suitable for those systems that can be stabilized by a proportional gain.

- Step 1: Start with all two gains (K_i and K_d) equal to zero, with K_P stabilizing the system.
- Step 2: Increase K_P graduately until the system is marginally stable (steady oscillations in output response observed). Then set K_P to 0.4 to 0.5 time of the corresponding valve. For simulation study, continuously increasing K_P could cause diverged oscillations; the K_P value making steady oscillation will be used to set K_p gain.
- Step 3: With the selected K_P, gradually increase gain K_i until the system is marginally stable (steady oscillations in output response observed). Then set K_i to 0.4 to 0.5 time of the corresponding valve. Similarly, for simulation study, continuously increasing K_i could cause diverged oscillations; the K_i value making steady oscillation will be used to set K_i gain.
- Step 4: Gradually increase the derivative gain for K_d to achieve the desired performance (only if required transient response time is not satisfactory).
- Step 5: As needed, repeat the process from Steps 2 to 4 with calibrated gains, or manually tune.

After these gains are tuned from the process stated above, fine-tuned gains can be used to finalize the PID control gain tuning. Note that increasing proportional (or integral) gain would improve the system response time, but it would also reduce the system relative stability (increase overshoot)—a tradeoff relationship. Meanwhile, increasing the derivative gain could improve system relative stability (reduce overshoot), but it could increase the system sensitivity to noise. In general, continuously increasing system gains will lead to system instability. Example 3.14 looks at using the direct-tuning method for gain tuning.

Example 3.14

We will use the same plant model as that in Examples 3.12 and 3.13. In Example 3.13, when studying the ultimate sensitivity method, we found that when $K_p = 8$ with $K_i = \ K_d = 0$, the step response oscillates with a fixed magnitude (Figure 3.30). Based on Step 2 of the direct-tuning method, the gain K_p is fixed at $0.4 \times 8 = 3.2$. Then in Step 3 of the direct-tuning method, we vary gain K_i with $K_p = 3.2$ and $\ K_d = 0$.

Figure 3.32 shows the system step response with $K_i = 1,\ 2,\ 2.25$, and 2.5. We can see that when $K_i = 2.25$, the step response (red line) oscillates with a constant magnitude. When $K_i = 2.5$, the step response (green line) oscillates with an increased magnitude. Therefore, integral gain K_i is selected as $K_i = 2.25 \times 0.4 = 0.9$ in Step 3 of the direct-tuning method.

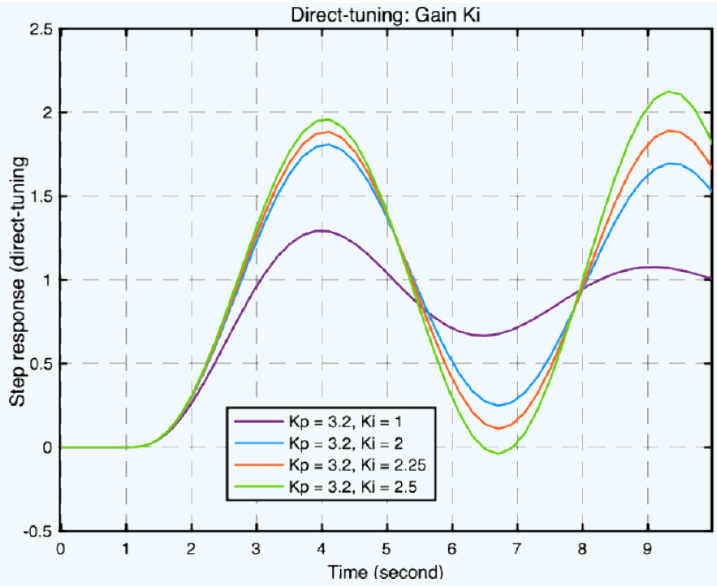


Figure 3.32: Step responses with $K_p = 3.2$, varying K_i, and $K_d = 0$ for the direct-tuning method.

Now, we proceed to Step 4. We will gradually increase the derivative gain to have desired performance with target overshoot to be less than 15%. With proportional gain $K_p = 3.2$ and integral gain $K_i = 0.9$, the derivative gain was set to $K_d = 0.5,\ 1.0,\ 1.5$, and 2.0. Figure 3.33 shows the associated step responses.

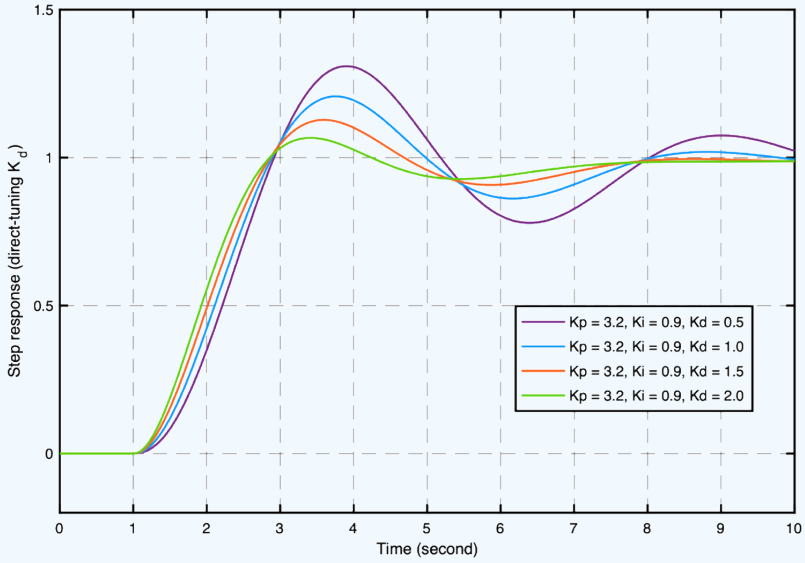


Figure 3.33: Step responses with $K_p = 3.2$, $K_i = 0.9$, and varying K_d for the direct-tuning method.

Table 3.9 presents calculated settling time and percentage overshoot. We can see that when $K_d = 1.5$, we have a satisfactory response performance with 5% settling time of around 6 seconds and percentage overshoot (PO) of 12.7% (below the 15% target). Therefore, the derivative gain is selected as $K_d = 1.5$.

Table 3.9: Direct-tuning settling time and percentage overshoot as a function of gain K_d.

Derivative gain K_d	0.5	1.0	1.5	2.0
5% settling time (s)	8.65	6.42	5.97	5.39
Overshoot (PO)	30.8%	20.7%	12.7%	6.7%

Table 3.10 compares the PID gains obtained earlier, where well-tuned gains are from Example 3.12, the step response and ultimate sensitivity gains are based on the Ziegler-Nichols method in Example 3.13, and the direct tuning gains are from this example (Example 3.14). From these examples, we found that both well-tuned and direct tuning gains provide good response performances. Although the Ziegler-Nichols methods provide relatively aggressive gains, they need to be detuned to get satisfactory performance. For the proportional gains, Ziegler-Nichols gains are large with direct tuning in the middle. The direct tuning integral gain is the smallest among all gains, and for the derivative gains, the Ziegler-Nichols method provides the lowest gains.

Table 3.10: PID gain comparison.

Controller type	K_p	K_i	K_d
Well-tuned	2.5	1.5	2.0
Step response	4.29	1.58	0.395
Ultimate sensitivity	4.8	1.85	0.463
Direct tuning	3.2	0.9	1.5

As a summery, PID control is the most popular control method used in industry since it is model-free with a simple control structure, easy to tune, and widely applicable. The Ziegler-Nichols method provides a starting-point for gain tuning, rather than final control gains in a single shot. The direct tuning method provides an easy way for PID gain tuning, but it may require certain control tuning experience. In addition, the gain set achieving desired performance target is not unique.

Section Summary

In practical applications, PID (proportional-integral-derivative) controllers are widely used due to their simplicity and easy gain tuning. A PID controller has only three control parameters (proportional, integral, and derivative gains), which makes controller gain tuning easy. We addressed the Ziegler-Nichols PID gain tuning rules in detail [3]. Normally the Ziegler-Nichols PID gains can be used as initial PID controller gains for gain tuning. We also presented an ad hoc (direct) PID control gain tuning method. In industrial applications, PID gains are often scheduled as a function of response error to improve closed-loop system performance. Section 3.6 will use examples to demonstrate how the gain scheduled PID control can improve system performance.

Section Keywords

Practice Problems

1. Tune PID control gains with Zeigler-Nichols methods (both step response and ultimate sensitivity methods) for the following plant model used in Examples 3.12, 3.13, and 3.14.

$$P(s) = \frac{1}{(s+1)^3}$$

2. Tune PID control gains with the direct tuning method for the following plant model used in Examples 3.12, 3.13, and 3.14.

$$P(s) = \frac{1}{(s+1)^3}$$

This page titled 3.4: PID Control and Its Gain Tuning Methods is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

3.5: Review of Model-Based Control Design and Gain Tuning Methods

Learning Objectives

After completing this section, you will be able to:

1. Design model-based controllers utilizing pole assignment, LQR (linear quadratic regulation), and LQG (linear quadratic Gaussian) methods.
2. Design OCC (output covariance constrained) controllers.
3. Describe model-based gain tuning method based on the OCC control design process.

This section reviews various model-based control design methods for mechatronic systems. These include pole assignment [18], linear quadratic regulation (LQR) [5], and linear quadratic Gaussian (LQG) methods [4], as well as output covariance constrained (OCC) control [6] design methods, where the OCC control methodology can be used for gain tuning for model-based control design method. Note that gain tuning is very important for model-based control because no other methodology exists that can be used directly for model-based control gain tuning (see Chapter 6 of reference [19] for details).

For model-based control, we use the following state-space system model

$$G(s): \begin{cases} \dot{x}_p = A_p x_p + B_p u \\ y_p = C_p x_p + H_p u \end{cases} \tag{3.5.1}$$

where $x_p \in \mathbb{R}^n$ is the system (plant) state vector with dimension n (also called system order), $u \in \mathbb{R}^{n_u}$ is the system control vector with dimension n_u, $y_p \in \mathbb{R}^{n_y}$ is the system output vector with dimension n_y, and A_p, B_p, C_p, and H_p are state-space system model matrices with proper dimensions. This state-space model can be also converted in a transfer function matrix as below.

$$G(s) = C_p(sI - A_p)^{-1}B_p + H_p \tag{3.5.2}$$

with $Y_p(s) = G(s)U(s)$, where $Y_p(s)$ and $U(s)$ are Laplace transforms of $y_p(t)$ and $u(s)$, respectively. A set of system model differential equations can be easily converted into a state-space model; Example 3.15 provides a demonstration.

Example 3.15

Consider Example 2.4 in Chapter 2 with the following differential equation.

$$\ddot{z}(t) + 3\dot{z}(t) + 2z(t) = f(t) \tag{3.5.3}$$

Letting $x = [x_1,\ x_2]^T = [z,\ \dot{z}]^T$ and $u = f$, we have the following.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -2x_1 - 3x_2 + u \end{aligned} \tag{3.5.4}$$

Based on these differential equations, the system described by equation 3.5.3 can be put into the form of equation 3.5.1 with the following system matrices

$$A_p = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}, B_p = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C_p = \begin{bmatrix} 1 & 0 \end{bmatrix}, H_p = 0 \tag{3.5.5}$$

where the transfer function from $F(s) = L[f(t)]$ to $Z(s) = L[z(t)]$ is shown below.

$$G(s) = \frac{Z(s)}{F(s)} = C_p(sI - A_p)^{-1}B_p + H_p = \frac{1}{s^2 + 3s + 2} \tag{3.5.6}$$

Before proceeding with model-based control, we need the following definitions for system controllability (stabilizability) and observability (detectability).

Definition 3.4

Consider a linear time-invariant (LTI) dynamic system, as defined in equation 3.5.1, or the pair $(A_p,\ B_p)$. The LTI system is said to be *controllable* if for any initial state $x_p(0) = x_0$, given the final state $x_1 = x_p(t_1)$, and for $t_1 > t_0$, there exists a piecewise continuous input $u(\cdot)$ such that the solution of the LTI system satisfies $x_p(t_1) = x_1$. Otherwise, the system is said to be *uncontrollable*.

An LTI dynamic system or the pair $(A_p,\ B_p)$ is said to be *stabilizable* if there exists a state feedback control, $u = Fx_p$, such that the feedback system is stable, That is, $A_p + B_pF$ is stable or $Re\left[eig(A_p + B_pF)\right] < 0$, where $eig(\cdot)$ denotes eigenvalues of a square matrix and $Re[\cdot]$ represents the real part of the complex number.

Theorem 3.5 [20]

For the system defined in equation 3.5.1, the following statements are equivalent.

1. $(A_p,\ B_p)$ is controllable.
2. The controllability matrix $\mathcal{C} = \left[B_p,\ A_pB_p,\ A_p^2B_p,\ \cdots,\ A_p^{n-1}B_p\right]$ has full row rank.
3. The matrix $[A_p - \lambda I,\ B_p]$ has full row rank for all λ (eigenvalues of A_p) in $\mathbb{C}$ (complex plane).

In addition, for the system defined in equation 3.5.1, the following statements are equivalent.

1. $(A_p,\ B_p)$ is stabilizable.
2. The matrix $[A_p - \lambda I,\ B_p]$ has full row rank for all $Re(\lambda) \geq 0$ (eigenvalues of A_p).
3. There exists a matrix F such that $A_p + B_pF$ is stable, i.e., $\mathrm{Re}\left(\lambda\left(A_p + B_pF\right)\right) < 0$.

Definition 3.5

An LTI dynamic system denoted by the pair $(C_p,\ A_p)$, defined in equation 3.5.1, is said to be *observable* if for any $t_1 > 0$, the initial state $x_p(0) = x_0$ can be determined from the history of the input $u(t)$ and the output $y(t)$ over the interval of $[0,\ t_1]$. Otherwise, the system is said to be *unobservable*.

An LTI dynamic system denoted by the pair $(C_p,\ A_p)$ is said to be *detectable* if there exists a matrix L such that $A_p - LC_p$ is stable. That is, $Re[eig\left(A_p - LC_p\right)] < 0$, where $eig(\cdot)$ denotes eigenvalues of a square matrix.

Theorem 3.6 [20]

For the system defined in equation 3.5.1, the following statements are equivalent.

1. $(C_p,\ A_p)$ is observable.
2. The observability matrix $\mathcal{O} = \left[C^T,\ A_p^TC_p^T,\ \left(A_p^T\right)^2C_p^T,\ \cdots,\ \left(A_p^T\right)^{n-1}C_p^T\right]^T$ has full row rank.
3. The matrix $\left[A_p^T - \lambda I,\ C_p^T\right]$ has full row rank for all λ (eigenvalues of A_p) in $\mathbb{C}$ (complex plane).

In addition, for the system defined in 3.5.1, the following statements are equivalent.

1. $(C_p,\ A_p)$ is detectable.
2. The matrix $\left[A_p^T - \lambda I,\ C_p^T\right]$ has full row rank for all $Re(\lambda) \geq 0$ (eigenvalues of A_p).
3. There exists a matrix L such that $A_p - LC_p$ is stable, i.e., $\mathrm{Re}\left(\lambda\left(A_p - LC_p\right)\right) < 0$.

Now, we are ready to discuss the pole assignment with state feedback. In Section 3.3, we discussed the relationship between system performance indices (e.g., overshoot and settling time) and the closed-loop system pole locations. If the system is stable,

a small $|Re[\lambda_i]|$ produces slow response convergence (λ_i is the ith eigenvalue of A_p). A large $|Re[\lambda_i]|$ leads to fast convergence. A real λ_i produces a monotonic response, whereas a pair of complex λ_i and λ_{i+1} result in an oscillatory response. In addition, for a pair of complex λ_i and λ_{i+1}, the response becomes more oscillatory as the ratio $|Re[\lambda_i]/Im[\lambda_i]|$ decreases. This last can also be expressed as $\tan^{-1}(|Re[\lambda_i(A_p)]/Im[\lambda_i(A_p)]|)$. Therefore, in order to achieve required system performance, there is a motivation to assign the closed-loop poles to desired locations using state feedback control.

Consider the state feedback control $u = Fx + v$, where F is $n_u \times n$ is called state feedback gain and $v(t)$ is an n_u dimensional vector. The resulting closed-loop system of 3.5.1 is shown below.

$$G_{CL}(s): \quad \{ \quad \dot{x}_p = (A_p + B_pF)x_p + B_pv \quad y_p = (C_p + H_pF)x_p + H_pv \tag{3.5.7}$$

The stability and transient response of the closed-loop system are determined by the eigenvalues of $(A_p + B_pF)$. Theorem 3.7 states the condition for relocating these open-loop poles to the desired locations. The uncontrollable eigenvalues λ_i of $(A_p,\ B_p)$, which makes matrix $[A_p - \lambda_i I,\ B_p]$ no longer full row rank, cannot be relocated by state feedback control. That is, if you have any uncontrollable unstable eigenvalue λ_i of $(A_p,\ B_p)$, the closed-loop system cannot be stabilized by a state feedback control.

Theorem 3.7 [18]

For the dynamic system defined in equation 3.5.1, let $(A_p,\ B_p)$ be controllable and λ_1, λ_2, ..., λ_n be a given set of complex numbers with complex numbers being conjugate pairs. Then, there exists a real $n_u \times n$ matrix F such that eigenvalues of $(A_p + B_pF)$ are λ_1, λ_2, ..., λ_n.

For single-input and single-output (SISO) systems, the system transfer function can be expressed by the following transfer function.

$$G(s) = \frac{b_{n-1}s^{n-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} + d \tag{3.5.8}$$

The associated state-space realization results in the following system matrices.

$$A_p = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix}, B_p = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, C_p^T = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix}, H_p = d \tag{3.5.9}$$

For a given set of desired poles λ_1, λ_2, ..., λ_n, we can define $q(s)$ as below.

$$q(s) = \prod_{i=1}^{n}(s - \lambda_i) = s^n + \overline{a}_{n-1}s^{n-1} + \cdots + \overline{a}_1 s + \overline{a}_0 \tag{3.5.10}$$

Then, the following state feedback gain F results in the closed-loop system poles of λ_1, λ_2, ..., λ_n.

$$F = \begin{bmatrix} a_0 - \overline{a}_0 & a_1 - \overline{a}_1 & a_2 - \overline{a}_2 & \cdots & a_{n-1} - \overline{a}_{n-1} \end{bmatrix} \tag{3.5.11}$$

Note that

$$A = A_p + B_pF = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ -\overline{a}_0 & -\overline{a}_1 & -\overline{a}_2 & \cdots & -\overline{a}_{n-1} \end{bmatrix} \tag{3.5.12}$$

Example 3.16

Consider the system defined in equation 3.5.1 with system matrices A_p and B_p shown below.

$$A_p = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, B_p = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.5.13}$$

The system characteristic polynomial is $\det(sI - A_p) = s^3 - s^2 + 0s - 1$ with eigenvalues of A_p as $s_1 = 1.4656$ and $s_{2,3} = -0.2328 \pm 0.7926j$. The open-loop system is not stable because $s_1 > 0$. We would like to design a state feedback gain F to assign the eigenvalues of $(A_p + B_pF)$ to $\lambda_1 = -1$ and $\lambda_{2,3} = -1 \pm j$ so that the closed-loop system is stable. Based on the derivation above, the desired characteristic polynomial is shown below.

$$q(s) = (s+1)(s^2 + 2s + 2) = s^3 + 3s^2 + 4s + 2 \tag{3.5.14}$$

Therefore, the feedback gain assigning eigenvalues of $(A_p + B_pF)$ to $\lambda_1 = -1$ and $\lambda_{2,3} = -1 \pm j$ can be calculated as below.

$$F = \begin{bmatrix} -1-2 & 0-4 & -1-3 \end{bmatrix} = \begin{bmatrix} -3 & -4 & -4 \end{bmatrix} \tag{3.5.15}$$

For multi-input and multi-output (MIMO) systems, the formulae for calculating the feedback gain F can be found in [18]; the solution may not be unique. Consider MATLAB [21] command F=place(Ap,Bp,p), where $Ap = A_p$ and $Bp = B_p$. It can be used to calculate the state feedback gain, where matrices A_p and B_p are system matrices, and $p = \begin{bmatrix} \lambda_1, & \lambda_2 & \cdots & \lambda_n \end{bmatrix}^T$ is a vector of poles to be assigned.

In cases where not all system states are available for feedback control, a state observer can be designed if the pair $(C_p,\ A_p)$ is detectable, and the estimated states can be used in place of measured states. The state observer [22] is in the following form

$$\begin{aligned} \dot{\hat{x}} &= A_p\hat{x} + B_pu + L(y_p - \hat{y}) \\ \hat{y} &= C_p\hat{x} + H_pu \end{aligned} \tag{3.5.16}$$

where $\hat{x}$ is the estimated state vector, $\hat{y}$ is estimated system output, and L is the estimation gain matrix. To make sure the estimated state vector $\hat{x}$ converges to the original state vector x, the matrix $(A_p - LC_p)$ will be stable, i.e., all eigenvalues of $(A_p - LC_p)$ have negative real parts or $Re[A_p - LC_p] < 0$. Note that the eigenvalues of $(A_p - LC_p)$ can be freely assigned if (C_p, A_p) is observable. The problem of assigning observer eigenvalues of $(A_p - LC_p)$ can be converted to a pole assignment problem. Subtracting the estimated state equation in 3.5.16 by the state equation in 3.5.1 leads to the following.

$$\dot{e} = \dot{x}_p - \dot{\hat{x}} = (A_p - LC_p)(x - \hat{x}) = (A_p - LC_p)e \tag{3.5.17}$$

Because $(A_p - LC_p)$ is stable, as time goes to infinity, the error vector e converges to zero. Now, let $u = F\hat{x} + v$ and $x_{cl} = \begin{bmatrix} x^T, & e^T \end{bmatrix}^T$. The resulting closed-loop system of 3.5.1 with state estimator defined in equation 3.5.16 can be reorganized into the following form.

$$\dot{x} = \begin{bmatrix} \dot{x}_p \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A_p + B_pF & -B_pF \\ 0 & A_p - LC_p \end{bmatrix} \begin{bmatrix} x_p \\ e \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} v = Ax + Bv \tag{3.5.18}$$

The closed-loop system is stable because eigenvalues of the closed-loop system matrix A_{cl} are those from matrices $A_p + B_pF$ and $A_p - LC_p$ that are stable. Therefore, as long as the eigenvalues of $A_p + B_pF$ and $A_p - LC_p$ are assigned to be stable, the closed-loop system based on the estimated state vector is asymptotically stable.

During the state estimator design, we normally want the convergence rate of estimated state vector (associated with poles of $A_p - LC_p$) to be much faster than the convergence rate of closed-loop system dynamics (associated with poles of $A_p + B_pF$). In other words, it is recommended that the minimal absolute value of real parts of the assigned estimator poles shall be at least a few times larger than the maximum absolute value of real parts of the assigned closed-loop system poles. In this case, the state estimation error will have minimal effect on the closed-loop system response, compared with the case using state feedback.

In the LQG (linear quadratic Gaussian) control to be discussed later, the estimation gain L will be optimized based on intensity matrices of system and measurement noises. For a given level of system noise, the lower the level of measurement noise (in

terms of maximum singular value of the measurement noise intensity matrix), the larger the optimized estimation gain L in term of its maximum singular value, and vice versa. Note that large estimation gain L is able to move the estimator poles to further left on the complex plane, resulting in fast estimation convergence.

In addition to the pole assignment method based on the system state-space model, linear optimal control techniques are widely used for design optimal controllers. These include linear quadratic regulation (LQR or LQ), linear quadratic Gaussian (LQG), output covariance constraint (OCC), and linear parameter-varying (LPV) control. Note that LPV control is not addressed in this book; see references [9], [10], [11], [12], [13] for more details.

LQR Control Problem

Let's start with LQR control. Consider the following system

$$\dot{x}_p = A_p x_p + B_p u, x_p(0) = x_0 \tag{3.5.19}$$

where system state vector x_p, control input vector u, and system matrices A_p and B_p are defined earlier. In addition, x_0 is an arbitrary initial system state vector. Note that x_0 is deterministic but can be any given vector with compatible dimension.

Assuming that the pair $(A,\ B)$ is stabilizable, find an optimal controller, $u(t) = Fx_p(t)$, that stabilizes the system defined in equation 3.5.19 and minimizes the following cost function

$$J_{LQR} = \int_0^\infty \left\{x_p^T(t) Q x_p(t) + u^T(t) R u(t)\right\} dt \tag{3.5.20}$$

for any given initial system state vector $x_p(0) = x_0$ with $R > 0$ (positive definite) and $Q = C^T C \geq 0$ (positive semidefinite).

From references [5], [6], and [20], the global optimal solution exists if $(A_p,\ B_p)$ is stabilizable and $(C,\ A_p)$ is detectable, and the resulting optimal state feedback control is in the following form

$$u(t) = Fx_p(t), F = -R^{-1}B_p^T K \tag{3.5.21}$$

where $K \geq 0$ is the positive semidefinite solution of the following Riccati equation.

$$0 = A_p^T K + K A_p - K B_p R^{-1} B_p^T K + Q \tag{3.5.22}$$

Note that the above Riccati equation has a positive semidefinite solution if $(A_p,\ B_p)$ is stabilizable and $(C,\ A_p)$ detectable; see [6] and [20] for details.

Remark 3.5

The optimal control law defined in equation 3.5.21 is global optimal for any given initial state vector. In MATLAB, the optimal LQR control can be designed using the command F=-lqr(Ap,Bp,Q,R,0), where $Ap = A_p$ and $Bp = B_p$, by providing required system and weighting matrices. The requirement of $(C,\ A_p)$ detectable is a sufficient condition; it can be reduced down to the condition that $(C,\ A_p)$ has no unobservable mode on the imaginary axis [20]. Although the LQR controller is global optimal, it is optimal with respect to the cost function (equation 3.93) with given weighting matrices $R > 0$ and $Q \geq 0$. How to select the weighting matrices Q and R to achieve desired closed-loop system performance is an open question. The output covariance constraint (OCC) control design generates an optimal $Q \geq 0$ by converting the problem into a constrained optimization problem [6].

Example 3.17

Consider the system defined in equation 3.5.19 with system matrices A_p and B_p shown below, and with output weighting matrix $Q = 10I_3$ and control weighting $R = 1$, where I_3 is an identity matrix with dimension of three.

$$A_p = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, B_p = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.5.23}$$

An LQR controller can be designed using the MATLAB command F=-lqr(Ap,Bp,Q,R,0), where ($Ap = A_p$ and $Bp = B_p$) with state feedback control gain listed below.

$$F = \begin{bmatrix} -4.3166 & -6.3473 & -5.8677 \end{bmatrix} \tag{3.5.24}$$

The resulting closed-loop system poles of $(A_p + B_pF)$ are $\lambda_1 = -3.2138$ and $\lambda_{2,3} = -0.827 \pm 0.59j$.

Next, we will discuss the LQG control problem, where the following system model is considered with notations used in [6].

$$\begin{cases} \dot{x}_p = A_p x_p + B_p u + D_p w_p \\ y_p = C_p x_p \\ z = M_p x_p + v. \end{cases} \tag{3.5.25}$$

where system state vector x_p, control input vector u, and system matrices A_p and B_p are defined in equation 3.5.1. In addition, w_p and v are system and measurement noises, respectively; D_p, C_p, and M_p are system noise input, performance output, and measurement output matrices, respectively; and $z \in \mathbb{R}^{n_z}$ is the system measurement vector.

Note that both w_p and v are uncorrelated zero-mean white noises with intensity matrices $W_p > 0$ and $V > 0$. That is, letting $\mathcal{E}$ be an expectation operator, we have the following.

$$\mathcal{E}\left[w_p(t)\right] = 0, \quad \mathcal{E}\left[w_p(t) w_p^T(t-\tau)\right] = W_p \delta(\tau) \; \mathcal{E}\left[v(t)\right] = 0, \quad \mathcal{E}\left[v(t) v^T(t-\tau)\right] = V\delta(\tau) \tag{3.5.26}$$

For a strictly proper output feedback dynamic stabilizing control law given below,

$$\begin{cases} \dot{x}_c = A_c x_c + B_c z \\ u = C_c x_c \end{cases} \tag{3.5.27}$$

The resulting closed-loop system is

$$\begin{cases} \dot{x} = Ax + Dw \\ y = \begin{bmatrix} y_p \\ u \end{bmatrix} = \begin{bmatrix} C_y \\ C_u \end{bmatrix} x = Cx \end{cases} \tag{3.5.28}$$

where $x = \begin{bmatrix} x_p^T & x_c^T \end{bmatrix}^T$ and $w = \begin{bmatrix} w_p^T & v^T \end{bmatrix}^T$ and system matrices A, D, and C are defined below.

$$A = \begin{bmatrix} A_p & B_p C_c \\ B_c M_p & A_c \end{bmatrix}, \quad D = \begin{bmatrix} D_p & 0 \\ 0 & B_c \end{bmatrix}, \quad C = \begin{bmatrix} C_y \\ C_u \end{bmatrix} = \begin{bmatrix} C_p & 0 \\ 0 & C_c \end{bmatrix} \tag{3.5.29}$$

For the closed-loop system defined in equation 3.5.28, let $W = block\ diag\left[W_p,\ V\right]$ be the closed-loop covariance matrix. Then the closed-loop system state covariance matrix X can be defined by the positive semidefinite solution of the following Lyapunov equation.

$$0 = AX + XA^T + DWD^T \tag{3.5.30}$$

Note that X is also the controllability Gramian of the pair $(A,\ DW^{0.5})$. Let $\mathcal{E}_\infty[\,\cdot\,] \triangleq \lim_{t\to\infty} \mathcal{E}[\cdot]$ and we are ready to define the LQG control problem below.

LQG Control Problem

Assuming that the pair $(A_p,\ B_p)$ is stabilizable and (M_p, A_p) is detectable, find an optimal full-order dynamic controller in the form defined in equation 3.5.27 that stabilizes the closed-loop system defined in equation 3.5.28 and minimizes the following cost function

$$J_{LQG} = \mathcal{E}_\infty \left[y_p^T(t) Q y_p(t-\tau) \right] + \mathcal{E}_\infty \left[u^T(t) R u(t-\tau) \right] = trace\left(Q C_y X C_y^T + R C_u X C_u^T \right) \tag{3.5.31}$$

for zero mean white noise input with covariance matrix $W = block\ diag\,[W_p,\ V]$, where $X \geq 0$ is defined in equation 3.5.30, $R > 0$ (positive definite), and $C_p^T Q C_p \geq 0$ (positive semidefinite).

From references [5], [6], and [20], the global optimal solution exists if $(A_p,\ B_p)$ is stabilizable and (M_p, A_p) is detectable, and the optimal full order dynamic controller is in the form of 3.5.27, where

$$A_c = A_p + B_p F - L M_p, B_c = L = \tilde{X} M_p^T V^{-1}, C_c = F = -R^{-1} B^T K \tag{3.5.32}$$

where $\tilde{X} \geq 0$ and $K \geq 0$ are the positive semidefinite solutions of the following Riccati equations.

$$\begin{aligned} 0 &= A_p \tilde{X} + \tilde{X} A_p^T - \tilde{X} M_p^T V^{-1} M_p \tilde{X} + D_p W_p D_p^T \\ 0 &= A_p^T K + K A_p - K B_p R^{-1} B_p^T K + C_p^T Q C_p \end{aligned} \tag{3.5.33}$$

Note that these Riccati equations have a positive semidefinite solution $\tilde{X} \geq 0$ if $(M_p,\ A_p)$ is detectable and $(A_p,\ D_p)$ is stabilizable. They have a positive semidefinite solution $K \geq 0$ if $(A_p,\ B_p)$ is stabilizable and $(Q^{0.5} C_p,\ A_p)$ is detectable. The derivation can be found in references [6] and [20] for a generalized case. Note that $\tilde{X} \geq 0$ is also called the state estimation error covariance matrix.

Remark 3.6

Note that the optimal control cost defined in equation 3.5.31 is the summation of weighted closed-loop system output variance and weighted control variance. The optimal control law defined in equation 3.5.32 is global optimal.

Remark 3.7te

In MATLAB, the optimal LQG control gains F and L in equation 3.5.32 can be designed using MATLAB command lqr by providing required system and weighting matrices. Examples include F=-lqr(Ap,Bp,Qp,R,0)($Ap = A_p$, $Bp = B_p$, and $Qp = C_p^T Q C_p$) and Lt=-lqr(Apt,Bpt,Wp,V,0 (Ap=A_p,\) $Bp = B_p$, $Wp = D_p W_p D_p^T$, $Lt = L^T$). Similar to the LQR case, the requirement of $(Q^{0.5} C_p,\ A_p)$ detectable is a sufficient condition. It can be reduced down to the condition that $(Q^{0.5} C,\ A_p)$ has no unobservable mode on the imaginary axis (see [20] for details). Similarly, the requirement of $(A_p,\ D_p)$ stabilizable is a sufficient condition; it can be relaxed to the condition that $(A_p,\ D_p)$ has no unstabilizable mode on the imaginary axis.

Example 3.18

Consider the system defined in equation 3.5.25 with system matrices A_p, B_p, D_p, C_p, and M_p. These are shown below with output weighting matrix $Q = 10 I_2$, control weighting $R = 1$, system input noise intensity matrix $W_p = 1$, and measurement noise intensity matrix $V = 0.001 I_2$, where I_2 is an identity matrix with dimension of two.

$$A_p = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, B_p = D_p = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},\ C_p^T = M_p^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{3.5.34}$$

An LQG controller can be designed using the MATLAB commands F=-lqr(Ap,Bp,Qp,R,0)($Ap = A_p$, $Bp = B_p$, and $Qp = C_p^T Q C_p$) and Lt=-lqr(Apt,Bpt,Wp,V,0 (Ap=A_p,\) $Bp = B_p$, $Wp = D_p W_p D_p^T Q$, $Lt = L^T$) with dynamic feedback control gain F and the estimation gain L listed below.

$$\begin{aligned} F &= -\begin{bmatrix} 4.3166 & 5.5794 & 4.4869 \end{bmatrix} \\ L^T &= \begin{bmatrix} 0.9997 & 1.0246 & 1.2449 \\ 1.0246 & 8.9503 & 40.5793 \end{bmatrix} \end{aligned} \tag{3.5.35}$$

The resulting closed-loop system poles are the combination of these from $(A_p + B_pF)$ ($\lambda_1 = -1.1441$ and $\lambda_{2,3} = -1.1714 \pm 1.2356j$) and these from $(A_p - LM_p)$ ($\lambda_4 = -1.0015$ and $\lambda_{5,6} = -3.9743 \pm 3.9745j$).

One of the key barriers of applying advanced control theories (such as LQR and LQG) to solve practical control problems is that no systematic approach exists for tuning model-based control gains to optimize the system performance. Difference (or modeling error) always exists between the control design model and the physical system. Due to the modeling error, extremely high gain controllers may destabilize the closed-loop physical system [7]. Therefore, it is important to have a method to design a series of model-based controllers with their control gains varying from low to high. In that way, we can start testing with the low gain controller and gradually increase the control gain experimentally until the closed-loop system performance is no long improving. This result would indicate that the best closed-loop system has been achieved; see reference [7] for an application.

To further improving the closed-loop system performance, the control design model needs to be further improved to reduce modeling error. This process of model-based control turning is motivated to form the following control problem called an output covariance constraint (OCC) control problem (introduced in reference [6] along with a guaranteed convergence control design algorithm). Before presenting the OCC control problem, partition the performance output y_p from equations 3.5.25 and 3.5.28 into $y_p \cong \left[y_1^T,\ y_2^T,\ \cdots, y_m^T\right]^T$, where $y_i = C_i x = C_{p,i} x_p \in \mathbb{R}^{m_i}$ ($C^T = \left[C_1^T,\ C_2^T, \ldots, C_m^T\right]$ and $C_p^T = \left[C_{p,1}^T,\ C_{p,2}^T, \ldots, C_{p,m}^T\right]$) for $i = 1, 2,\ \ldots,\ m$ ($\sum_{i=1}^{m} m_i = n_y$).

OCC Control Problem [6]

Assuming that the pair $(A_p,\ B_p)$ is stabilizable and (M_p, A_p) is detectable. Find an optimal full-order dynamic controller in the form defined in equation 3.5.27 that stabilizes the closed-loop system defined in equation 3.5.28 and minimizes the following cost function

$$J_{OCC} = \left[u^T(t)\, Ru(t-\tau)\right] = \operatorname{trace}\left(RC_u X C_u^T\right) \tag{3.5.36}$$

subject to output performance constraints

$$\mathcal{E}_\infty\left[y_i(t) y_i^T(t-\tau)\right] = C_i X C_i^T \le \overline{Y}_i, i = 1, 2, \ldots, m \tag{3.5.37}$$

for zero mean white noise input with intensity matrix $W = block\ diag\left[W_p,\ V\right]$, where $X \ge 0$ is defined in equation 3.5.30, $R > 0$ (positive definite), and $\overline{Y}_i > 0\ \ (i = 1, 2, \ldots, m)$ (positive definite) are constraint matrices on the output covariance matrices.

Remark 3.8

Reference [6] indicates that the solution of the OCC control problem stated above is an LQG controller with a special choice of output weighting matrix $Q = block\ diag[Q_1,\ Q_{2,\ \ldots,}\ Q_m] \ge 0$, $Q_i \in \mathbb{R}^{m_i \times m_i} (i = 1,\ 2,\ \ldots,\ m)$ such that the following conditions are satisfied.

$$0 = \left(Y_i - \overline{Y}_i\right) Q_i = \left(C_i X C_i^T - \overline{Y}_i\right) Q_i, Y_i \le \overline{Y}_i,\ (i = 1,\ 2,\ \ldots,\ m) \tag{3.5.38}$$

The proof of this theory can be found in reference [6].

An Iterative OCC Control Design Algorithm [6]

Now, the question is how to find the weighting matrix Q satisfying conditions stated in equation 3.5.38 in the form of $= block\ diag[Q_1,\ Q_{2,\ \ldots,}\ Q_m] \ge 0$, $Q_i \in \mathbb{R}^{m_i \times m_i}\ (i = 1,\ 2,\ \ldots,\ m)$. Fortunately reference [6] provides an iterative algorithm to find such a weighting matrix Q with guaranteed convergence if the optimal OCC solution exists; see reference [6] for details.

1. Assume this system has matrices $A_p,\ B_p,\ D_p, C_p$ and M_p. It has system and measurement noise intensity matrices W_p and V and control weighting matrix $R > 0$. In addition, output covariance bound matrix $\overline{Y} = block\ diag\ \left[\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_m\right] > 0$,initial weighting matrix $Q(0) = block\ diag\ [Q_1(0),\ Q_2(0),\ \ldots,\ Q_m(0)] > 0$, convergence criterion $\varepsilon > 0$, and constants $\alpha > 0,\ 0 < \beta < 1$. Let $j = 0$, and proceed to Step 2.

2. Solve the following Riccati equation for $\tilde{X} \geq 0$ and let $L = \tilde{X} M_p^T V^{-1}$

$$0 = A_p \tilde{X} + \tilde{X} A_p^T - \tilde{X} M_p^T V^{-1} M_p \tilde{X} + D_p W_p D_p^T \tag{3.5.39}$$

3. Compute $K(j) \geq 0$ by solving the following Riccati equation and letting $F(j) = -R^{-1} B_p^T K(j)$.

$$0 = A_p^T K(j) + K(j) A_p - K(j) B_p R^{-1} B_p^T K(j) + C_p^T Q(j) C_p \tag{3.5.40}$$

4. Compute $X_p(j)$ by solving the following Lyapunov equation.

$$0 = (A_p + B_p F(j)) X_p(j) + X_p(j) (A_p + B_p F(j))^T + L V L^T \tag{3.5.41}$$

5. Compute the output covariance matrices.

$$Y_i(j) = C_i \begin{bmatrix} I \\ 0 \end{bmatrix} \left\{ \tilde{X} + X_p(j) \right\} \begin{bmatrix} I & 0 \end{bmatrix} C_i^T = C_{p,i} \left\{ \tilde{X} + X_p(j) \right\} C_{p,i}^T, \; i = 1, \; 2, \; \ldots, \; m \tag{3.5.42}$$

6. If the following conditions are satisfied,

$$\sum_{i=1}^{m} \left\| \left[Y_i(j) - \overline{Y}_i \right] Q_i(j) \right\| \leq \varepsilon, \; Y_i(j) \leq \overline{Y}_i, \; i = 1, \; 2, \; \ldots, \; m \tag{3.5.43}$$

the OCC algorithm is converged, and the resulting dynamics controller matrices in the form of 3.5.27 are listed in equation 3.5.44. Otherwise, proceed to Step 7.

$$A_c = A_p + B_p F(j) - L M_p, \quad B_c = L = \tilde{X} M_p^T V^{-1}, \; C_c = F(j) = -R^{-1} B^T K(j) \tag{3.5.44}$$

7. Update output weighting matrix $Q(j)$ as follows.

$$Q(j+1) \; = \; \beta Q(j) + (1 - \beta) \mathcal{P}[Q(j) + \alpha \{ Y(j) \; - \; \overline{Y} \}] \;) \tag{3.5.45}$$

where $Y(j) = block \; diag \; [Y_1(j), \; Y_2(j), \; \ldots, \; Y_m(j)]$ and $\mathcal{P}[\,\cdot\,]$ is an operator that projects any symmetric matrix to the closest positive semidefinite matrix. Let $j = j + 1$, and proceed to Step 3.

Remark 3.9

The norm used in equation 3.5.43 can be either 2-norm (maximum singular value of the matrix) or Frobenius norm (square root of summation of all squared entries). Consider the projection operator $\mathcal{P}[\,\cdot\,]$.For a real symmetric matrix M, real Schur decomposition leads to $M = [U_1, \; U_2] \{ block \; diag [E_p, \; E_n] \} [U_1, \; U_2]^T$, where E_p and E_n are diagonal matrices containing strictly positive and nonpositive eigenvalues of M in decreasing order, respectively, and $[U_1, \; U_2]$ is a unitary orthogonal matrix. Then, $\mathcal{P}[\,\cdot\,]$ can be defined below.

$$\mathcal{P}[M] = \begin{cases} 0, & M \leq 0 \\ U_1 E_p U_1^T, & otherwise \end{cases} \tag{3.5.46}$$

Remark 3.10

Theorems 2.2 and 2.5 in reference [6] state if the OCC algorithm is converged, it provides a global optimal solution to the OCC problem. In addition, if the optimal solution exists, for a small enough parameter $\alpha > 0$, the algorithm convergence is guaranteed.

With the OCC control problem introduced along with its iterative control design algorithm, we are now ready to discuss how to use the OCC control design method for control gain tuning to achieve the best performance for the given control design model. Note that $\tilde{Y}_i = C_{p,i} \tilde{X} C_{p,i}^T \; (i = 1, \; 2, \; \ldots, \; m)$ provides the minimal output covariance matrices that can be achieved.

If the open-loop system is stable, calculate the open-loop system state covariance matrix X_o by solving the following Lyapunov equation. If it is not stable, find a low gain stabilizing controller to stabilize it. For instance, design an LQG controller with extremely small output weighting matrix Q to obtain a stabilizing controller.

$$0 = A_p X_o + X_o A_p^T + D_p W_p D_p^T \tag{3.5.47}$$

Then, the open-loop output covariance matrices can be calculated using the following formula.

$$\hat{Y}_i = C_{p,i} X_o C_{p,i}^T, i = 1,\ 2,\ \ \ldots,\ m \tag{3.5.48}$$

We can generate a sequence of output covariance constraint by varying parameter $0 < \gamma < 1$ as follows.

$$\overline{Y}_i(l) = \tilde{Y}_i + \gamma(l)\left(\hat{Y}_i - \tilde{Y}_i\right), i = 1,\ 2,\ \ \ldots,\ m \tag{3.5.49}$$

For example, letting $\gamma(l) = 0.1 \times l - 0.05$ $(l = 1,\ 2,\ \ \ldots,\ 10)$ generates 10 upper bound covariance matrices that can be used to design 10 OCC controllers with gains from low to high with $l = 10,\ 9,\ \ \ldots,\ 1$. Now Example 3.19 demonstrates this control design process.

Example 3.19

Consider the system defined in equation 3.5.25 with system matrices A_p, B_p, D_p, C_p, and M_p shown below, where control weighting is $R = 1$, system input noise intensity is $W_p = 1$, and the measurement noise intensity matrix is $V = 0.001 I_2$.

$$A_p = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, B_p = D_p = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},\ C_p^T = M_p^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{3.5.50}$$

We will first design an OCC controller using the algorithm proposed before Remark 3.9. The constraints on output variances for outputs y_1 and y_2 $(m_1 = m_2 = 1$ and $m = 2)$ are listed below.

$$\overline{Y}_1 = 0.2073,\ \overline{Y}_2 = 0.1128 \tag{3.5.51}$$

For $\varepsilon = 0.0001$ with $\beta = \alpha = 0.5$, the OCC algorithm converges in 51 iterations with the following controller matrices defined in equation 3.5.27

$$A_c = \begin{bmatrix} -0.9997 & -0.0246 & 0 \\ -1.0246 & -8.9503 & 1.0000 \\ -3.6393 & -49.0405 & -4.2335 \end{bmatrix},\ B_c = \begin{bmatrix} 0.9997 & 1.0246 \\ 1.0246 & 8.9503 \\ 1.2449 & 40.4793 \end{bmatrix},\ C_c^T = \begin{bmatrix} -3.3944 \\ -8.4612 \\ -5.2335 \end{bmatrix} \tag{3.5.52}$$

and the output variances $Y_1 = \overline{Y}_1 = 0.2073$ and $Y_2 = \overline{Y}_2 = 0.1128$ with converged output weight matrix Q as below.

$$Q = \begin{bmatrix} 4.7332 & 0 \\ 0 & 53.3180 \end{bmatrix} > 0 \tag{3.5.53}$$

Note that Q is positive definite because achieved variances equal the provided upper bounds, respectively, and the resulting Q is different from the given $Q = 10 I_2$ of LQG control. The resulting closed-loop system poles are the combination of these from $(A_p + B_p C_c)$ $(\lambda_1 = -0.3346,\ \lambda_{2,3} = -1.9495 \pm 1.8320j)$ and these from $(A_p - L M_p)$ $(\lambda_4 = -1.0015$ and $\lambda_{5,6} = -3.9743 \pm 3.9745j)$. The system estimation and control gains (L and F) are listed in equation 3.5.54. We can see that the estimation gain L is the same as that from the LQG control example (Example 3.18) because the same input and measurement noise intensity matrices are used. In addition, the control gain F is quite different from the LQG one because the resulting output weighting matrix Q is quite different from the LQG one ($Q = 10 I_2$).

$$\begin{aligned} F &= -\begin{bmatrix} 3.3944 & 8.4612 & 5.2335 \end{bmatrix} \\ L^T &= \begin{bmatrix} 0.9997 & 1.0246 & 1.2449 \\ 1.0246 & 8.9503 & 40.5793 \end{bmatrix} \end{aligned} \tag{3.5.54}$$

Now, we will discuss the tradeoff relationship between control and output variances. The open-loop system is not stable. Therefore, a stabilizing LQR controller is needed with very small weighting matrix $Q = 10^{-6} I_2$ and $R = 1$, and using MATLAB command Fs=-lqr(Ap,Bp,Qp,R,0)($Ap = A_p$, $Bp = B_p$, and $Qp = C_p^T Q C_p$, $Fs = F_s$,) to obtain the stabilizing controller below.

$$F_s = -\begin{bmatrix} 2.0000 & 1.3647 & 2.9311 \end{bmatrix} \tag{3.5.55}$$

The corresponding stabilizing output covariance matrix $\hat{Y}$ and lower bound output covariance matrix $\tilde{Y}$, defined in equation 3.5.49, are shown below. In this case, the stabilizing controller is used only to calculate an upper bound for output variance bound matrix $\hat{Y}$ below.

$$\hat{Y} = \begin{bmatrix} 0.5904 & 0 \\ 0 & 0.3057 \end{bmatrix} > 0, \tilde{Y} = \begin{bmatrix} 0.0009997 & 0.0010246 \\ 0.0010246 & 0.0089503 \end{bmatrix} \tag{3.5.56}$$

As discussed earlier, assume fifteen-dimensional vector γ as below. Note that as γ reduces, the resolution for γ should become fine because the sensitivity of designed controller gain goes high as γ reduces.

$$\gamma^T = [0.95,\ 0.85,\ 0.75,\ 0.65,\ 0.55,\ 0.45,\ 0.35,\ 0.3,\ 0.25,\ 0.20,\ 0.15,\ 0.125,\ 0.1,\ 0.075,\ 0.05] \tag{3.5.57}$$

Using the formulae from equation 3.5.49, fifteen constraints of $\overline{Y}_i(l)$ $(i = 1, 2, l = 1, 2, \ \ldots, \ 15)$ can be generated based upon $\hat{Y}$ and $\tilde{Y}$ defined in equation 3.5.53. In addition, fifteen OCC controllers with gains low to high can be designed. In fact, the OCC controller shown earlier is the seventh OCC controller with $\gamma(7) = 0.35$. With the fifteen designed OCC controllers, we can calculate the unweighted OCC control and output performance costs ($\mathcal{E}_\infty[u(t)u^T(t-\tau)]$ and $\mathcal{E}_\infty[y_1(t)y_1^T(t-\tau)] + \mathcal{E}_\infty[y_2(t)y_2^T(t-\tau)]$). They are denoted as $\mathcal{E}_\infty[u^Tu]$ and $\mathcal{E}_\infty[y^Ty]$, respectively, in Figure 3.34 as horizontal axis and vertical axis.

Note that the blue line with circles in Figure 3.34 is the tradeoff relationship between $\mathcal{E}_\infty[u^Tu]$ and $\mathcal{E}_\infty[y^Ty]$ without any system modeling error. That is, the designed controller is applied to the control design model without any modeling error. Under this ideal situation, the closed-loop system performance $\mathcal{E}_\infty[y^Ty]$ continuous improving as the control energy $\mathcal{E}_\infty[u^Tu]$ increases. In practice, however, there are always modeling errors, which could lead to a quite different system performance. Consider the actual physical system model described by modified A_p and B_p as below (D_p, C_p, and M_p remain unchanged). Uncertainties are marked in red.

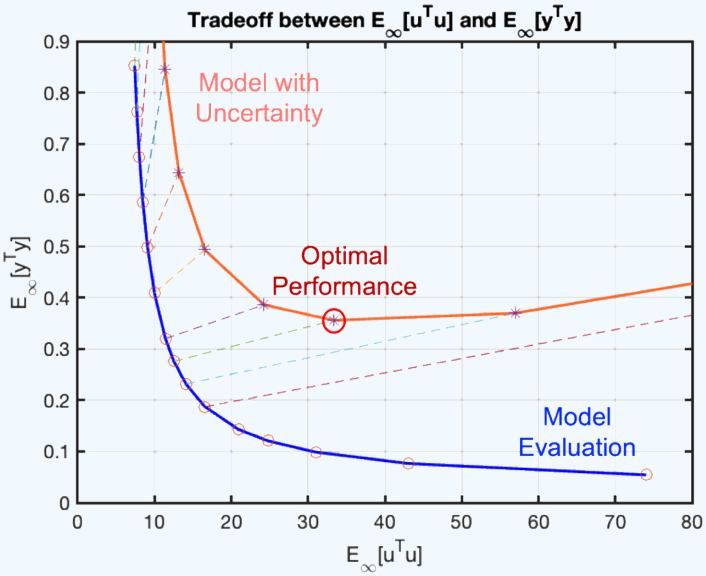


Figure 3.34: Control and performance tradeoff relationship between design and uncertainty models.

$$A_p = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1.0 & 1 \\ 1 & 0.2 & 1 \end{bmatrix}, B_p = \begin{bmatrix} 0 \\ -0.15 \\ 1 \end{bmatrix} \tag{3.5.58}$$

Fifteen closed-loop systems were formed using the actual physical model in equation 3.5.58 based upon the fifteen designed OCC controllers with gains from low to high. Both control and output variances, $\mathcal{E}_\infty[u^Tu]$ and $\mathcal{E}_\infty[y^Ty]$, are calculated for these fifteen closed-loop systems. They are plotted in Figure 3.34 in red line with stars. We can see that the output variance is higher than those from the closed-loop system evaluated with the design model with increased control variance.

In addition, beyond certain control variance $\mathcal{E}_\infty[u^T u]$ level (around 33), the output performance in terms of output variance $\mathcal{E}_\infty[y^T y]$ gets worse with increased $\mathcal{E}_\infty[y^T y]$. This indicates that due to the modeling error, the best closed-loop system performance in terms of output variance $\mathcal{E}_\infty[y^T y]$ is achieved at a certain level of control effort in terms of $\mathcal{E}_\infty[u^T u]$. Beyond that level, increasing control effort will not improve the closed-loop system performance. Instead, it could make the resulting closed-loop system performance worse or unstable. That is, practically, control effort cannot be increased to a very high level.

Remark 3.11

From Example 3.19, it is important to use the OCC control design algorithm to find a family of controllers with control gains from low to high. As a model-based control gain tuning method, we can evaluate each controller by starting from the lowest gain and gradually increasing the gain until the system performance gets worse. Using this process, we can find the best performance controller from the family of designed OCC controllers. References [7] and [39] address this control tuning method with application to the Hubble Telescope [7] and JPL LSCL space antenna [39].

Section Summary

In addition to PID control, model-based control techniques are often used for mechatronic systems. In this section, we first introduced the closed-loop pole assignment technique. Then we reviewed the LQR (linear quadratic regulation) and LQG (linear quadratic Gaussian) optimal control techniques [4], [5] for linear mechatronic systems. The main challenge of utilizing the LQR and LQG control for mechatronic systems is determining how to tune the model-based controller. Note that PID control is easy to tune because it has only three gains to deal with. The OCC (output covariance constraint) control technique [6], [7], [8] was presented to address the issue of model-based control gain tuning; a family of OCC controllers can be easily generated with gains from low to high, making it feasible to tune the model-based control gain experimentally.

Section Keywords

Practice Problems

1. Consider the system defined in equation (3.74) with system matrices A and B shown below.

$$A_p = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 2 & 1 \end{bmatrix}, \quad B_p = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

   Design a state feedback gain F to assign the eigenvalues of $(A_p + B_p F)$ at $\lambda_1 = -2$ and $\lambda_{2,3} = -2 \pm j$ so that the closed-loop system is stable.

2. Design an LQR controller using the MATLAB command `lqr` for the system model described in P3.18 above with $R = 1$ and $Q = \text{diag}[1, 2, 3]$.

3. Design an LQG controller defined in equation 3.5.27 by finding the estimation gain L and control gain F using MATLAB command "`lqr`" for the system model defined in (3.98) with the following system matrices.

$$A_p = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 2 & 1 \end{bmatrix}, \quad B_p = D_p = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad C_p^T = M_p^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}^T$$

   where $W_p = 1, V = 0.001 I_2, R = 1$, and $Q = I_2$. Form the dynamic controller system matrices A_c, B_c, and C_c defined in equation (3.100) and closed-loop system matrices A, D, and C. Compare eigenvalues of A and those of $(A_p + B_p F)$ and $(A_p - L M_p)$.

4. Use the system model in Example 3.19 to design an OCC controller with $\gamma(7) = 0.35$.

This page titled 3.5: Review of Model-Based Control Design and Gain Tuning Methods is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

3.6: Machine Learning-Based Control

Learning Objectives

After completing this section, you will be able to:

1. Describe the concept of machine learning control.
2. Provide a process of applying machine-learning technique to optimize PID control gains.

PID control is a model-free control method. In the practical control of nonlinear systems, it plays a crucial role due to its simple structure and performance-oriented tuning process. To improve the control performance, an advanced gain scheduling method is often used to schedule the PID control gains based on operating conditions and tracking errors. However, determining the scheduled gain is a major challenge because PID control gains need to be determined at each operating condition.

This section presents a model-assisted online optimization method. The method is based on the modified non-dominated sorting genetic algorithms-II (NSGA-II) [23], [24], [25], [26]. The goal is to obtain the optimal gain-scheduled PID controller. The optimal control is one where a model-assisted, offline optimization—found through software-in-the-loop (SiL) simulation—provides the initial scheduled gains so that the online iterative NSGA-II algorithm can automatically schedule and tune PID gains by online searching in the parameter space.

The genetic algorithm (GA) [27], [28] can be used to solve the optimization problem of control gain tuning, especially for highly nonlinear systems. It is based on biological principles of natural selection and regeneration. The GA search (optimization) does not guarantee achieving the global optimal solution in theory. Still, it conducts a model-free search over a large parameter space.

For the traditional single-objective genetic algorithm, there is only one predefined objective function to be optimized. However, in order to evaluate the closed-loop system performance, a few performance indices should be considered simultaneously with trade-off relationships. These indices include overshoot, settling time, and steady-state error.

One method is to combine these into a weighted sum to form a single objective optimization problem [29], [30], [31]. Therefore, the obtained optimal solution is dependent on these selected weightings. With advancement of multi-objective optimization methods [32], this limitation can be overcome.

The NSGA-II method [33] is one of the best-known multi-objective optimization algorithms that are computationally efficient. The NSGA-II algorithm provides a Pareto-front set of solutions. These solutions allow decision makers to observe the trade-off relations among different objectives. Then, a final solution can be selected based on desired performance requirements.

Because there is no weighting to be selected for multiple performance criteria, this method improves the multi-objective optimization process. See the early NSGA-II PID gain tuning studies [23], [24], [25], [26] and other multi-objective optimization problems [34], [35]. Compared with a fixed-gain PID controller, performance improvement can be achieved by using scheduled gains for systems with high nonlinearity. Tuning scheduled gains for the best performance introduces additional challenge over fixed gains. But the automatic tuning method can be developed based on the NSGA-II searching algorithm.

Figure 3.35 shows two common control optimization processes, where the path with black arrows represents the traditional offline process and the one with blue arrows the online one. Based on the identified or physical system model, offline optimization can be conducted with efficient searching in the simulation environment. The optimal controller obtained from simulation study is then used on the test bench or physical system, assuming that the identified (or physical) model is accurate enough to represent the real system.

Because the modeling error can never be avoided, however, especially for systems with high nonlinear friction, the PID controller obtained from offline optimization may not provide satisfactory performance for the physical system. Although the system modeling accuracy can be improved with increased effort, it is impossible to completely eliminate modeling error. The online optimization method applies the NSGA-II algorithm directly during bench tests (see the red arrow path in Figure 3.35). To proceed with an intense search, a large population size with many generations is necessary, resulting in a massive number of test points.

Unlike simulation design, the online global search is usually computationally infeasible due to the limitations of real-time microprocessor, actuator durability, etc. Instead, a co-optimization strategy was proposed using both offline and online information, as shown by the blue arrow path in Figure 3.35. With the advantage of offline global search, an optimal gain-scheduled PID controller can be obtained based on adequate population size and deep generation based on the offline model. The offline optimization result is then used as an initial condition for online optimization using an iterative NSGA-II algorithm. The PID controller is automatically tuned online with reduced computational cost based on the proposed iterative NSGA-II algorithm. Note that this approach combines the comprehensiveness of offline search and the authenticity of online search.

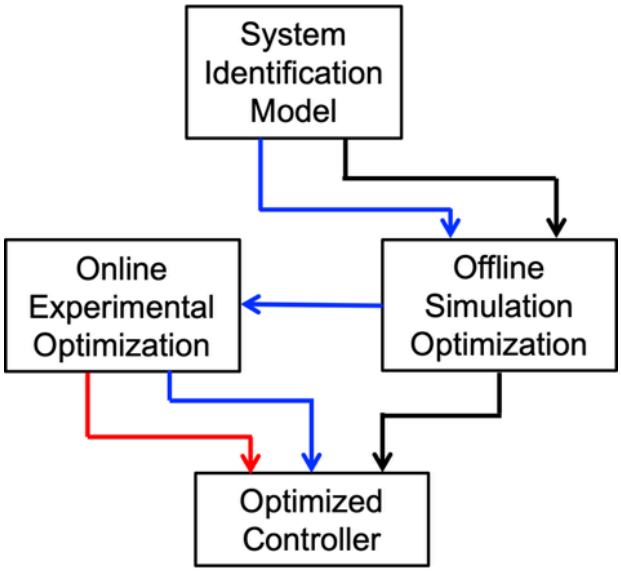


Figure 3.35: Control design: offline (block), online (red), and model-assisted (blue) optimization.

This section uses an EGR (exhaust gas recirculation) valve system with highly nonlinear friction as an example for tuning scheduled PID gains. This approach is based on the proposed NSGA-II optimization algorithm that uses offline optimized control gains to obtain initial gains. The scheduled PID gains are optimized (tuned) using the NSGA-II algorithm with reduced scheduling parameters and then validated experimentally. Figure 3.36 shows the EGR valve system diagram.

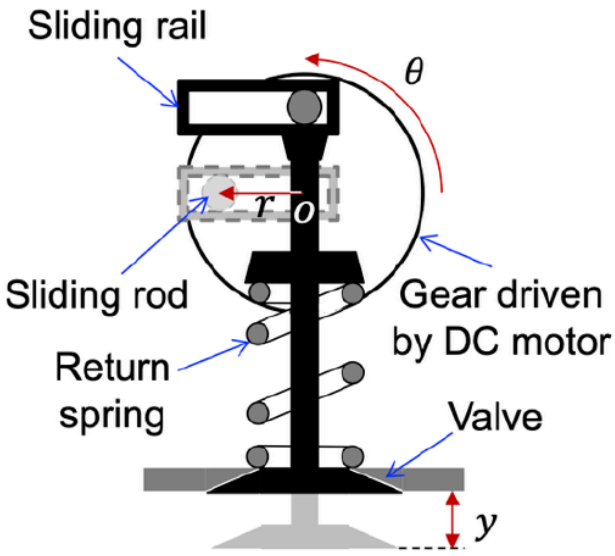


Figure 3.36: EGR valve system diagram.

The EGR valve can be modeled using the following differential equation.

$$\begin{aligned}\left(J_r + m_l r^2 \sin^2\theta\right)\ddot{\theta} &= K_i u - K_s\left(\theta + \theta_{pre}\right) - m_l r^2 \dot{\theta}^2 \cos\theta \sin\theta \\ &= -\left(T_c + \frac{T_r - T_c}{1 + \left(\dot{\theta}/\dot{\theta}_s\right)^2} + K_v\dot{\theta}\right)\end{aligned} \tag{3.6.1}$$

where the nonlinear friction torque is modeled by the Stribeck friction model [36], [37] in the second line of equation 3.6.1. In addition, u is the control input (DC motor armature voltage), and θ is the rotational angle of actuator defined in Figure 3.36. The rest of the model parameters are defined with their values in Table 3.11. Note that this model matches well with the bench test results under different operational conditions.

Table 3.11: EGR model parameter definition and their values.

Total rotational assembly moment of inertia	J_r	$0.42\ kg/m^2$
Rotational radius to moving mass m_l	r	$0.015\ m$
Moving mass including valve mass	m_l	$0.28\ kg$
Actuator torque coefficient	K_i	$205.3\ Nm/dutycycle$
Return spring stiffness	K_s	$0.393\ Nm/rad$
Spring preload angle	θ_{pre}	$0.873\ rad$
Static friction torque	T_r	$57.01\ Nm$
Coulomb friction torque	T_c	$0.285\ Nm$
Viscous torque coefficient	K_v	$0.689\ Nm/rad$

Consider the fixed-gain PID controller optimized by the NSGA-II algorithm (see discussions after equation 3.6.3) from bench tests with PID gains of $P = 2494.7$, $I = 32.07$, and $D = 9.769$. In this case, the I (integral) and D (derivative) gains are relatively small, compared with the P (proportional) gain. Note that a large I gain could result in quick windup and instability. A large D gain could make the controller sensitive to system and measurement noises.

Normally, the proportional, integral, and derivative scheduled gains (functions) should be optimized simultaneously, but this could increase search space dimension and result in extremely high number of iterations to converge. Because we are interested in finding an optimization scheme feasible for real-time implementation, a three-step searching process is used to find optimal scheduled PID gains sequentially. See Figure 3.37, where e denotes the valve position error between the reference and feedback signals. The three-step optimization process described in Figure 3.37 is self-explanatory.

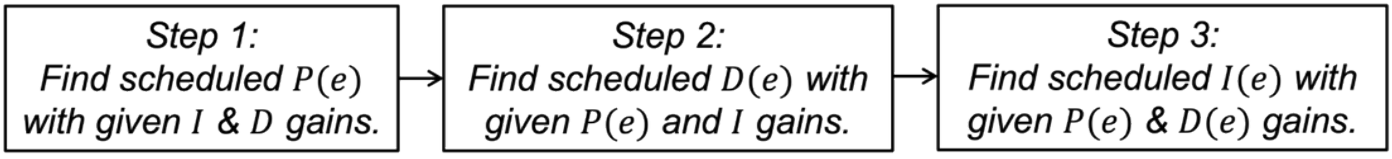


Figure 3.37: Three-step search process.

Due to the high static friction characteristics, the valve position tracking error e is selected as the scheduling parameter. High PID control gains are expected at the region with small tracking error to overcome the increased friction force to reach the final destination. Considering the computational cost, the number of variables in the scheduled gain function should be low. This is true even though, in many cases, complex scheduling functions could improve the closed-loop system performance further. After studying multiple scheduling functions, we can select the following $f(e)$ in equation 3.6.2 as the gain functions of $K_p(e)$, $K_i(e)$, and $K_d(e)$ with an saturation at $|e| = 0.1$ when $|e| < 0.1$.

$$f(e) = \begin{cases} \frac{a}{|e|^b} + c, & |e| \geq 0.1 \\ \frac{a}{0.1^b} + c, & |e| < 0.1 \end{cases} \tag{3.6.2}$$

Figure 3.38 shows the gain scheduled PID control system architecture. Here, e and u represent the valve position error and control, respectively. In addition, $K_p(e)$, $K_i(e)$, and $K_d(e)$ are the scheduled proportional, integral and derivative gains in the form of $f(e)$, respectively. In another words, the optimization is aimed to search for the best shape of $K_p(e)$, $K_i(e)$, and $K_d(e)$.

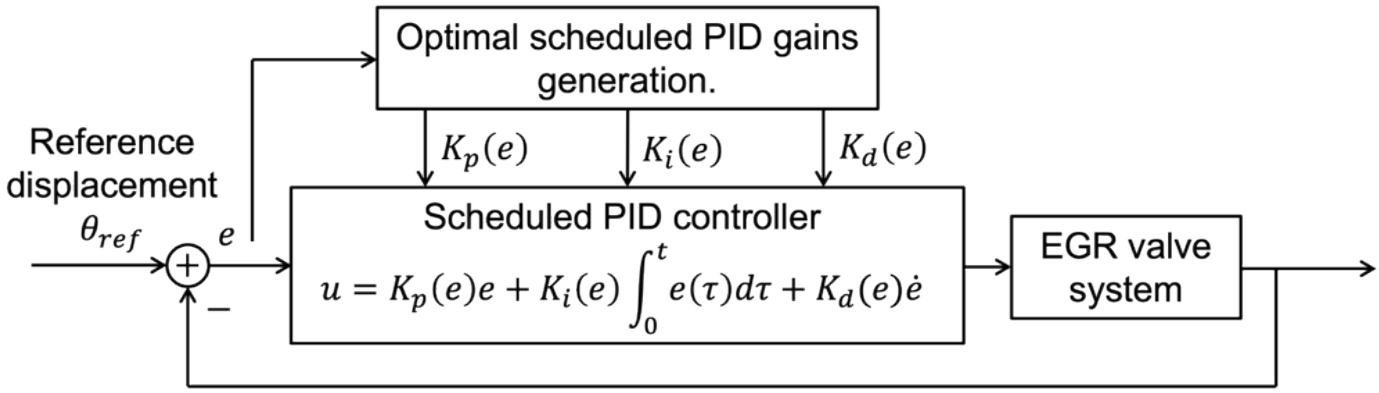


Figure 3.38: Gain-scheduled PID control system architecture.

With the gain-scheduled PID controller architecture presented, we are ready to discuss the optimization objective functions. In order to obtain the optimal control performance under multiple operational conditions, four simulation or test conditions are designed to evaluate the proposed algorithm as follows.

- Case 1: Step response from close to 2% opening
- Case 2: Step response from close to 50% opening
- Case 3: Step response from close to 80% opening
- Case 4: Step response from 50% to 30% opening

Performance criteria of the closed-loop system are settling time, overshoot, and steady-state error. These three criteria are included into the objective functions. Normally, for single-objective optimization, the objective function can be defined as $J = \int_0^\infty e^2(t)\,dt$, i.e., the integral of squared error (ISE). However, for a system with high static friction, if the single objective function is used, large overshoot could exist in the optimized closed-loop response with minimal steady-state error. Thus, an additional objective function, related to the percentage overshoot (PO), is used to form a multi-objective optimization problem. Based on the discussions above, two objective functions are defined in equation 3.6.3.

$$\begin{aligned} J_1 &= \frac{1}{k}\sum_{i=1}^{4} PO(i) = \frac{1}{k}\sum_{i=1}^{4}\frac{|\theta_{peak}(i) - \theta_{ss}(i)|}{\theta_{ss}(i)} \\ J_2 &= \frac{1}{k}\sum_{i=1}^{4}\int_0^{t_1}\left(\frac{|\theta_i(t) - \theta_{ss}(i)|}{\theta_{ss}(i)}\right)^2 dt \end{aligned} \tag{3.6.3}$$

Here, i represents case number and total cases number is $k = 4$. In addition, $\theta_i(t)$, $\theta_{peak}(i)$ and $\theta_{ss}(i)$ are the valve position time response, response peak and steady state position for Case i ($i = 1,\ 2,\ 3,\ 4$), respectively. Furthermore, t_1 is the time when the EGR valve stopped moving for all four cases. In this case, the system percentage overshoot (PO) is associated with objective J_1. Both settling time and steady-state error are considered in J_2 in terms of ISE.

Traditionally, a multi-objective optimization problem can be converted into a single-objective one by using a sum of weighted objective functions. Thus, the optimal solution depends crucially on the selected weightings. Furthermore, some profitable tradeoff relationships may be ignored because a slight sacrifice of one objective may lead to a significant improvement of the other objective function. In a different way, the NSGA-II algorithm generates optimal solutions on a Pareto-front which includes multiple non-dominated solutions.

The tradeoff relationship between each objective can be clearly presented, allowing the decision-maker to pick the optimal solutions with specific criteria. There is no unique way to select the optimal individual solution among the nondominated solution set from NSGA-II because the performance evaluation criteria vary with different requirements. In this study, reference-point-based multiple-criteria decision-making (MCDM) [38] is applied by considering the origin point as the reference point, with equal weight for both objectives J_1 and J_2. Thus, we will select the point on the Pareto-front with minimum distance to the origin ($J_1 = 0$ and $J_2 = 0$) as the final optimal solution.

The detailed NSGA-II optimization method implemented in this section is described below. The parameters are predefined for calibrating the PID scheduled gains; see reference [33] and [34] for details.

- **Step 1. Initialization:** Letting $n = 0$, the initial parent population P_n is generated with size N_p. For each individual, all

decision variables x^i are randomly generated among the related search space $[x^i_{min}, x^i_{max}]$ with $i = 1,\ 2,\ \ldots,\ N_p$.

- **Step 2. Nondomination sorting:** The resulting individual is evaluated based on the nondomination level with the nondomination rank assigned; rank 1 is the best level, 2 the next-best level, and so on.
- **Step 3. Offspring generation:** Usual binary tournament selection is applied with binary crossover at P_C probability and D_C distribution index, polynomial mutation at P_M probability, and D_M distribution index for generating an offspring population Q_n with the size equal to N_p.
- **Step 4. Combining and sorting:** An extended set is generated by combining P_n and Q_n with a size of $2N_p$. After the nondominated sorting stated in Step 2, selection starts from low-ranked individuals and forms a new population P_{n+1} of size N_p until it is filled. If the rank of multiple individuals is the same and the remaining space in P_{n+1} is limited, the individual with larger crowding distance will have the priority to be selected; see details of the crowding distance in [33], [34]. Lastly, the new population set P_{n+1} is filled with the selected individuals.
- **Step 5. Iteration:** After updating the generation, let $n = n + 1$. If the stopping criterion is not satisfied, go to Step 2. Otherwise, the optimization is completed.

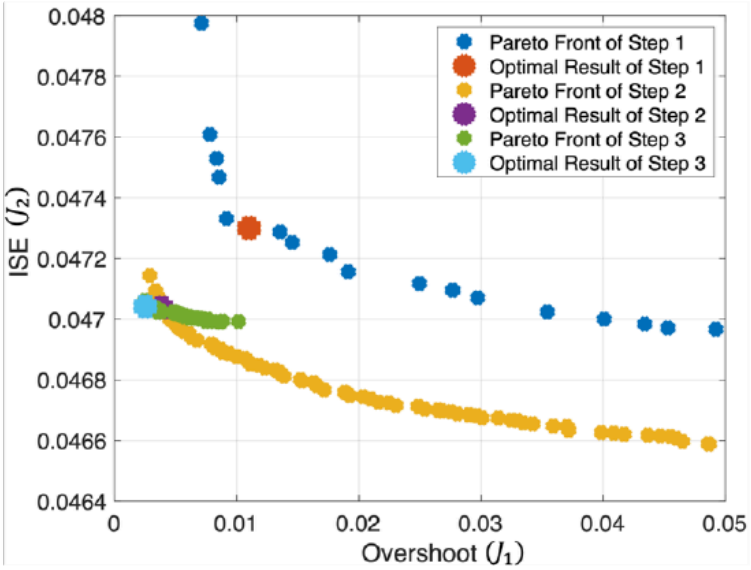


Figure 3.39: Pareto fronts of fixed and scheduled PID costs [15].

To demonstrate the advantages of the gain-scheduled PID controller, a fixed-gain PID controller is also optimized, using NSGA-II, for the high static friction model. To have a fair comparison, the search space, population, and generation size must be identical to the gain scheduled case, where the fixed gain PID gains are $P = 2494.7$, $\mathrm{I} = 32.07$, and $\mathrm{D} = 9.769$. Table 3.12 lists the search parameters for the NAGA-II algorithm (first parameter column for fixed-gain PID controller). Then, using the search parameters defined in Table 3.12, scheduled PID controller gains are searched in three steps (defined in Figure 3.37). resulting in the following scheduled PID gains.

$$\begin{cases} P_{off}(e) = \dfrac{1466}{max\{0.1, abs(e)\}^{0.4937}} + 916.17. \\ I_{off}(e) = 57.1\left(\dfrac{1.3763}{max\{0.1, abs(e)\}^{0.717}} + 0.098\right) \\ D_{off}(e) = 11.3\left(\dfrac{0.066}{max\{0.1, abs(e)\}^{1.0}} + 0.785\right) \end{cases} \tag{3.6.4}$$

Table 3.12: NSGA-II generic algorithm parameters used for all cases.

Parameter	Fixed PID	Step 1: PID	Step 2: PID	Step 3: PID
N_p	100	100	100	100
a_{range}	N/A	$P(e) : [0, 1500]$	$I(e) : [0,\ 2]$	$D(e) : [0,\ 2]$
b_{range}	N/A	$P(e) : [0,\ 20]$	$I(e) : [0,\ 3]$	$D(e) : [0,\ 3]$

Parameter	Fixed PID	Step 1: PID	Step 2: PID	Step 3: PID
c_{range}	N/A	$P(e):[0,\ 1000]$	$I(e):[0,\ 2]$	$D(e):[0,\ 2]$
P_{range}	$[0,\ 5000]$	N/A	N/A	N/A
I_{range}	$[0,\ 100]$	N/A	N/A	N/A
D_{range}	$[0,\ 100]$	N/A	N/A	N/A
P_C	90%	90%	90%	90%
P_M	30%	30%	30%	30%
D_C	10	10	10	10
D_M	20	20	20	20
Max Generation #	100	100	100	100
Optimized J_1	0.0428	0.0110	0.0039	0.0025
Optimized J_2	0.0479	0.0473	0.0470	0.0470

The optimized two costs (J_1 and J_2) of all four cases are also listed in Table 3.12. We can see that cost J_1 is significantly reduced from 0.0428 (fixed PID) to 0.0025 (scheduled). However, the reduction for cost J_2 is small (from 0.0479 to 0.0470). This can be also observed from the Pareto fronts obtained during the optimization process (see Figure 3.39).

Figure 3.40 presents the simulated step responses with the optimized PID gain-scheduled controller. We can see that the gain scheduled PID do improve the system performances; all PID gains gradually increase as position error reduces to overcome nonlinear friction.

Now, we are ready to discuss the online optimization process. In the offline model-based optimization, the optimal scheduled-gains are solved by searching three variables defined in equation 3.6.2. For the online optimization, the offline optimization problem is too complex to be used. Because of the larger number of decision variables, the offline optimization problem could lead to large population size and large number of generations. Also, the shape variation of the gain-scheduled function could bring additional complexity in the NSGA-II iterative strategy due to physical system noise and nonlinearity. As a result, the optimization problem is simplified to optimize three parameters k_p, k_i, and k_d that are used to scale the offline optimized scheduled gains $K_p^{off}(e)$, $K_i^{off}(e)$, and $K_d^{off}(e)$.

$$K_p(e) = k_p K_p^{off}(e), K_i(e) = k_i K_i^{off}(e), K_d(e) = k_p K_d^{off}(e) \tag{3.6.5}$$

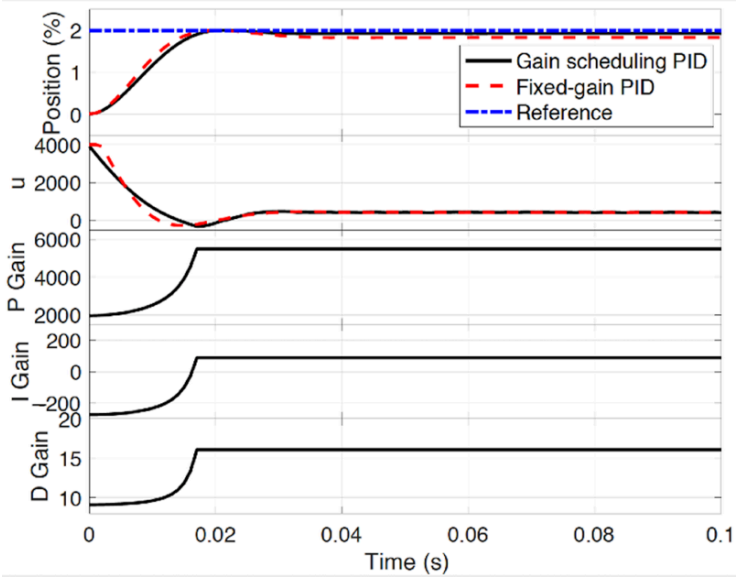


Figure 3.40: Simulation results for the offline optimized gain scheduled PID controller [15].

That is, the shape of a gain-scheduled function is fixed during the online optimization with only magnitude changes. The left plot of Figure 3.41 shows the step responses of simulation and experimental studies, where the initial and intermediate responses of the experimental result matches closely to the simulated ones because the system model is relatively accurate.

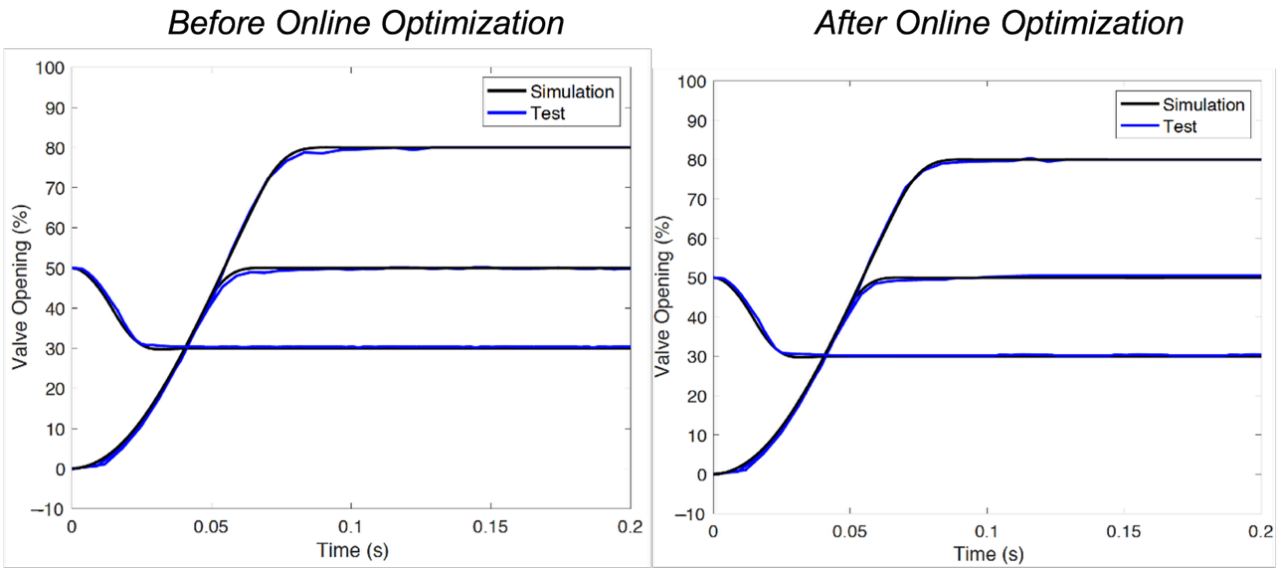


Figure 3.41: Simulation and test results of offline and online optimized gain scheduled PID controllers [15].

The simulated and experimental responses deviate when the valve approaches the set point. Although a well-calibrated Stribeck friction curve can be used to describe the nonlinear friction torque at low valve velocity, there still exists certain modeling error, leading to degradation of the control performance. As a result, in order to further improve closed-loop system performance, it is necessity to conduct an online optimization study based on the offline optimized controller. By analyzing the offline simulation results, we find that as long as the optimal scheduled gain function is obtained, the optimal gain scheduled PID controller can be used under different operational conditions. Therefore, to reduce computational cost and the number of experiments, we select the 0 to 50% opening case (Case 2) for online optimization.

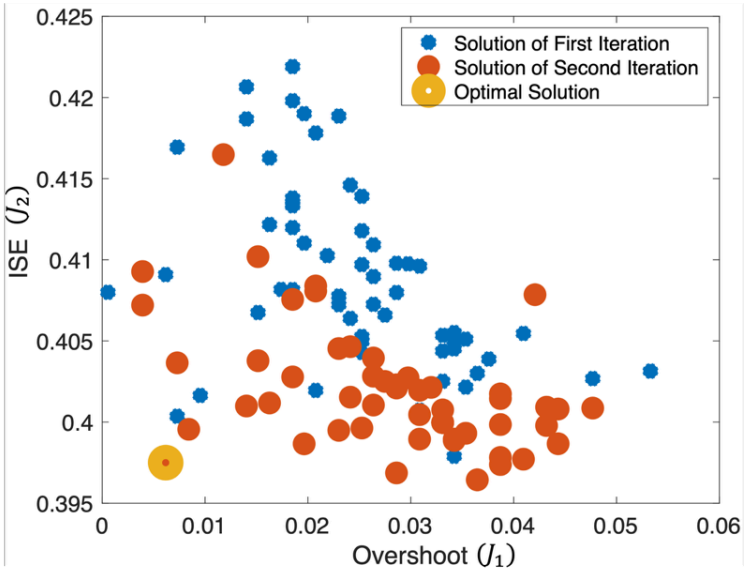


Figure 3.42: Pareto fronts of online optimized scheduled PID costs [15].

The NSGA-II optimization algorithm parameters in Table 3.12 are used for the online optimization with the parameter range of k_p, k_i, and k_d set to $0.99,\ 1.0,$ and 1.0, respectively. By evaluating the results of the first optimization, certain saturations on k_p, k_i , and k_d were observed near the Pareto front, leading to the search space migration of the second iteration. As we can see from Figure 3.42, after migrating to the new search space, the overall performance of all the population and Pareto front population of the second iteration (red dots) are improved over the first iteration (blue dots).

Finally, the yellow circled point was selected as the optimal solution because it has a minimal distance to the origin, and the online optimized control performance is improved (see the right plot of Figure 3.41). In summary, Table 3.13 shows the comparison of offline fixed gain (F-G), offline gain scheduling (G-S), and online optimized (O-G-S) PID controller. It looks like that the major improvement is in the settling time.

Table 3.13: Summary of control performances.

Controller type	Fixed gain PID	Offline Optimized PID	Online Optimized PID
2% settling time	70.1 (ms)	68.7 (ms)	62.7 (ms)
Steady state error	0.3%	0.2%	0.2%
Overshoot (%)	0.007%	0.007%	0.007%

Section Summary

This section presents an offline model-assisted online optimization method to automatically tune the gain-scheduled PID gains for an exhaust-gas-recirculation (EGR) valve with large nonlinear friction. The optimized gain-scheduled PID controller can adaptively compensate for the increased friction force at low-velocity regions. In the optimization problem, two objective functions are proposed to consider commonly used performance indices such as settling time, overshoot, and steady-state error. The NAGA-II optimization result provides decision makers with a clear tradeoff Pareto front. The optimal initial gain-scheduled functions can be obtained using the multi-objective NSGA-II algorithm, eliminating the process of manually tuning control gains. Next, an online iterative NSGA-II optimization is proposed and used to further optimize closed-loop system performance with significantly reduced bench tests requirement. The experimental results show that the automatic-tuned optimal gain-scheduled PID controller has better overall performance than the fixed-gain one.

Section Keywords

Gain-scheduled PID control: This strategy adjusts the proportional (P), integral (I), and derivative (D) gains of a PID controller based on the current operating conditions of a dynamic system.

Generic algorithm (GA): This optimization technique mimics natural selection to find solutions to complex problems, or to a generic programming concept where algorithms are written to be flexible and reusable across various data types.

NSGA-II (Non-dominated Sorting Genetic Algorithm II): This popular, elitist algorithm solves multi-objective optimization problems.

Practice Problem

1. Read reference [15]. Summarize the findings regarding the concept of machine learning (ML) control and PID control gain turning based on machine-learning techniques.

This page titled 3.6: Machine Learning-Based Control is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

3.7: Chapter References

References

1. R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 12th ed., Englewood Cliffs, NJ, USA: Prentice Hall, 2011, ISBN-13: 978-0-1360-2458-0.
2. H. K. Khalil, *Nonlinear Systems*, 3rd ed., Upper Saddle River, NJ, USA: Pearson, 2021, ISBN-13: 978-0-1306-7389-3.
3. J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Trans. of the ASME*, vol. 64, pp. 759–768, 1942.
4. D. S. Naidu, *Optimal Control Systems*, 1st ed., Boca Raton, FL, USA: CRC Press, 2003, ISBN-13: 978-1-3152-1442-9.
5. M. Athens and P. L. Flab, *Optimal Control: An Introduction to the Theory and Its Applications,* Garden City, NY, USA: Dover Publications, 2006, ISBN-10: 0-48-645328-6.
6. G. Zhu, M. A. Rotea, and R. E. Skelton, "A convergent algorithm for the output covariance constraint control problem," *SIAM J. of Control and Optim.,* vol. 35, no. 1, pp. 341–361, Jan. 1997.
7. G. Zhu, K. M. Grigoriadis, and R. E. Skelton, "Covariance control design for the Hubble Space Telescope," *AIAA J. of Guid., Control, and Dyn.,* vol. 18, no.2, pp. 230–236, May 1995.
8. G. Zhu and R. E. Skelton, "Mixed L_∞ and L_2 problems by weight selection in quadratic optimal control," *Int. J. of Control,* vol. 63, no. 5, pp. 1161–1176, 1991.
9. S. Zhang, J. Yang, and G. Zhu, "LPV modeling and mixed constrained H_2/H_∞ control of an electronic throttle," *IEEE/ASME Trans. on Mechatronics,* vol. 20, no. 5, pp. 2120–2132, Oct. 2015, doi: 10.1109/TMECH.2014.2364538.
10. A. K. Al-Jabouri, A. White, S. Zhang, G. Zhu, and J. Choi, "LMI solutions to the input covariance constraint control: application to electronic throttle control," *ASME J. of Dyn. Syst., Meas., and Control,* vol. 137, p. 091010–1, Sep. 2015, doi: 10.1115/1.4030525.
11. A. K. Al-Jabouri, G. Zhu, S. SM Zwei, W. Su, and N. T. Nguyen, "LPV modeling of a flexible wing aircraft using adaptive model gridding and alignment methods," *J. of Aerosp. Sci. and Tech.,* vol. 66, pp. 92–102, Jul. 2017, doi: 10.1016/j.ast.2017.03.009.
12. T. He, G. Zhu, S. S-M Swei, and W. Su, "Smooth-switching LPV control for vibration suppression of a flexible airplane wing," *Aerosp. Sci. and Tech.,* vol. 84, pp. 895–903, Jan. 2019, doi: 10.1016/j.ast.2018.11.029.
13. T. He, G. Zhu, and S. S-M Swei, "A Sequential Design Approach for Switching H_∞ LPV Control," *Int. J. of Control, Automat., and Syst.,* vol. 19, no. 10, pp. 3354–3367, Oct. 2021, doi: 10.1007/s12555-020-0363-3.
14. J. Tang, D. Wen, C. Archer, J. Yi, and G. Zhu, "Borderline knock adaptation based on online updated surrogate models," *Int. J. of Engine Res.,* vol. 24, no. 7, pp. 2958–2972, Jul. 2023, doi: 10.1177/14680874221137485.
15. S. Qu, T. He, and G. Zhu, "Gain-scheduled PID control optimized by iterative genetic algorithm of NSGA-II with application to a nonlinear valve system," *MPDI J. of Appl. Sci. (Special Issue of Res. and Appl. of Intell. Control Algorithm),* vol. 13, no. 5, p. 6444, May 2023, doi: 10.3390/app13116444.
16. S. Chapman, *MATLAB Programming for Engineers*, 6th ed., Independence, KY, USA: Cengage Learning, 2019, ISBN-13: 978-0-3570-3039-4.
17. A. White, G. Zhu, and J. Choi, "Hardware-in-the-loop simulation of robust gain scheduling control of port-fuel-injection process," *IEEE Trans. on Control Syst. Tech.,* vol. 19, no. 2, pp. 1433–1443, Feb. 2011, doi: 10.1109/TCST.2010.2095420.
18. J. P. Hespanha, *Linear Systems Theory*, Princeton, NJ, USA: Princeton Univ. Press, 2009, ISBN-13: 978-0-6911-4021-6.
19. G. Zhu, *$\mathcal{L}_2$ to $\mathcal{L}_\infty$ multi-objective control for linear systems*, Ph.D. dissertation, School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN, USA, 1992.
20. K. Zhou and J. Doyle, *Essentials of Robust Control*, Upper Saddle River, NJ, USA: Prentice Hall, 1998, ISBN-10: 0-13-525833-2.
21. MATLAB, Accessed: December 28, 2025. [Online] Available: https://www.mathworks.com.
22. C.-T. Chen, *Linear System Theory and Design*, Oxford, NY, USA: Oxford Univ. Press, 1999, ISBN-13: 978-0-1951-1777-6.
23. H. V. H. Ayala and L. dos Santos Coelho, "Tuning of PID controller based on a multi-objective genetic algorithm applied to a robotic manipulator," *Expert Syst. with Appl.*, vol. 39, no. 10, pp. 8968–8974, Aug. 2012, doi: 10.1016/j.eswa.2012.02.027.
24. S. Panda, "Multi-objective PID controller tuning for a FACTS-based damping stabilizer using non-dominated sorting genetic algorithm-II," *Int. J. of Elect. Power & Energy Syst.*, vol. 33, no. 7, pp. 1296–1308, Sep. 2011, doi: 10.1016/j.ijepes.2011.06.002.
25. G. K. Pedersen and Z. Yang, "Multi-objective PID-controller tuning for a magnetic levitation system using NSGA-II," in *Proc. of the 8th Annu. Conf. on Genetic and Evolutionary Comput.* (ACM), Seattle, WA, USA, Jul. 2006, pp. 1737–1744.
26. X. Z. Guan, "Multi-objective PID controller based on NSGA-II algorithm with application to mainstream temperature control," in *Proc. of the 2009 Int. Conf. on Artif. Intell. and Comput. Intell. (IEEE),* vol. 4, Shanghai, China, Nov. 2009, pp. 21–25.
27. J. R. Koza, K. Deb, et al., *Genetic programming 1997: proceedings of the second annual conference*, San Francisco, CA, USA: Morgan Kauffmann Publishers, Jul. 1997, ISBN-10: 1-55-860483-9.
28. D. Whitley, "A genetic algorithm tutorial," *Statist. and Comput.*, vol. 4, no. 2, pp. 65–85, Oct. 1994, doi: 10.1007/BF00175354.

29. D. S. Pereira and J. O. Pinto, “Genetic algorithm-based system identification and PID tuning for optimum adaptive control,” in *Proc. of the 2005 IEEE/ASME Int. Conf. on Adv. Intell. Mechatronics* (IEEE), Monterey, CA, USA, Jul. 2005, pp. 801–806.
30. Y. Mitsukura, T. Yamamoto, and M. Kaneda, “A design of self-tuning PID controllers using a genetic algorithm,” in *Proc. of the 1999 American Control Conference,* vol. 2, San Diego, CA, USA, Jun. 1999, pp. 1361–1365.
31. M. S. Saad, H. Jamaluddin, and I. Z. Darus, “PID controller tuning using evolutionary algorithms,” *WSEAS Trans. on Syst. and Control,* vol. 7, no. 4, pp. 139–149, Oct. 2012.
32. T. Murata and H. Ishibuchi, “MOGA: Multi-objective genetic algorithms,” in *Proc. of the IEEE Int. Conf. on Evol. Comput.,* vol. 1, Perth, Western Australia, Nov. 29 to Dec. 1, 1995, pp. 289–294.
33. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multi-objective genetic algorithm: NSGA-II,” *IEEE Trans. on Evol. Comput.,* vol. 6, no. 2, pp. 182–197, Jan. 2002.
34. A. Pal, L. Zhu, Y. Wang, and G. Zhu, “Data-driven model-based calibration for optimizing electrically boosted diesel engine performance,” *Int. J. of Engine Res.,* vol. 24, no. 4, pp. 1515–1529, Apr. 2023.
35. J. Tang, D. Wen, C. Archer, J. Yi, and G. Zhu, “Borderline knock adaptation based on online updated surrogate models,” *Int. J. of Engine Res.,* vol. 24, no. 7, pp. 2958–2972, Jul. 2023.
36. H. Olsson, K. J. Åström, C. C. De Wit, M. Gäfvert, and P. Lischinsky, “Friction models and friction compensation,” *Eur. J. of Control*, vol. 4, no. 3, pp. 176–195, 1998.
37. S. Andersson, A. Söderberg, and S. Björklund, “Friction models for sliding dry, boundary and mixed lubricated contacts,” *Tribology Int.*, vol. 40, pp. 580–587, 2007.
38. A. P. Wierzbicki, “Reference point approaches and objective ranking,” in *Proc. of the Dagstuhl Seminar Proc., Schloss Dagstuhl-Leibniz-Zentrum für Informatik*, Hanover, Germany, Mar. 2007.
39. G. Zhu and R. E. Skelton, “Integrated modeling and control for the Large Spacecraft Control Laboratory experimental facility,” *AIAA, J. of Guid., Control, and Dyn.,* vol. 17, no.2, pp. 250–256, 1994.

Solutions to Practice Problems

1. The answer is a).

1.
2.

and

$$y_t(t) = 0, \quad y_{ss}(t) = 8(1 - \cos t) \tag{3.7.1}$$

1.

where $|G(j\omega)|$ and $\angle G(j\omega)$ are defined below.

$$|G(j\omega)| = 4\left((4-\omega^2)^2 + 4\omega^2\right)^{-0.5}, \qquad \angle G(j\omega) = -\tan^{-1}\left(\frac{4-\omega^2}{4\omega^2}\right) \tag{3.7.2}$$

1. The answer is b).
2. The answer is b).

1. These answers are listed below.
2. Marginally stable.
3. Stable.
4. Marginally stable.
5. Unstable.
6. Unstable.

1.

The stable range for K_I is \(0

1. Answers are listed below.
2. Stable.
3. Unstable.
4. Stable.
5. Unstable.

1.

Note that $\omega_g \approx 31$ rad/s, $\omega_p = \infty$, gain margin = ∞, and phase margin = 47°.

1.

Hence, we have the following solutions:

a) PO is zero (since first order system does not have any overshoot.

b) For the stable system above, the steady-state response for the unit input is $y_{ss} = \lim_{s \to 0} G(s) \ = G(0) = 2$.

c) $t_s\,(2\%) = 4T = 16$ (second) and $t_s\,(5\%) = 3T = 12$ (second).

1.

\[PO=100e^{\dfrac{-πζ}{\sqrt{1-ζ^2}}}<20\ \ \ \to \ \ \ ζ>\sqrt{\dfrac

ParseError: invalid DekiScript (click for details)

```
Callstack:
    at (Bookshelves/Introductory_Engineering/Mechatronics:_Fundamentals_Design_Integration_and
```

=0.456\ \ \to \ \ θ<{{\mathrm{cos}}^{-1} \left(0.456\right)\ }={62.87}^{{}^\circ }\]

The associated feasible region is plotted below, where red-shaded area is the feasible region.

1. See Example 3.13 for solution.
2. See Example 3.14 for solution.

1.
2.

with closed-loop eigenvalues of

$$s_1 = -2.7390, \quad s_{2,3} = -0.6185 \pm 0.3657j \tag{3.7.3}$$

1.

$$L^T = \begin{bmatrix} 0.9973 & 1.0734 & 1.7458 \\ 1.0734 & 9.2005 & 42.9005 \end{bmatrix} \tag{3.7.4}$$

The associated dynamic controller system matrices defined in ??? are

$$A_c = \begin{bmatrix} -0.9973 & -0.0734 & 0 \\ -1.0734 & -9.2005 & 1.0000 \\ -3.1600 & -46.4879 & -3.4892 \end{bmatrix}, \ B_c = \begin{bmatrix} 0.9973 & 1.0734 \\ 1.0734 & 9.2005 \\ 1.7458 & 42.9005 \end{bmatrix}, \ C_c^T = \begin{bmatrix} -2.4142 \\ -5.5873 \\ -4.4892 \end{bmatrix} \tag{3.7.5}$$

The eigenvalues of the closed loop system matrix defined in ??? are as follows.

$$s_{1,2} = -4.0981 \pm 3.8466j, \quad s_3 = -1.0015 \tag{3.7.6}$$

$$s_{4,5} = -0.6929 \pm 04384j, \quad s_6 = -2.1033 \tag{3.7.7}$$

Note that the eigenvalues of $(A_p + B_p F)$ are $s_{4,5}$, s_6 and eigenvalues of $(A_p - LM_p)$ are $s_{1,2}$, s_3, which confirms that the closed-loop system poles are these poles of $(A_p + B_p F)$ and $(A_p - LM_p)$.

1. Find solutions in Example 3.19.
2. These findings should include the gain scheduling PID controller structure and how the machine learning is applied to obtain scheduled gains.

This page titled 3.7: Chapter References is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

4: SIGNAL PROCESSING

Learning Objectives

1. Introduce the continuous-time analog signal and its frequency analysis using Laplace transforms.
2. Discuss discrete-time digital signal and its frequency analysis using both discrete Fourier transforms (DFT) and fast Fourier transforms (FFT). In addition, be familiar with the discrete-time z-transform of signal and discrete-time system models in z-transform that is important for digital filter and controller design.
3. Define signal conversions between analog and digital signals, along with realization hardware for analog-to-digital (A/D) and digital-to-analog (D/A) converters. Understand data conversion between binary and decimal numbers. For graduate-level courses, provide the single and double precision floating-point representation in microprocessors.
4. Describe low-pass, high-pass, band-pass, and band-kill (notch) filter design for both analog and digital signals. Also describe their implementation and characteristics.
5. Introduce Nyquist-Shannon sample theory, sample aliasing, and anti-aliasing filter design.
6. Define the digital interface between the physical and information domains using pulse-train signals such as pulse-width-modulation and frequency modulation methods.
7. Introduce advanced filter design such as moving average, Chebyshev, and Butterworth filters.

This chapter introduces the concepts of signal processing in mechatronic systems, including analog-to-digital and digital-to-analog conversions; analog and digital signal frequency responses; low-pass, high-pass, band-pass, and band-kill (notch) filtering; Nyquist-Shannon sample theory; anti-aliasing filtering; and certain advanced filtering techniques.

4.0: Prelude

4.1: Analog Signals and Their Frequency Analysis

4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)

4.3: Digital Signals and Their Frequency Analysis

4.4: Analog and Digital Filter Design

4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters

4.6: Pulse-Width and Frequency Modulated Signals

4.7: Advanced Analog and Digital Filter Design and Implementation

4.8: Chapter References

This page titled 4: Signal Processing is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

4.0: Prelude

In Chapter 1, we discussed that the mechatronic system intersects mainly with four engineering disciplines: control systems, computer science and engineering, electrical and electronic engineering, and mechanical engineering. Within a mechatronic system, these signals can be divided into two domains: digital information and physical domains. The digital domain handles digital information by the microprocessor (or microcontroller) whereas the physical domain handles analog information from the physical systems, including the sensors and actuators.(See Figure 1.2 in Chapter 1 for details as well as Figure 4.1).

This chapter addresses analog and digital signals in the physical and digital domains. We discuss how to convert analog signals to digital ones and vice versa, using analog-to-digital (A/D) and digital-to-analog (D/A) converters. We will present and analyze frequency responses of both analog and discrete signals as well as fast-Fourier transform (FFT). Consequently, we will introduce analog and digital filter design techniques. Later in the chapter, we discuss the Nyquist-Shannon sample frequency theorem along with sample aliasing and anti-aliasing filters. Figure 4.2 depicts the contents of Chapter 4, indicating the connection between the information (discrete-time) and physical (continuous-time) domains.

Section 4.1 addresses the analog signal in a continuous-time physical domain and its frequency domain analysis in terms of the signal magnitude and phase. The Bode plot, introduced in Section 3.2, is used to plot the signal frequency responses (magnitude and phase). We will also discuss the concept of angular frequency in radian per second (rad/s) and oscillation frequency in Hertz (oscillation per second, Hz) as well as their relationship. Note that in the Bode plot [1], the horizontal axis is in angular frequency ω (rad/s) and, in industry, the oscillation frequency f in Hertz (Hz) is widely used, where $\omega = 2\pi f$.

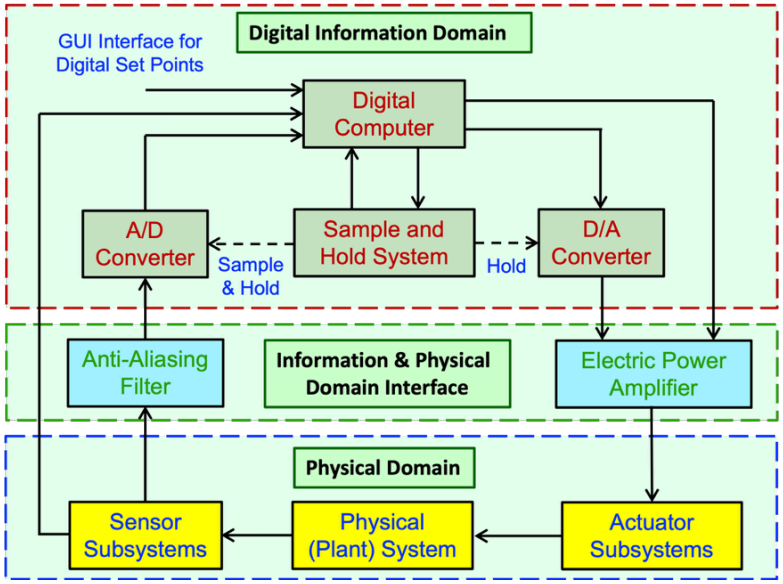


Figure 4.1: Information and physical domains with their interface.

As Figure 4.1 indicates, the information of mechatronic systems is often processed in the information domain using a microcontroller (or microprocessor). Therefore, the physical domain analog signals need to be converted into digital ones using analog-to-digital (A/D) converters [2], [3]. Furthermore, the information domain digital signals can be converted back into analog ones using digital-to-analog (D/A) converters [4], [5]. Section 4.2 discusses the sample and hold circuit, along with the relationship between the analog and sampled digital signals. The section also looks at typical analog-to-digital (A/D) and digital-to-analog (D/A) converter electronic circuits.

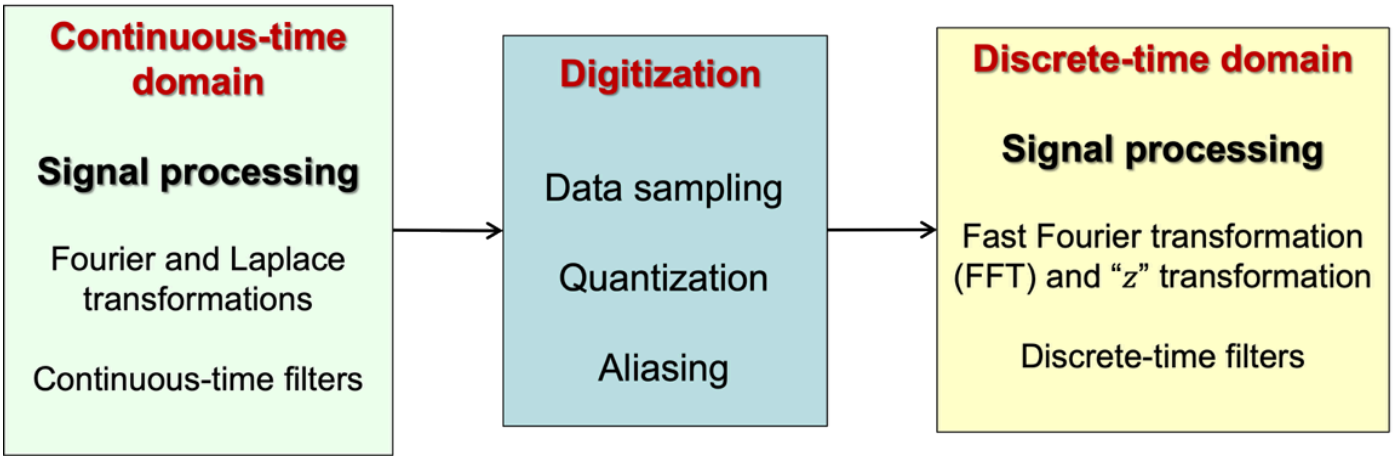


Figure 4.2: System excitation and associated response.

Section 4.3 focuses on the digital signals inside the information domain of a mechatronic system. They are often obtained by analog-to-digital (A/D) conversion of system sensor signals. Sometimes these signals come through a so-called sample and hold process. Other times, the signals are input directly through microcontroller (microprocessor) digital input channels either by counting the number of pulses or by recording the time duration between two pulses. The microcontroller of a mechatronic system processes these digital signals through filtering, signal conditioning, and generating control signals.

The digital signal can be analyzed in the frequency domain similar to the case of analog signals using the discrete Fourier transform [6], [7], [8], which requires a lot of computational time. In order to make the real-time frequency analysis possible, we often use the fast Fourier transform (FFT) [7]. This requires the signal length l equal to $l = 2^n$, where $n > 0$ is an integer. The FFT can reduce transform calculation throughput by more than 90% when n is large, compared with the conventional discrete Fourier transform. For the discrete Fourier transform, the highest frequency information is limited by the signal sample frequency, and the lowest frequency information is limited by the signal length l.

Analog and digital signals in physical and information domains, respectively, use analog and digital filters widely to get rid of the undesired information over certain frequency range contained in the signal. For instance, the low-pass filter is often used to get rid of high-frequency noise contained in the sensor signal. Meanwhile, the high-pass filter can be used to eliminate the DC bias in the signal, especially in the case integration of the signal is required. Band-pass and band-kill (notch) filters can also be used to allow the signal with a specific frequency range to pass the filter or to get rid of a specific frequency component.

Note that analog and digital filters [9], [10] have different characteristics. For certain applications (such as eliminating sampling aliasing), only analog filters can be used. Section 4.4 addresses designing both analog and digital filters, along with the realization of analog filter using electronic circuits. Digital filters are implemented in software as computational algorithms with no hardware required. Therefore, it is inexpensive to implement digital filters. For extremely fast application, FPGA (field programmable gate arrays) silicon is often used for implementing digital filters with certain cost [11].

During the conversion process of the analog signal in the physical domain to the digital signal in the information domain through sample and hold in the form of analog-to-digital (A/D) converter, it is both desired and important to retain the information contained in the analog signal once it is in the converted digital signal. Otherwise, the sampled signals are not useful in the information domain because of lost information. In addition, if the sample frequency is too low, the digital signal may contain information in these frequencies (called aliasing frequencies) that are not contained in the original analog signal. This can lead to errors not only in the sampled digital signals, but also, as a result, in the control signals generated by the microcontroller.

In Section 4.5, we introduce the Nyquist-Shannon theorem, which provides a sample frequency condition to avoid sample aliasing [12], [13]. The sample frequency should be at least two times higher than the highest frequency component contained in the analog signal. In order to avoid sample aliasing, after the sample frequency is selected, an analog anti-aliasing filter shall be designed and placed before the A/D converter. Here, the low-pass filter has a corner frequency below at least one half of the sample frequency [14].

Next, we discuss another class of signals in Section 4.6. Although these pulse-width modulated and frequency modulated signals are digital in nature, they represent analog signals in the physical domain. The pulse-width modulated signal is a square pulse train with fixed frequency but varying duty-cycle [15]. The frequency modulated signal [16] is a fixed duty-cycle signal (such as square waive or sinusoid signal) with varying frequency. Note that a frequency modulated sinusoid signal is often used to transmit

the radio signal.

For square-wave pulse-width modulated signals, we can convert the signal from physical domain into information domain by first counting the clock pulse activated by the square-wave. We then transfer the signal from the information domain to the physical domain by using a fixed frequency saw-wave and a comparator. Note that the pulse-width modulated signal is often used to control DC motors to reduce the energy consumption of the amplifying circuit.

Many advanced filter design methodologies can be used to improve signal quality; these are discussed in Section 4.7. Note that first and second order filters do not have a clean cutoff around the corner frequency. As a result, a certain amount of the unwanted signal (noise) beyond or below the corner frequency passes through the filter, which is undesirable. To improve filtering quality, high order filters such as Butterworth and Chebyshev filters [17] are often designed and implemented. This section addresses the design of both analog Butterworth and Chebyshev filters, along with its implementation hardware, especially for analog Butterworth and Chebyshev filters. We also discuss digital filter implementation with an example of moving average.

Orienting Questions

1. How do we conduct frequency domain analysis for both analog and digital signals?
2. What are the processes of analog-to-digital (A/D) and digital-to-analog (D/A) signal conversions?
3. How do we design analog low-pass, high-pass, band-pass, and band-kill (notch) filters?
4. What is the Nyquist-Shannon sample theory?
5. How do we select the sample and anti-aliasing filter frequencies to avoid sampling frequency aliasing?

This page titled 4.0: Prelude is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

4.1: Analog Signals and Their Frequency Analysis

Learning Objectives

After completing this section, you will be able to:

1. Define analog (continuous-time) signals.
2. Conduct analog frequency analysis using the Laplace transform.

This section discusses the analog signals in the physical domain and their frequency responses. Continuous-time signals (or analog signal) often represent the physical variables or parameters in the physical domain. For example, in a vehicle mechatronic system, the vehicle speed, acceleration, and cabin temperature are among the physical parameters. These variables can be converted into analog signals using speed, acceleration, and temperature sensors. On the other hand, analog control signals for electric motors or other actuators are generated by the microcontroller in the information domain. These are then converted into analog signals using D/A converters. In certain applications, such as pulse-width modulated signal (Section 4.6), digital signals can be amplified and used directly to drive a DC motor. Physically, the analog signal in a mechatronic system is often represented by a voltage signal $u(t)$ as a function of time or a current signal $i\,(t)$. Mathematically, both voltage $u(t)$ and current $i(t)$ signals can be represented by piecewise continuous functions of time t. The definition of a piecewise continuous function is shown below.

Definition 4.1 *[18]*

A function $f(t)$ is called piecewise continuous (or sectionally continuous) on an interval \(α

1. $f(t)$ is continuous on each subinterval \(t_0
2. On each subinterval $(t_{k-1}, t_{k1})\,(k = 1, 2, \ldots, n)$ both limits $\lim_{t\to t_{k-1}^{+}} f(t)$ and $\lim_{t\to t_{k}^{-}} f(t)$ exist for $k = 1, 2, \ldots, n-1$, i.e., these limits are finite.

Remark 4.1

Note that function $f(t)$ does not need to be defined at the points $t_k,\ \ (k = 1,\ 2,\ \ldots,\ n-1)$. In addition, if function $f\,(t)$ is defined on an interval $\alpha \le t < \beta$, then $\lim_{t\to\alpha^{+}} f(t)$ exists and $f\,(\alpha)$ is finite. For $f\,(t)$ defined on an interval \(α

Table 4.1: Typical excitation functions and their Laplace transforms

Description	$f\,(t)\,,\ \ t \ge 0$	$F\,(s) \triangleq L\,[f\,(t)]$
Unit impulse	$\delta\,(t)$	1
Unit Step	$u\,(t)$	$\dfrac{1}{s}$
Unit ramp	t	$\dfrac{1}{s^2}$
Sinusoid	$\sin\omega_0 t$	$\dfrac{\omega_0}{s^2+\omega_0^2}$

Table 4.1 considers continuous-time (analog) signals that are used to study system characteristics such as excitation functions. You can see the table summarizes their piecewise-continuous properties at time 0 and frequency responses. Let's start with the unit impulse input $\delta\,(t)$. The unit impulse function $\delta(t)$ is defined in equation 4.1.2 and its Laplace transform is 1, as indicated in Table 4.1.

$$\delta\,(t) = \begin{cases} \infty, & t = 0 \\ 0, & t \ne 0 \end{cases} \tag{4.1.1}$$

and

$$\int_0^\infty \delta(t)\,dt = 1 \tag{4.1.2}$$

The time domain unit impulse function was discussed in Chapter 3. Here, we are looking into its frequency response. Note that the Laplace transform of the unit impulse $\delta(t)$ is 1 (a real number) and, as a result, its magnitude will be 1 and phase is 0. Figure 4.3 (generated using the MATLAB command bode) shows this finding. Note that the unit impulse function $\delta(t)$ is an idealized continuous-time signal. It cannot be realized physically due to its infinity function value at time zero. Especially by Definition 4.1 and Remark 4.1, the unit impulse function $\delta(t)$ is not piecewise continuous; at its boundary of time equal to zero, the associated function value is unbounded.

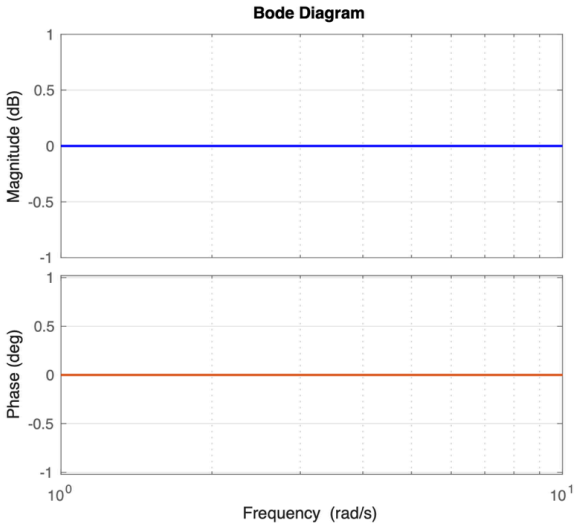


Figure 4.3: Frequency response of the unit impulse $\delta(t)$.

Now let's consider two continuous-time (analog) excitation signals: the unit step and unit ramp functions defined in Table 4.1. Both functions are defined over $t \in [0,\ \infty)$ and continuous over $t \in (0,\ \infty)$. Because the limit $\lim_{t\to 0^+} f(t)$ exists for both functions, they are both piecewise continuous based on Definition 4.1 and Remark 4.1. The unit step function $u(t)$ is defined as below and its Laplace transform is $1/s$ as indicated in Table 4.1.

$$u(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases} \tag{4.1.3}$$

The unit ramp function $r(t)$ is defined as below and its Laplace transform is $1/s^2$ as indicated in Table 4.1.

$$r(t) = \begin{cases} 0, & t < 0 \\ t, & t \geq 0 \end{cases} \tag{4.1.4}$$

Time domain unit step and ramp functions were discussed in Chapter 3. Here, we are looking mainly into their frequency responses. The Laplace transform of the unit step $u(t)$ is $1/s$ and unit ramp $1/s^2$. As a result, the magnitudes of unit step and ramp decrease at rates of -20 dB and -40 dB, respectively, with their phases remain at -90° and -180°, respectively. Those facts can be found in Figure 4.4 (generated using the MATLAB command bode). Both unit step and ramp functions are piecewise continuous.

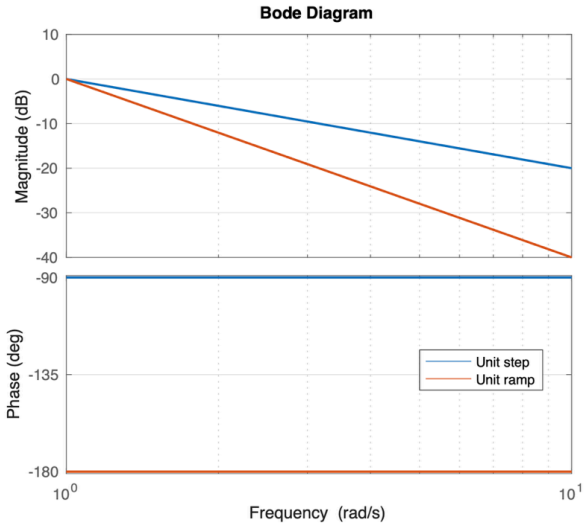


Figure 4.4: Frequency responses of unit step $u(t)$ and unit ramp $r(t)$.

Last, let's consider the sinusoidal continuous-time (analog) excitation signal, $sin(\omega_0 t)$, defined in Table 4.1. The sinusoidal function is defined over $t \in [0,\ \infty)$ and continuous over $t \in [0,\ \infty)$. Recall that the unit sinusoidal function $sin(\omega_0 t)$ and its Laplace transform are given in Table 4.1, while $\omega_0 = 1$ is provided below for convenience.

$$\mathcal{L}\left[\sin(t)\ \right] = \int_0^\infty e^{st}\sin(t)\ dt = \frac{1}{s^2+1} \tag{4.1.5}$$

The time domain unit sinusoidal function was also discussed in Chapter 3. Here, we are mainly looking into its frequency responses. Note that the Laplace transform of the unit sinusoidal function $\sin(t)$ is shown in equation 4.1.5. As a result, the magnitude of unit sinusoidal frequency response is shown below, along with its phase.

$$\left|\mathcal{L}\left[\sin(t)\ \right]\right| = \left|\frac{1}{1-\omega^2}\right|, \quad \angle\left(\mathcal{L}\left[\sin(t)\ \right]\right) = -\tan^{-1}\left(\frac{1-\omega^2}{0}\right) \tag{4.1.6}$$

It can be seen that the magnitude has a discontinuity at $\omega = 1$ with infinity magnitude. See the top plot in Figure 4.5 for the case with natural frequency ω_0 equal to one. In general, for a sinusoidal function, its frequency response magnitude always has discontinuity at its natural frequency ω_0 with infinity magnitude. When the frequency ω is below its natural frequency ω_0, the magnitude will be close to one. When the frequency ω is above its natural frequency ω_0, the magnitude will decay at a rate of -40 dB (see the top plot below in Figure 4.5).

The phase of the frequency response can also be calculated using equation 4.1.6. When ω is below its natural frequency ω_0 , the phase is zero. When ω is above its natural frequency ω_0, the phase is -180°. As a result, the phase response also has a discontinuity at ω_0; see the bottom plot of Figure 4.5. Both plots are generated using the MATLAB command bode.

Example 4.1

Consider a summation of sinusoidal signals with different frequencies and amplitudes in equation 4.1.7.

$$f(t) = \sin(t)\ + 0.5\sin(4t)\ + 0.25\sin(3t) \tag{4.1.7}$$

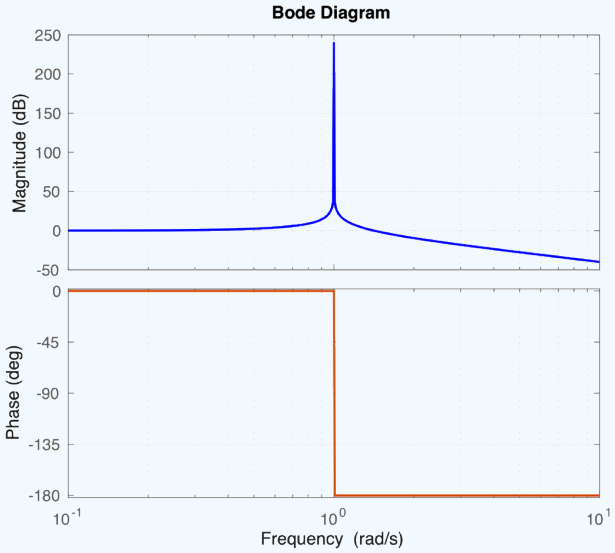


Figure 4.5: Frequency responses of the sinusoidal function $\sin(t)$.

The individual sinusoidal analog signals can be plotted in Figure 4.6. We can see three sinusoidal signals with frequencies of 1, 4, and 3 rad/s, respectively. The unit of the oscillation frequencies ω for sinusoidal function is in radian per second, not in Hertz (oscillation per second) represented by f. They can be related as below.

$$\omega = 2\pi f \tag{4.1.8}$$

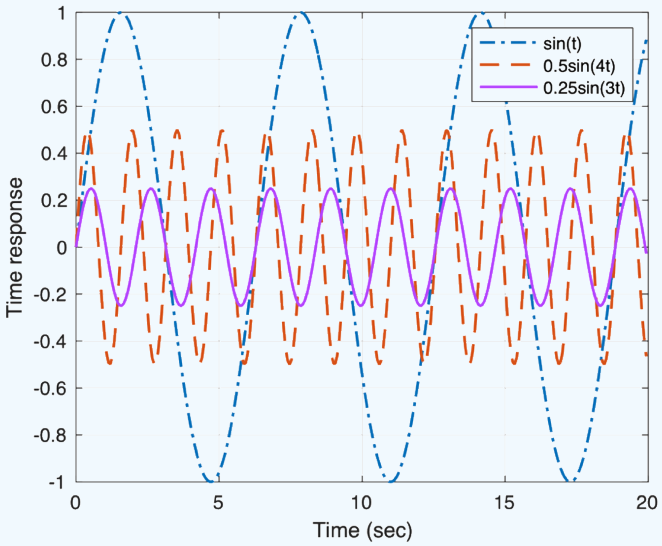


Figure 4.6: Sinusoidal time responses of $\sin(t)$, $0.5\sin(4t)$, and $0.25\sin(3t)$.

The weighted summation function $f(t)$ of $\sin(t)$, $0.5\sin(4t)$, and $0.25\sin(3t)$ is plotted in Figure 4.7. It is still a periodic function with many frequency components, which can be seen from its frequency response (magnitude) in Figure 4.8.

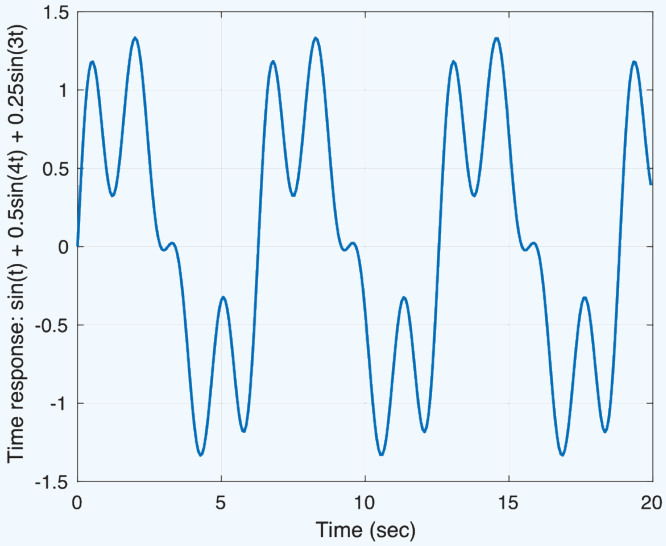


Figure 4.7: Summation sinusoidal time responses of $\sin(t)$, $0.5\sin(4t)$, and $0.25\sin(3t)$.

To plot the frequency response of $f(t)$, the Laplace transform is calculated as

$$\mathcal{L}\left[f(t)\right] = \frac{1}{s^2+1} + 0.5\frac{4}{s^2+16} + 0.25\frac{3}{s^2+9} \tag{4.1.9}$$

which can then be simplified into the following.

$$\mathcal{L}\left[f(t)\right] = \frac{3.75s^4 + 57.75s^2 + 174}{s^6 + 26s^4 + 169s^2 + 144} \tag{4.1.10}$$

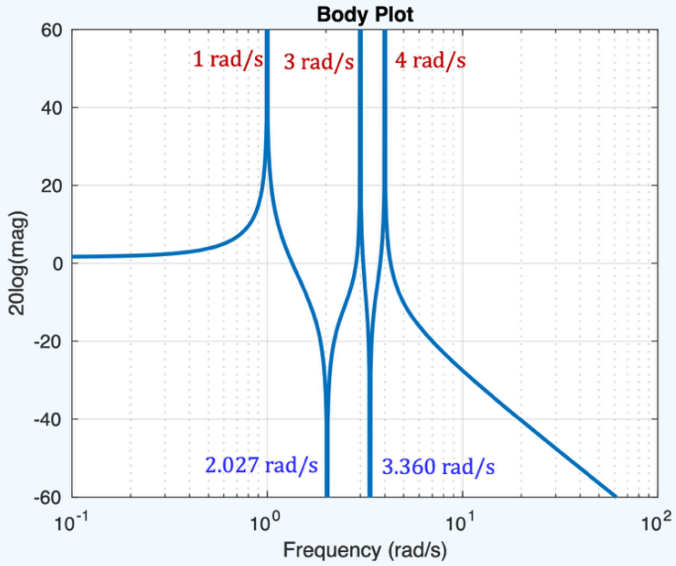


Figure 4.8: Frequency response (magnitude) of the summation of $sin(t)$, $0.5sin(4t)$, and $0.25sin(3t)$.

The frequency response magnitude is plotted in Figure 4.8. We can see three peaks in the frequency response responding to the three sinusoidal signal frequencies at 1, 3, and 4 rad/s (see red labels on Figure 4.8). These are due to the denominator's polynomial of $s^6 + 26s^4 + 169s^2 + 144$ in $\mathcal{L}\left[f(t)\right]$. In addition, there are two zero magnitudes at 2.027 and 3.360 rad/s (see blue

labels on Figure 4.8). These are due to the roots of the numerator's polynomial of $n(s) = 3.75s^4 + 57.75s^2 + 174$ in $\mathcal{L}[f(t)]$. Normally, we expect that magnitude peaks will occur at the frequencies that the signal contains these frequency components. However, the zero magnitude frequencies cannot be directly correlated.

Section Summary

This section addresses the analog signal in continuous-time physical domain and its frequency domain analysis in terms of the signal magnitude and phase. We use the Bode plot, introduced in Section 3.2, to plot the signal frequency responses (magnitude and phase). We also discuss the concepts of angular frequency in radian per second (rad/s) and oscillation frequency in Hertz (oscillation per second, Hz), as well as their relationship.

Section Keywords

Analog frequency response: This measure of how a dynamic system reproduces or processes signals of different frequencies can be calculated using the Laplace transform and can be presented in Bode plot.

Analog signal: This continuously variable physical measurement, such as voltage or current, is used to represent real-world phenomena like sound or temperature, differing from digital signals that use discrete binary values.

Bode plot: It provides two frequency response graphs of a system transfer function, which are a combination of a Bode magnitude plot with magnitude in decibels and a Bode phase plot with phase in degrees. For both plots, the horizontal axis of angular frequency is in logarithm of base 10.

Practice Problems

1. The analog signal inside a mechatronic system is located in which of the following domains? Select the correct answer.
    1. Physical domain
    2. Information domain
    3. Physical and information domains
2. A digital signal in the information domain can be obtained using which of the following devices of a microcontroller? Select the correct answer.
    1. Digital-to-analog converter
    2. Analog-to-digital converter
    3. Analog-to digital and digital-to-analog converters
3. An analog signal in the physical domain can be obtained using which of the following devices of a microcontroller? Select the correct answer.
    1. Digital-to-analog converter
    2. Analog-to-digital converter
    3. Analog-to digital and digital-to-analog converters
4. Consider the analog signal in the physical domain: $f(t) = \cos(t) + 0.5\sin(4t)$.
    1. Plot its time response using MATLAB.
    2. Find its Laplace transform.
    3. Plot its frequency response (magnitude only) using MATLAB.

This page titled 4.1: Analog Signals and Their Frequency Analysis is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)

Learning Objectives

After completing this section, you will be able to:

1. Describe the conversion process of analog-to-digital and digital-to-analog signals.
2. Calculate the quantization error in terms of the signal-to-noise ratio during the analog-to-digital conversion as a function of A/D conversion bits.
3. Describe the hardware (conversion circuits) for A/D and D/A converters.
4. Define binary and decimal representations of a digital number in the information domain. Then conduct the associated conversion between the two representations.

From Figure 4.1, a mechatronic system consists of two domains: physical and information domains. The physical domain is where the physical system hardware is located. Signals inside the physical domain are usually analog. The information domain is inside the microprocessor (or microcontroller) to process the digital signals and send the control signals to the physical domain. The mechatronic system control law is normally implemented into digital devices such as a microcontroller equipped with microprocessors, and both analog-to-digital (A/D) and digital-to-analog (D/A) converters. The microcontroller connects the discrete-time information domain to the continuous-time physical domain using A/D and D/A converters and other mechanisms that will be discussed in Section 4.6.

Note that A/D and D/A converters are connected to physical sensors and actuators, respectively. For example, a servo-motor mechatronic system consists of physical domain components (including a DC motor, gear system, and position sensor) and information domain components (including mainly the microcontroller that converts the analog sensor signal into a digital one). The system generates the corresponding digital control signal and converts the digital control signal to an analog one. This section mainly addresses the conversion from analog to digital and from digital to analog signals. Figure 4.9 shows the detailed processes of A/D and D/A conversion located within the information domain seen earlier in Figure 4.1.

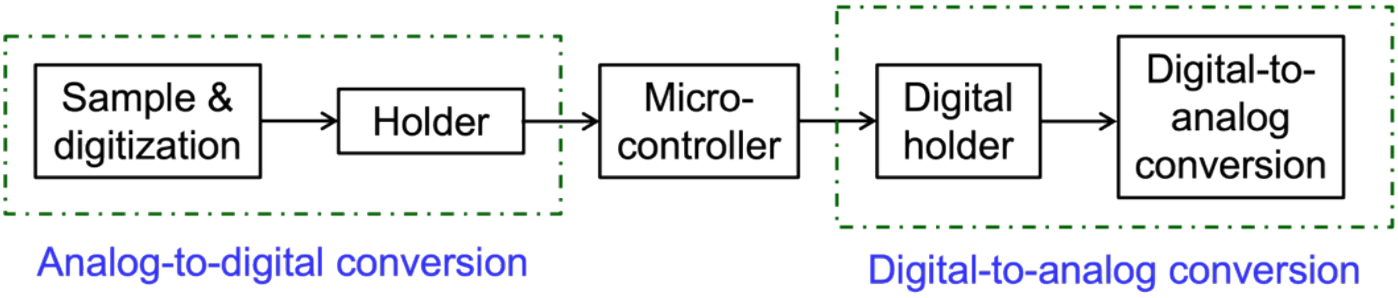


Figure 4.9: Mechatronic system signal conversion process.

Now, let's dive into the A/D conversion process. It can be divided into three steps as shown in Figure 4.10: sample, digitization, and hold. In Figure 4.10 (a), the red curve represents the analog signal that is normally a DC voltage signal. To convert the analog signal to the digital one, a sample period T in second needs to be determined. (Section 4.5 will show how to determine a proper sample period.) The green dots in Figure 4.10 (a) is the sampled DC voltage.

In a microprocessor, a digital number has a finite number of bits. Therefore, the sampled analog signal needs to be approximated by its nearest digital number. Figure 4.10 (b) shows a digitization process for a 3-bit binary number (see Table 4.2 for details), where the smallest binary number is 000 and the largest binary number is 111. Within the microprocessor, all numbers are represented using binary number; this will be discussed later in this section. In Figure 4.10 (b), the sampled data points (green dots) are approximated by the nearest digital number (red dots) below the sampled data points. Because the digital data will be updated only every sampled points (0, T, $2T$, $\cdots$), a sample hold will be used to represent the signal between any two nearby sample points, as demonstrated in Figure 4.10 (c) with the solid red line.

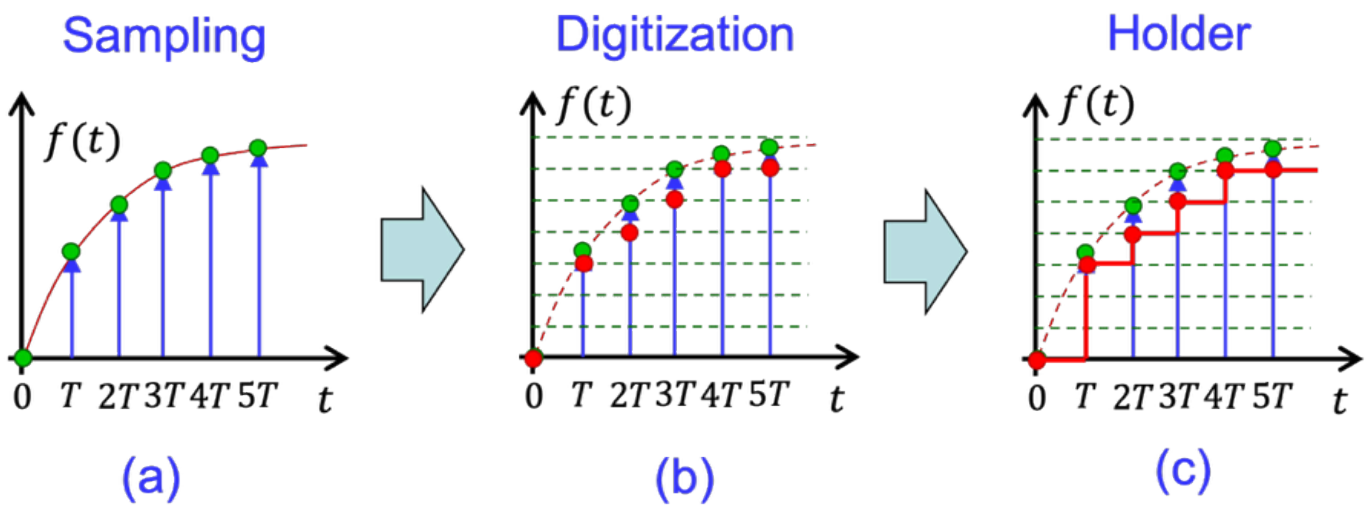


Figure 4.10: Analog-to-digital conversion process.

Mathematically, the analog-to-digital conversion can be represented using the following steps. First, the sampling process can be shown below, where $f(t)$ is the analog function to be converted (sampled), T is the sample period in second, k is the sample index (or step) in discrete-time domain similar to t in continuous-time domain with $t = kT$), and $f_s(k)$ is the sampled signal (as shown in Figure 4.10 (a) as green dots).

$$f_s(k) = f(kT) \tag{4.2.1}$$

After the signal is sampled, the digitization process can be expressed by the following formula, where $\mathcal{D}[\cdot]$ is a digitization operator that approximates the sampled analog signal $f_s(k)$ by its nearest digital number $f_d(k)$ (shown in Figure 4.10 (b) as red dots).

$$f_d(k) = \mathcal{D}[f_s(k)] \tag{4.2.2}$$

Next, we will use an example to demonstrate the digitization process below.

Table 4.2: Analog voltage and binary digital number relationship for Example 4.2.

Digital binary #	Analog voltage
000	[0.0, 0.0625)
001	[0.0625, 0.1875)
010	[0.1875, 0.3125)
011	[0.3125, 0.4375)
100	[0.4375, 0.5625)
101	[0.5625, 0.6875)
110	[0.6875, 0.8125)
111	[0.8125, 0.9375)

Example 4.2

This example will demonstrate how the digitization (or quantization) is completed in the process of analog-to-digital conversion. First, let's define the desired relationship between the input analog signal in voltage and the output digital binary number in Table 4.2, where conventional mathematical notations are used. That is, the low limit is closed with a square

bracket "[" and the upper limit is open with a rounded bracket ")". Table 4.2 indicates the corresponding binary digital number for a given analog voltage range. For example, if the input analog voltage is 0.35 volt, the corresponding binary output is 011, based upon the fourth data row in the table.

Figure 4.11 shows the conversion process. The purple curve represents the given analog signal. The green dots are the sampled analog signal with a sample period of T second (note that the signal is sampled at $0, T, 2T, \ldots$). In addition, the red dots are the binary approximations of the sampled signal (green dots) calculated based on Table 4.2.

The quantization (or digitization) error is indicated in Figure 4.11 by the pair of blue arrows. The horizontal axis is the time in seconds whereas the vertical axis is the input analog signal in volts. The red text box at the bottom shows the discretized binary number for each sample point. From Figure 4.11, it can be found that the maximum quantization error is 0.0625 volt. Note that as the number of A/D digits increases, the A/D conversion error decreases.

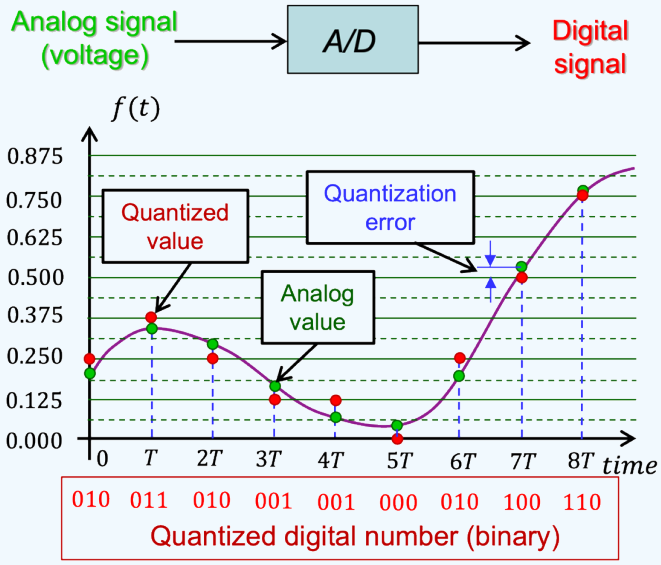


Figure 4.11: Analog-to-digital conversion diagram for a given signal.

For most of the applications, 10- and 12-bit A/D converters are widely used; for certain high precision applications, 16-bit A/D may be used instead. In general, selecting the number of bits for an A/D converter is highly dependent on the allowed conversion error and cost. For example, for a 12-bit A/D converter with 5-volt reference voltage, the conversion error is $5 \times 2^{-12} = 0.00122$ volt. In this case, the sensor signal noise level should be at a level similar to the conversion error in order to avoid affecting converted sensor signal accuracy.

We have discussed in principle how an analog signal is converted into a digital one in three steps; see Figure 4.10. Now, we will address the A/D converter hardware used to implement this three-step conversion process. Figure 4.12 shows a fast analog-to-digital 2-bit conversion circuit using four comparators. Note that the resistors and comparators are analog circuits whereas the decoder is digital. When the input signal V_{in} is greater than the reference input (e.g., $0.5 \times V_{ref}$), the corresponding bit (in this case, a_1) will be set to 1; otherwise, it will be set to 0.

On the digital side, a program will convert the converted bits ($a_3a_2a_1a_0$) into the 2-bit binary number (b_1b_0). This is a relatively expensive conversion circuit because a lot of comparators are required. For example, for a 12-bit A/D convertor, $2^{12} = 4096$ comparators would be needed. The advantage of this type of A/C conversion circuit is its fast conversion speed.

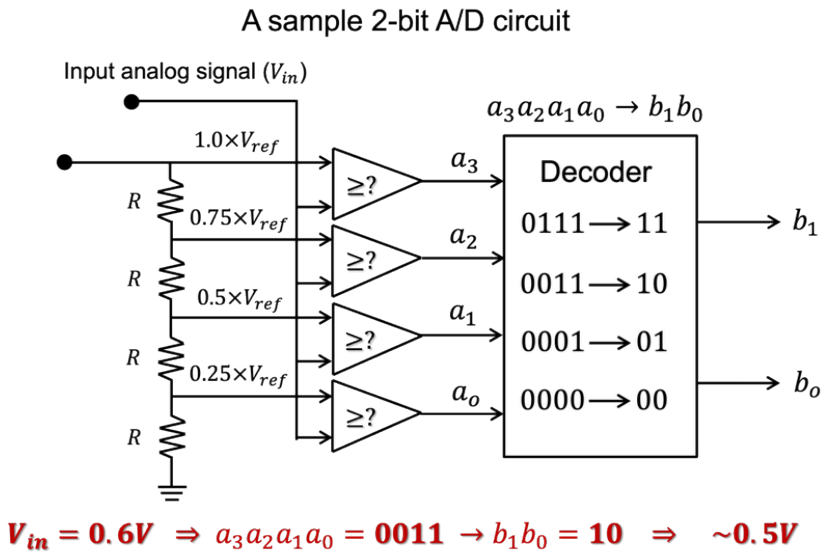


Figure 4.12: Example of realizing analog-to-digital (A/D) conversion.

Now we will discuss the digital-to-analog conversion process. Figure 4.13 shows a two-step digital-to-analog conversion process. First, the digital signal stored in the microprocessor is truncated to the proper bits that matches the D/A hardware. For example, if the microprocessor uses 32-bit floating digital numbers with a 10-bit D/A converter, these 32-bit floating numbers will be scaled, converted to a fixed-point binary number if needed (to be discussed later in this section), and truncated to match the number of bits for the D/A converter (10 bits in this case). Be aware that this truncation process introduces the D/A conversion error. Next, the fixed-bit digital number is sent to the D/A hardware to convert it to an analog signal (voltage). The D/A converter will hold the converted analog signal until the truncated digital number is updated. That is the reason that sometimes we call it a hold.

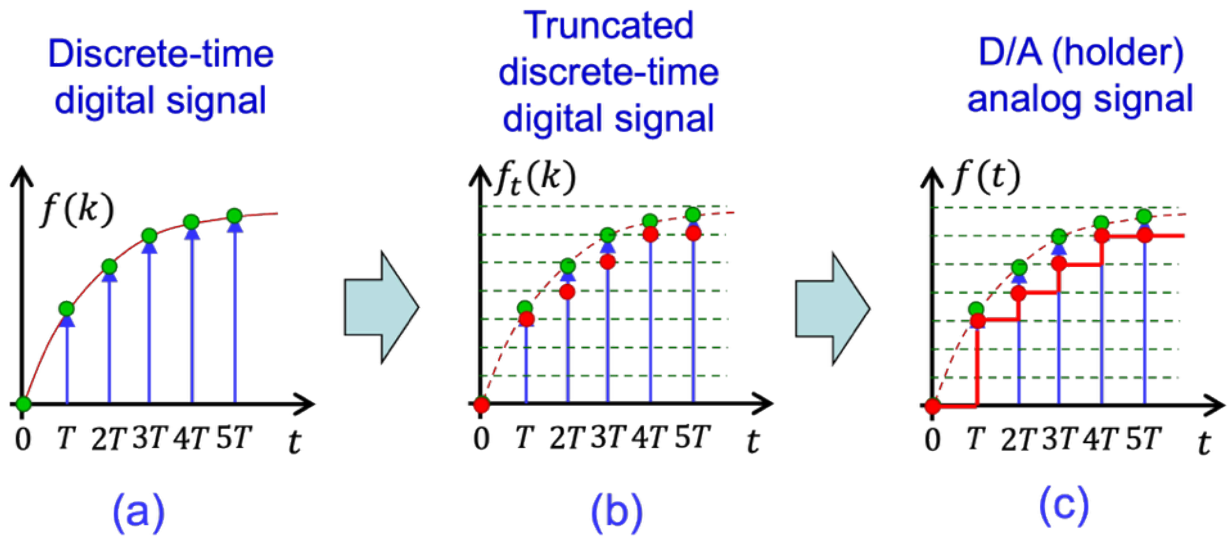


Figure 4.13: Digital-to-analog (D/A) conversion process.

Now that we have discussed in principle how a digital signal is converted into an analog one, let's address the D/A converter hardware used to implement the two-step conversion process. Figure 4.14 shows a 4-bit digital-to-analog conversion circuit using only one operational amplifier and a few resistors.

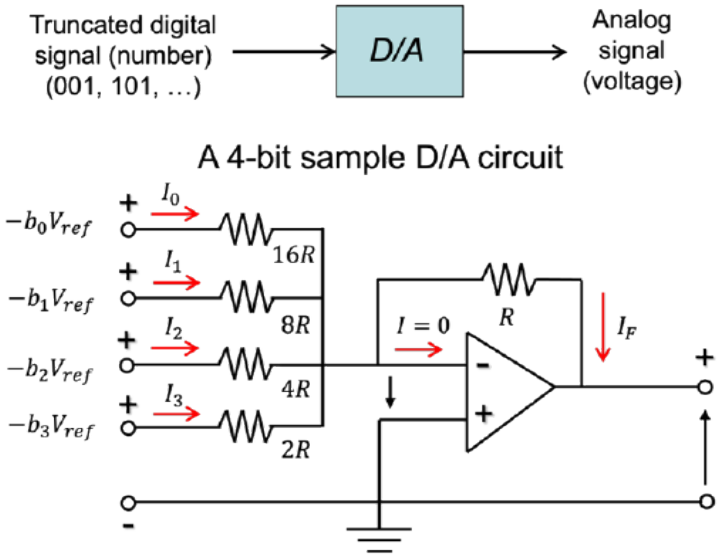


Figure 4.14: Example of realizing a 4-bit digital-to-analog (D/A) conversion.

When b_i $(i = 0,\ 1,\ 2,\ 3)$ is one, $-b_iV_{ref} = -V_{ref}$ so that there is a negative reference voltage of V_{ref} input to the circuit. This results in a current I_i, where $b_3b_2b_1b_0$ is the 4-bit binary number to be converted into the analog voltage. With the selected resistances in the circuit shown in Figure 4.14, we can have the current relationship shown in equation 4.2.3, based on the operational amplifier circuit review conducted in Chapter 2.

$$I_F = I_3 + I_2 + I_1 + I_0 = -b_3\frac{V_{ref}}{2R} - b_2\frac{V_{ref}}{4R} - b_1\frac{V_{ref}}{8R} - b_0\frac{V_{ref}}{16R} \tag{4.2.3}$$

As a result, we can see the output analog voltage of the operational amplifier in Figure 4.14.

$$V_{out} = -I_FR = b_3\frac{V_{ref}}{2} + b_2\frac{V_{ref}}{4} + b_1\frac{V_{ref}}{8} + b_0\frac{V_{ref}}{16} = V_{ref}\left(0.5b_3{+}0.25b_2 + 0.125b_1 + 0.0625b_0\right) \tag{4.2.4}$$

Example 4.3

Let $b_3b_2b_1b_0$ be a given 4-bit binary number ranging from 0 to 0.9375, where b_i is either 1 or 0 for i =0, 1, 2, 3. That is,

$$(b_3b_2b_1b_0)_{binary} = \left(b_32^{-1} + b_22^{-2} + b_12^{-3} + b_02^{-4}\right)_{decimal} \tag{4.2.5}$$

Therefore, if $b_3b_2b_1b_0 = 1111$, the analog voltage output of the operational amplifier is $0.9375V_{ref}$ based on equation 4.2.5. If $b_3b_2b_1b_0 = 1011$, the resulting analog output voltage is $0.6875V_{ref}$ based on equation 4.2.5.

In Figure 4.12, a fast A/D conversion circuit is presented with relatively high hardware cost because thousands of comparators are needed for a 12-bit A/D converter. Now, we will present a low-cost A/D conversion circuit using the D/A converter seen in Figure 4.14. Figure 4.15 shows a 4-bit low-cost A/D conversion circuit where one operational amplifier and one comparator are used along with a few analog switches and a digital counter.

Note that a 4-bit D/A conversion circuit is in the middle section of the conversion circuit. The A/D conversion process starts by resetting the count to zero. Because the input analog signal V_{in} is greater than zero, the comparator output is high (or 1); the operational amplifier output is zero volt due to resetting the counter. Therefore, after resetting, the counter will count up due to enabling counting from the comparator output as well as clock input. As the counted number increases, these analog switches will be closed with $b_i = 1$ $(i = 0,\ 1,\ 2,\ 3)$ and the operational amplifier output gradually increases.

The counting will stop when the comparator output becomes low, i.e., the operational amplifier output is greater than the input voltage V_{in}. In this case, the stopped counting number is the A/D converted binary digital number $b_3b_2b_1b_0$. The advantage of this A/D circuit is its low cost. However, the disadvantage is its slowness because we have to wait the counted number to increase in order to make the operational amplifier output match with the analog input. For a 12-bit A/D converter, this could take 4096

clock pulses, which could take some time. Meanwhile, the A/D conversion in Figure 4.12 is almost instantaneous.

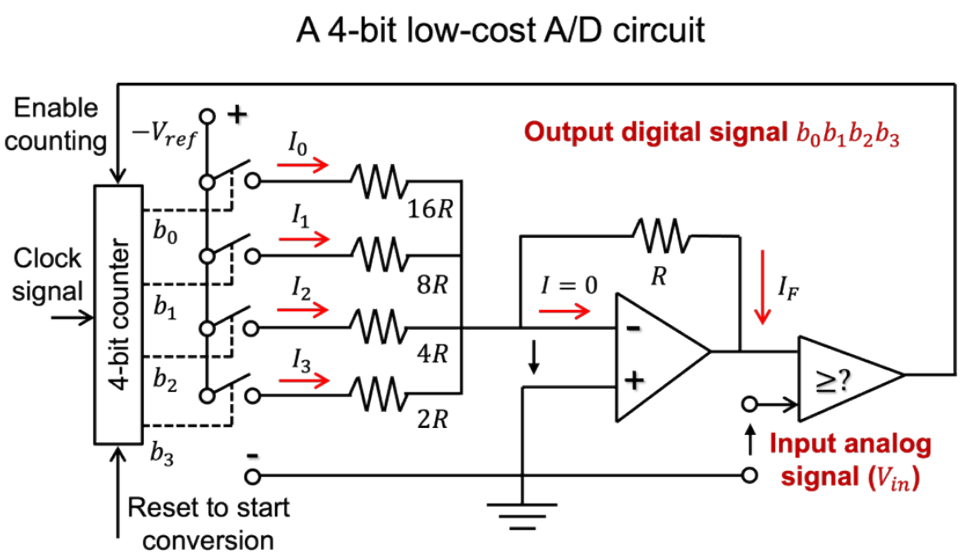


Figure 4.15: A 4-bit low-cost A/D conversion circuit.

Computers and digital devices represent and store information in a binary number format. The smallest unit is called a bit (binary digit) which can be represented by only two states: 0 or 1. Many bits together represent more states. There are two main reasons to use binary number for representation and arithmetic:

1. A transistor is the basic building block of nearly all current digital technologies. It is designed to act like a switch having two distinct states (0 and 1: on and off).
2. Semiconductor memory (such as RAM—random access memory), magnetic disks, and optical disks store only one of two possible states (0 or 1) at each physical location on the device.

An N-bit binary number can represent 2^N states; the stored integer is between 0 and $2^N - 1$.

Next, let's look at converting a number between its binary and decimal forms. This process is relatively simple compared with the reversed process. The general method of converting an 8-bit binary number $b_7b_6b_5b_4b_3b_2b_1b_0$ (where b_i is either zero or "1") is stated in the following equation.

$$(b_7b_6b_5b_4b_3b_2b_1b_0)_{binary} = \left(\sum_{i=0}^{7} b_i 2^i\right)_{decimal} \tag{4.2.6}$$

For example, if $b_7b_6b_5b_4b_3b_2b_1b_0 = 11010101$,

$$(11010101)_{binary} = \begin{pmatrix} 1\times 2^7 + 1\times 2^6 + 0\times 2^5 + 1\times 2^4 + 0\times 2^3 \\ +1\times 2^2 + 0\times 2^1 + 1\times 2^0 \end{pmatrix}_{decimal} = 213 \tag{4.2.7}$$

The conversion process from a decimal number to a binary one is more complicated than the process of converting from binary to decimal. Figure 4.1 illustrates the process of converting the decimal number 213 to its binary equivalent. We first compare 213 with 2^7 (or 128) associated with the highest bit. If it is greater than 128, bit b_7 is set to one and 128 is subtracted from 213. The resulting number of 85 is then compared with 2^6 (or 64). Because 85 is greater than 64, bit b_6 is set to one and 64 is subtracted from 85. The remainder 21 is smaller than 32, so bit b_5 is set to zero. In turn, 21 is used as the remaining number for the next bit. This process will be repeated through the last bit of b_0, resulting in a binary number of 11010101. If this process is followed correctly, it will end with zero remaining

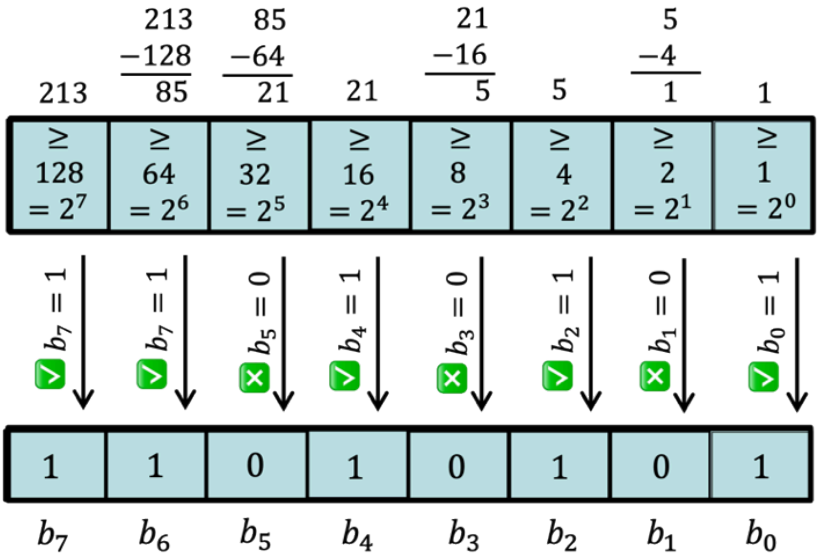


Figure 4.16: Decimal to binary number conversion process.

Note that the decimal number 213 can also be factored into the following summation weighted by binary power of 2^i ($i = 7,\ 6,\ \ldots,\ 0$).

$$213 = 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \tag{4.2.8}$$

The conversion process we've discussed is for integers. Now, the question is how to express a number with a fraction. Figure 4.17 shows how by moving the fraction point from the right-most point to the middle (blue dot). Those bits on the left of the blue dot represent the integer part and those on the right of the blue dot represent the fraction. The number converted in Figure 4.17 has the relationship shown in the equation below.

$$\begin{aligned}(01101.101)_2 \quad &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ &= (13.625)_{decimal}\end{aligned} \tag{4.2.9}$$

Here, the decimal number 13.625 is expressed by a summation of weighted binary power factors. The calculation method to obtain these weighting coefficients (binary expression) is the same as the one we used in Figure 4.16.

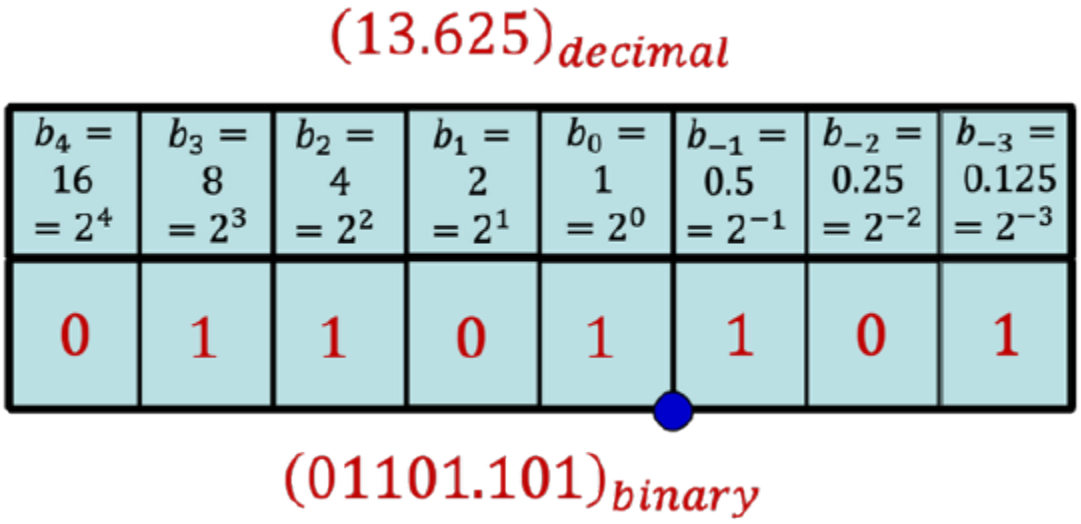


Figure 4.17: Fixed-point binary fractional number format.

Low-cost microcontrollers often use the fixed-point fraction expression (see Figure 4.17) because it is simple, uses less memory, and can be connected with A/D and D/A converters directly with a scaling factor. However, the disadvantage of using fixed point

format is its high digitization error, especially for small numbers. For example, given the decimal number 0.625234, the fixed-point fraction expression in Figure 4.17 is 0.625 (00000.101 binary), leading to word length error of 0.000234.

In order to reduce the transaction error based on a digital expression's word length, a floating-point fraction expression is often used in relatively high-end microcontrollers. The commonly used floating-point formats are 32-bit single precision and 64-bit double precision arithmetic; see [19] for IEEE 754-2018 standard. The 32-bit floating-point format uses 4 bytes (one byte containing 8 bits), where the first bit represents the sign of the number, the next 8 bits are for the exponent, and the remaining 23 bits are for the significand. For the double precision format (32-bit floating-point format), there are 11 bits for the exponent and 52 bits for the significand.

Next, we're going to concentrate on the 32-bit single precision format. The 64-bit double precision case is similar to the single precision case, but with different numbers of bits for exponent and significand. For the 32-bit single precision case, the first bit represents the sign of the number, where 0 is positive and 1 is negative. The 8-bit exponent from bits 2 to 9 (see Figure 4.18) is not a power of ten, but powers of two. That is, 8-bit stored exponents can range from -127 to 127, stored as 0 to 254, where 127 represents zero exponent of 2. Note that the value 2^{127} is roughly equivalent to 10^{38}, which is the actual limit of a single-precision number. The significand from bits 10 to 32 is stored as a binary fraction of the form $1.\times\times\times\times\cdots$.

This fraction has a value greater than or equal to 1 and less than 2. Real numbers are always stored in its normalized form. That is, the significand is left-shifted such that the high-order bit of the significand is always 1. Because this bit is always 1, we assume it is in the single-precision and double-precision formats implicitly. The binary point (not the decimal one) is assumed to be just to the right of the leading 1. That is, only the "$\times\times\times\cdots$" part is represented in bits 10 to 32. Example 4.4 and Figure 4.18 provide examples for the 32-bit single precision format.

	Bit 1	Bits 2-9	Bits 10-32
		$2^7 2^6 \cdots 2^1 2^0$	$2^{-1}2^{-2}2^{-3}2^{-4}2^{-5}2^{-6}2^{-7}2^{-8}\cdots 2^{-22}2^{-23}$
$(2 = 1.0\times2^1)_{decimal}$	0	10000000	00000000000000000000000
$(0.75 = 1.5\times2^{-1})_{decimal}$	0	01111110	10000000000000000000000
$(-13.625 = -1.703125\times2^3)_{decimal}$	1	10000010	10110100000000000000000

Figure 4.18: Floating point binary number format.

Example 4.4

The number 2.0 can be expressed as 1.0×2^1. Note that before proceeding to determine the number 2.0's single precision form, we need to factor it into a coefficient between 1 and 2 and multiply to an exponent of 2. The single precision form of the number 2.0 can be found in the second row of Figure 4.18. Note that bit 1 is zero because 2.0 is positive; bits 2–9 are 10000000 (its decimal form is 128) because 127 plus exponent 1 equals 128; and bits 10–32 are all zeros because the coefficient number (1.0) after 1 is zero.

Next, consider the number 0.75, which can be expressed as 1.5×2^{-1}. Note that 1.5 is between 1 and 2. The single precision form of number 0.75 can be found in the third row of Figure 4.18. Note that bit 1 is zero because 0.75 is positive. Bits 2–9 are 01111110 (its decimal form is 126) because 127 plus exponent -1 equals 126. Bits 11–32 are all zeros except bit 10, which is 1 because the coefficient number (1.5) after 1 is 0.5, which is equal to 2^{-1}.

Last, consider the number -13.625, which can be expressed as -1.703125×2^3. Note that 1.703125 is between 1 and 2. The single precision form of the number -13.625 can be found in the fourth row of Figure 4.18. Bit 1 is one because -13.625 is negative. Bits 2–9 are 10000010 (its decimal form is 130) because 127 plus exponent 3 equals 130. In turn, bits 10–15 are $.101101$ and bits 16–32 are all zeros. The decimal form of $.101101$ in this case is 0.703125 because it can be factored as follows.

$$\begin{aligned}(.101101)_2 &= 1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+1\times2^{-4}+0\times2^{-5}+1\times2^{-6}\\ &= (0.703125)_{decimal}\end{aligned} \tag{4.2.10}$$

In addition, 101.101 is the same as the last 6 binary digits for the fixed-point representation of 13.625.

There are also two special cases: zero and infinity. Note that the format discussed above cannot express both zero and infinity because we cannot express zero in the form of $1.\times\times\times\times\cdots$ and infinity with a finite exponent. As a result, zero will be expressed with exponent and significand parts equal to zero for all 31 bits; and the first bit can be either 0 or 1. Meanwhile, the plus infinity $(+\infty)$ and negative infinity $(-\infty)$ values are represented by an exponent of all ones with a significand of all zeroes. Positive and negative are represented by using the sign bit 0 for positive and 1 for negative.

Now, we will discuss the quantization error due to number of bits in the A/D converter. Note that quantization rounds to the closest discrete level. The accuracy of digitized signal depends on the number of bits used per sample. The error of quantization operation goes down as the number of bits increases. Using N bits to store each sample provides 2^N quantization levels. For example, 2 bits give 4 levels, 3 bits give 8 levels, 4 bits give 16 levels, 8 bits give 256 levels, 10 bits give 1024 levels, 12 bits give 4096 levels, and 16 bits give $65,536$ levels. Figure 4.19 shows the effect of the quantization error as a function of number of bits used to store the signal. We can see that the quantization error gradually reduces as the number of bits increases, where 3 bits show the largest error between red and blue curves and 5 bits have the smallest error between red and blue lines.

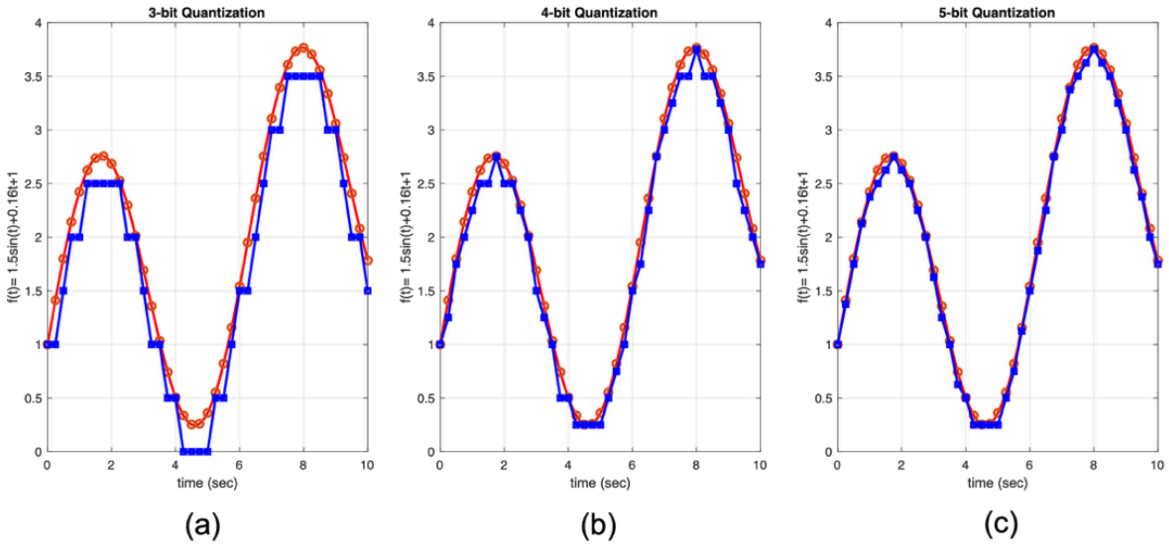


Figure 4.19: A/D number of bits and its quantization error.

The last content we will cover in this section is the concept of signal-to-noise ratio (SNR) due to signal quantization, which is defined below.

Definition 4.2

The signal-to-noise ratio (SNR) is defined as the magnitude ratio of the maximum signal level to the maximum noise (quantization error) level. The general form for SNR is defined for two cases. The first case is for positive numbers with N bits; the SNR can be calculated by the following formula,

$$SNR = \frac{2^N - 1}{2^{-1}} = 2^{N+1} - 2 \tag{4.2.11}$$

The second case is for positive and negative numbers with N bits; the SNR can be calculated by the following formula,

$$SNR = \frac{2^{N-1} - 1}{2^{-1}} = 2^N - 2 \tag{4.2.12}$$

Example 4.5

For Figure 4.20, calculate the corresponding signal-to-noise ratio (SNR) for positive and positive/negative cases.

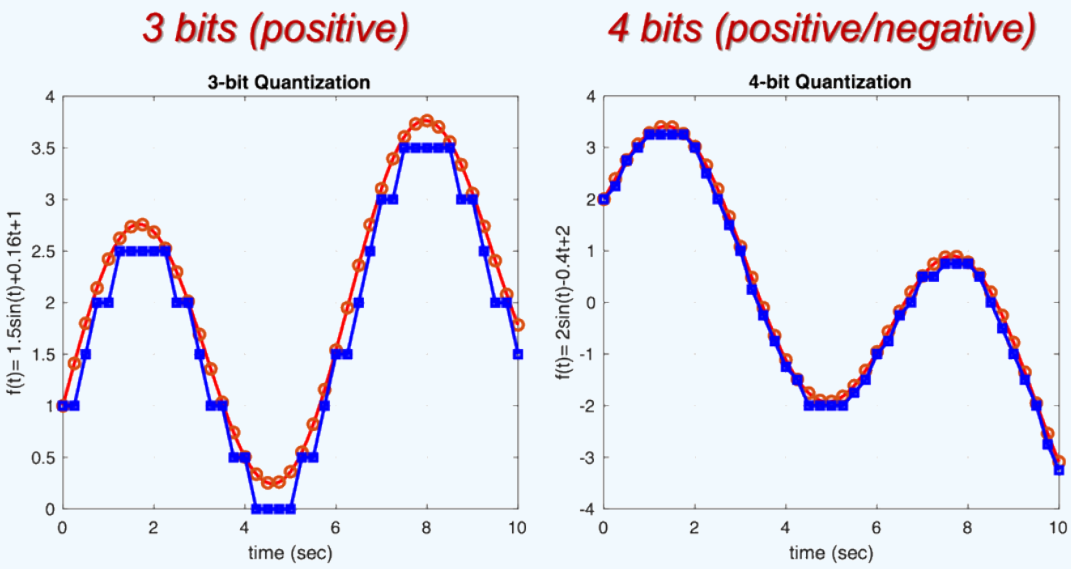


Figure 4.20: Positive and positive/negative signal-to-noise ratios.

For the positive case shown on the left plot of Figure 4.20, the number of bits $N = 3$ and the positive number signal-to-noise ratio ($SNR = 14$) can be calculated by the following equation.

$$SNR = \frac{2^N - 1}{2^{-1}} = \frac{7}{0.5} = 14 \tag{4.2.13}$$

For the positive/negative case shown on the right plot of Figure 4.20, the number of bits $N = 4$ and the positive/negative number signal-to-noise ratio ($SNR = 14$) can be calculated by the following equation.

$$SNR = \frac{2^{N-1} - 1}{2^{-1}} = \frac{7}{0.5} = 14 \tag{4.2.14}$$

Section Summary

This section discusses the analog-to-digital (A/D) and digital-to-analog (D/A) conversion processes. The electric circuits that perform these conversions are called analog-to-digital (A/D) and digital-to-analog (D/A) converters, respectively. Most A/D and D/A converters are fabricated as integrated circuits. They can be found at place where analog signals are to be sampled, quantized, or transmitted digitally, e.g., internet, cell phones, computers, and automobiles. Note that the quantization operation causes errors in representing digitized information. These errors are called quantization (or digitization) noises because the effect of quantization errors sounds like noise in a digitized music signal and looks like noises in a digital image. In general, the higher the A/D bits, the smaller the quantization error.

Section Keywords

Practice Problems

1. What is the signal-to-noise ratio of a 10-bit A/D converter due to quantization error (positive) with $N = 10$?

1. How many comparators are needed for an 8-bit the A/D conversion circuit shown in Figure 4.12?
    1. 32

2. 128
3. 256

1. An 8-bit digital number of 11110101 is sent to an 8-bit D/A converter with reference voltage of 5 volts. What is the resulting analog voltage output from the D/A converter?
   1. 3.256 volts
   2. 4.805 volts
   3. 6.256 volts

1. What is the 32-bit single precision representation of decimal number -4.625?
   1. 0 00000010 00101000...0
   2. 0 10000001 00101110...0
   3. 1 10000001 00101000...0

This page titled 4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions) is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

4.3: Digital Signals and Their Frequency Analysis

Learning Objectives

After completing this section, you will be able to:

1. Describe the relationship between analog and digital frequency responses.
2. Calculate a z-transform to obtain a discrete transfer function.
3. Distinguish the conventional discrete Fourier transform and the fast Fourier transform (FFT) for digital frequency analysis.

This section discusses the digital signals in the information domain and their frequency responses based on conventional discrete Fourier and fast Fourier transforms. In general, the discrete-time signals, or digital signals, often represent the physical variables or parameters in the information domain. For example, consider a vehicle's mechatronic system. The vehicle speed, acceleration, and cabin temperature signals, sampled by A/D converters, are information domain physical variables. Information domain variables can be converted into analog signals using D/A converters. Typically, a digital signal in a mechatronic system is represented by a discrete-time function as follows.

$$f(k), k = 0,\ 1,\ 2,\ 3,\ \ ... \tag{4.3.1}$$

There is no signal definition between sample points. Similar to the Laplace transform for the continuous-time signal $f(t)$ defined in Section 4.1, a z-transform [20], [21] is used for the discrete-time signals, where the z-transform is defined below for digital signal $f(k)$ ($k = 0,\ 1,\ 2,\ \ ...$).

$$F(z) \cong \mathcal{Z}[f(k)] = \sum_{k=0}^{\infty} f(k) z^{-k} \tag{4.3.2}$$

Here, z is a one-step advance operator with step size of T (sample period). Recall that for a continuous-time signal, the Laplace transform is used for frequency analysis and dynamic system modeling. In this case, the relationship between the Laplace transform and the z-transform is $z = e^{sT}$ (where T is the sample period); see Figure 4.21 for their relationship.

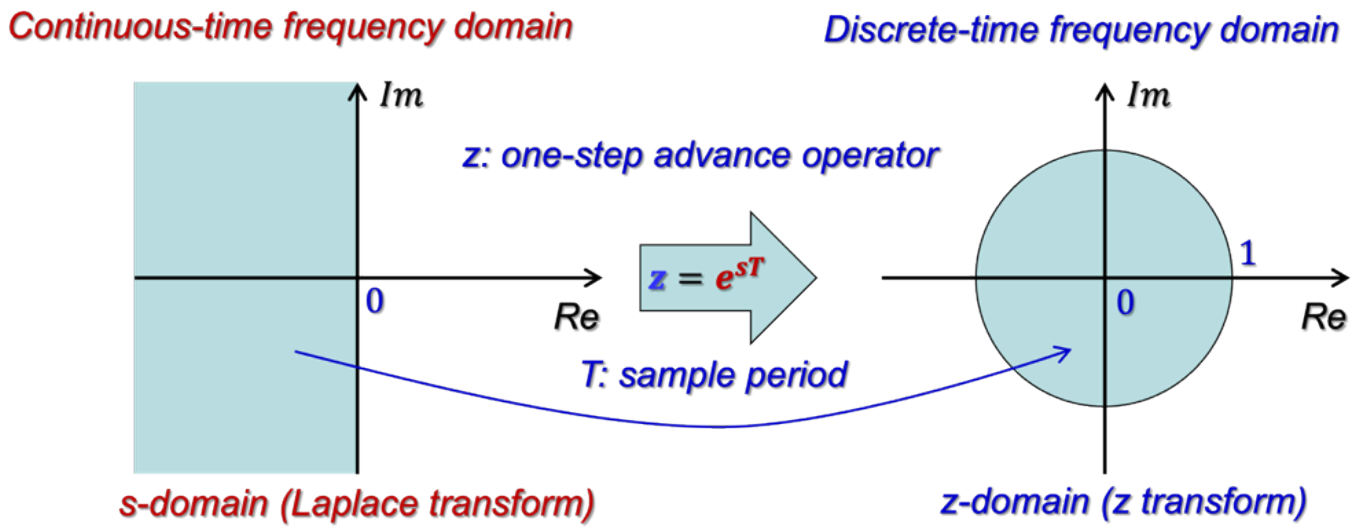


Figure 4.21: Relationship between Laplace transform and z -transform.

Note that for the Laplace transform, the continuous-time frequency domain is defined by "s" over the entire complex plan. If all system poles are located at the open left half plane (OLHP), the dynamic system is asymptotically stable. The discrete-time

frequency domain is defined by "z" over the entire complex plane. If all system poles are located within the unit circle of the complex plane, the dynamic system is asymptotically stable. This is due mainly to the fact that $z = e^{sT}$, which maps the open left half plane of "s" to inside the unit circle of the complex plane defined by "z". As a result, for a discrete-time system model described by z-transform, if the poles of the discrete-time transfer function in z-transform are all located inside the unit circuit, the discrete-time system is asymptotically stable; see [20] and [21] for details.

Next, we will present a few typical discrete-time signals, along with their z-transforms and their relationships to the Laplace transform, including unit pulse, unit step, unit ramp, and unit exponential (e^{kaT}) functions. First, the discrete-time unit pulse function is defined below.

$$\delta(k) = \begin{cases} 1, & k = 0 \\ 0, & k \neq 0 \end{cases} \tag{4.3.3}$$

By definition, the associated z-transform can be found below.

$$\mathcal{Z}[\delta(k)] = \sum_{k=0}^{\infty} \delta(k) z^{-k} = 1 \times z^{-0} = 1 \tag{4.3.4}$$

Next, the discrete-time unit step function is defined below.

$$u(k) = \begin{cases} 1, & k \geq 0 \\ 0, & k < 0 \end{cases} \tag{4.3.5}$$

and the associated z-transform can be found below.

$$\mathcal{Z}[u(k)] = \sum_{k=0}^{\infty} u(k) z^{-k} = \sum_{k=0}^{\infty} z^{-k} = \frac{z}{z-1} \tag{4.3.6}$$

Then, the discrete-time unit ramp function is defined below.

$$r(k) = \begin{cases} kT, & k \geq 0 \\ 0, & k < 0 \end{cases} \tag{4.3.7}$$

where T is the sample period, and the associated z-transform can be found below.

$$\mathcal{Z}[r(k)] = \sum_{k=0}^{\infty} r(k) z^{-k} = \sum_{k=0}^{\infty} kTz^{-k} = \frac{Tz}{(z-1)^2} \tag{4.3.8}$$

Last, the discrete-time unit exponential function is defined below.

$$e(k) = \begin{cases} \left(e^{-aT}\right)^k, & k \geq 0 \\ 0, & k < 0 \end{cases} \tag{4.3.9}$$

where T is the sample period and a is the continuous-time exponential coefficient. The associated z-transform can be found below.

$$\mathcal{Z}[e(k)] = \sum_{k=0}^{\infty} \left(\left(e^{aT}\right)^k z^{-k} = \sum_{k=0}^{\infty} kTz^{-k} = \frac{z}{z - e^{-aT}}\right. \tag{4.3.10}$$

The Laplace transform of continuous-time functions and their associated z-transform of discrete-time functions are listed in Table 4.3.

We will provide a few examples below to demonstrate how to use a z-transform to model a dynamic system in the information domain and solve the difference equation using the methodology similar to using Laplace transform in continuous-time case.

Example 4.6

For the dynamic system described by the following difference equation, find the discrete-time transfer function in the form of a z-transform.

$$y(k)+0.5y(k-1)+0.5y(k-2)=u(k)+0.5u(k-1) \tag{4.3.11}$$

Applying the z-transform leads to the following discrete-time transfer function.

$$Y(z)\left(1+0.5z^{-1}+0.5z^{-2}\right)=U(z)\left(1+0.5z^{-1}\right)\rightarrow \frac{Y(z)}{U(z)}=\frac{1+0.5z^{-1}}{1+0.5z^{-1}+0.5z^{-2}} \tag{4.3.12}$$

Table 4.3: Typical excitation functions and their Laplace and z transforms.

Laplace Transform $F(s)\triangleq L[f(t)]$	$f(t)\ \ (t\geq 0)$	$f(k)\quad (k=0,\ 1,\ 2,\ \ldots)$	**z-Transform** $F(z)\triangleq \mathcal{Z}[f(k)]$
1	$\delta(t)$	$\delta(k)$	1
$\frac{1}{s}$	$u(t)$	$u(t)$	$\frac{z}{z-1}$
$\frac{1}{s^2}$	t	kT	$\frac{Tz}{(z-1)^2}$
$\frac{1}{s+a}$	e^{-at}	e^{-akT}	$\frac{z}{z-e^{-aT}}$
$\frac{s+a}{(s+a)^2+\omega^2}$	$e^{-at}\cos\omega t$	$e^{-akT}\cos\omega kT$	$\frac{z^2-e^{-aT}\cos(\omega T\ )z}{z^2-2e^{-aT}\cos(\omega T)\ z+e^{-2aT}}$
$\frac{\omega}{(s+a)^2+\omega^2}$	$e^{-at}\sin\omega t$	$e^{-akT}\sin\omega kT$	$\frac{ze^{-aT}\sin(\omega T)}{z^2-2e^{-aT}\cos(\omega T)\ z+e^{-2aT}}$

Example 4.7

Assuming that the input to the system in Example 4.6 is a unit step, $U(z)=z/(z-1)$, find the time response $y(k)$ using inverse z-transform for the difference equation 4.3.11. From equation 4.3.12, we have the following.

$$Y(z)=\frac{z(z+0.5)}{z^2+0.5z^1+0.5}\cdot\frac{z}{z-1}=\frac{Az^2+Bz^1+C}{z^2+0.5z^1+0.5}+\frac{Dz}{z-1} \tag{4.3.13}$$

Making the numerators in equation 4.3.13 equal leads to the following equation.

$$\begin{aligned} z^3+0.5z^2&=\left(Az^2+Bz+C\right)(z-1)+Dz\left(z^2+0.5z+0.5\right)\\ &=(A+D)\,z^3+(-A+B+0.5D)\,z^2+(-B+C+0.5D)\,z-C\end{aligned} \tag{4.3.14}$$

Making the corresponding power coefficients equal yields the solutions for $A,\ B,\ C$, and D below.

$$A=0.25, B=0.375, C=0, D=0.75 \tag{4.3.15}$$

To convert the z-transform back to the time response using the conversion Table 4.3, the z-transform of $Y(z)$ can be reorganized as follows.

$$Y(z)=\frac{0.25\left(z^2-0.25z\right)}{z^2+0.5z+0.5}+\frac{0.4411\,(0.9353z)}{z^2+0.5z+0.5}+\frac{0.75z}{z-1} \tag{4.3.16}$$

Now, we need to determine aT and ωT (see Table 4.3). Note that $0.5=e^{-2aT}$. Therefore, $aT=-0.5\ln 0.5\ =0.3466$. Also note that $0.5=2\sqrt{0.5}\cos\omega T$ which leads to $\omega T=\cos^{-1}\sqrt{0.125}\ =1.209$. After applying the inverse z-transform using Table 4.3, we have the following discrete-time response.

$$y(k)=0.25e^{-0.3466k}\cos(1.209k)\ +0.4411e^{-0.3466k}\sin(1.209k)\ +0.75 \tag{4.3.17}$$

Example 4.8

This example will demonstrate how to obtain a discrete-time system model for a continuous-time system. Consider a mass-damper-spring system described by the following second order differential equation.

$$m\ddot{y}(t) + c\dot{y}(t) + ky(t) = u(t) \tag{4.3.18}$$

Here y, $\dot{y}$, and $\ddot{y}$ are displacement, speed, and acceleration, respectively; u is the applied force; and m, c, and k are mass, damping coefficient, and spring stiffness, respectively. Letting $x = [y,\ \dot{y}]^T$, we have the following state-space model.

$$\dot{x} = Ax + Bu,\ y = Cx \tag{4.3.19}$$

where

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix},\ B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix},\ C = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^T \tag{4.3.20}$$

Letting the system sample period be T, the corresponding discrete-time state-space model can be expressed as follows.

$$x(k+1) = A_d x(k) + B_d u(k), y(k) = C_d x(k) \tag{4.3.21}$$

where

$$A_d = e^{AT}, B_d = \int_0^T e^{A(T-\tau)} B d\tau, C_d = C \tag{4.3.22}$$

Applying the z-transform to equation 4.3.21 yields the following z transfer function model for the mass-damping-spring system.

$$Y(z) = C_d(zI - A_d)^{-1} B_d U(z). \tag{4.3.23}$$

This example shows that we can derive a continuous-time model in the form of differential equations, obtain a continuous-time state-space model, convert it into a discrete-time state-space model, and then apply the z-transform to obtain a discrete-time z transfer function for the system.

The discrete-time frequency domain analysis is often conducted based on the discrete Fourier transform. For a continuous-time transfer function, $s = j\omega$. For the discrete time transfer function, $z = e^{sT} = e^{j\omega T}$. In addition, for any integer k, $e^{j\omega T} = e^{j(\omega T + 2\pi k)}$. If $G(z) = G(e^{j\omega T})$ is used for calculating frequency response, the frequency magnitude and phase of $G(e^{j\omega T})$ will repeat every $2\pi/T$. Therefore, the z-transfer function $G(z)$ is mainly used for analyzing system asymptotical stability with all poles located within the unit circle. For the discrete-time periodic signal $x(k)$ ($k = 0,\ 1,\ 2,\ \ldots, N-1$), the discrete Fourier transform $X(k)$ is defined below for the given period N.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i\frac{2kn\pi}{N}},\ k = 0,\ 1,\ 2,\ \ldots,\ N-1 \tag{4.3.24}$$

where $W_N^k = e^{-i\frac{2kn\pi}{N}}$ ($k = 0,\ 1,\ 2,\ \ldots, N-1$) is called the k-th root of unity. Figure 4.22 shows the unity root location for $N = 2,\ 4$, and 8. We can see that W_N^k ($k = 0,\ 1,\ 2,\ \ldots, N-1$) are evenly distributed over the unit circle on the complex plane starting with $W_N^0 = 1$.

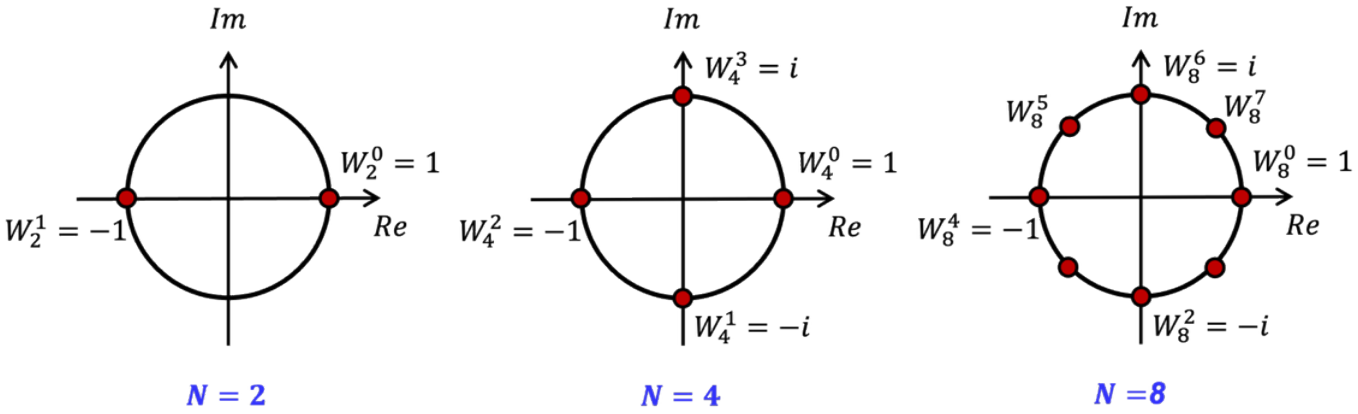


Figure 4.22: Roots of unity for $N = 2$, 4, and 8.

The inverse discrete Fourier transform is defined below.

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{i\frac{2kn\pi}{N}}, n = 0,\ 1,\ 2,\ \ldots,\ N-1 \tag{4.3.25}$$

From equations 4.3.24 and 4.3.25 we can see that for a time sequence of length N, its discrete Fourier transform also has length N. Due to the symmetric characteristics (equation 4.3.24), the discrete Fourier transform $X(k)$ is symmetric in a way shown below, assuming N is even.

$$X(k) = X(N-k), k = 1,\ 2,\ \ldots, N/2-1 \tag{4.3.26}$$

Therefore, only half of the discrete Fourier transform provides meaningful information. We will discuss the root cause of why only half of the discrete Fourier transform is useful in the next section. However, here we will use the following example to illustrate it. In addition, the frequency response of unit pulse, step, and ramp functions are similar to the continuous-time case in Section 4.1; we will not repeat them in this section.

Example 4.9

Consider the three sinusoidal functions shown below.

$$f_i(t) = \sin(2\pi\omega_i t), i = 1,\ 2,\ 3. \tag{4.3.27}$$

with

$$\omega_1 = 10, \omega_2 = 30, \omega_3 = 90. \tag{4.3.28}$$

The time responses of the three sinusoidal functions are plotted in Figure 4.23. We can see that these sinusoidal functions have frequencies at 10, 30, and 90 Hz.

Three N-point discrete-time sinusoidal signals were generated using the following formula, with $N = 1001$ and sample period of $T = 0.005$ second.

$$f_i(k) = \sin(2\pi(0.005k\omega_i), i = 1,\ 2,\ 3 \& k = 0,\ 1,\ 2,\ \ldots,\ N-1 \tag{4.3.29}$$

We can use the MATLAB command `fft(fi)` ($fi = f_i$) to conduct 1001-point discrete Fourier transforms of the three sinusoidal digital signals, resulting in the following 1001-point frequency responses.

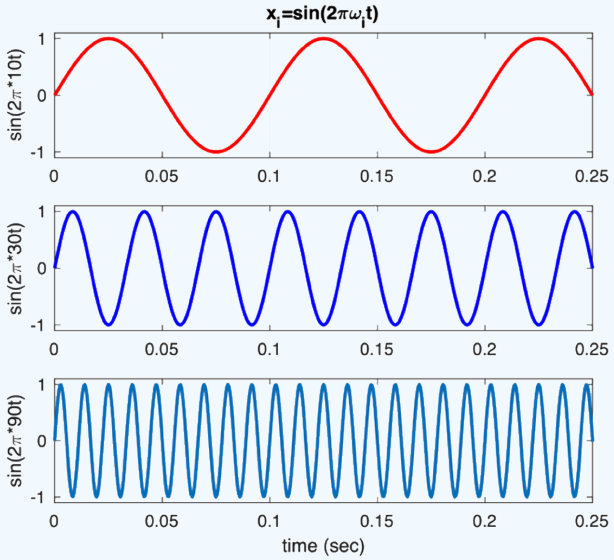


Figure 4.23: Time responses of $\sin(2\pi(10t))$, $\sin(2\pi(30t))$, and $\sin(2\pi(90t))$.

$$F_i(n) = \mathcal{F}\left[f_i\left(k\right)\right], i = 1,\ 2,\ 3 \text{and} n = 0,\ 1,\ 2,\ \ldots,\ N-1 \tag{4.3.30}$$

As discussed earlier, only half of frequency responses $F_i(n)$ $(n = 0,\ 1,\ 2,\ \ldots,\ (N-1)/2)$ are independent. The frequency magnitudes are plotted in Figure 4.24, using the first 501 points for the frequency response $F_i\left(n\right)$ $(n = 0,\ 1,\ 2 \ldots,\ (N-1)/2$), where the frequency vector of horizontal axis is calculated by the following equation.

$$x\left(k\right) = \frac{k}{T \times (N-1)}, k = 0,\ 1,\ 2,\ \ldots,\ (N-1)/2 \tag{4.3.31}$$

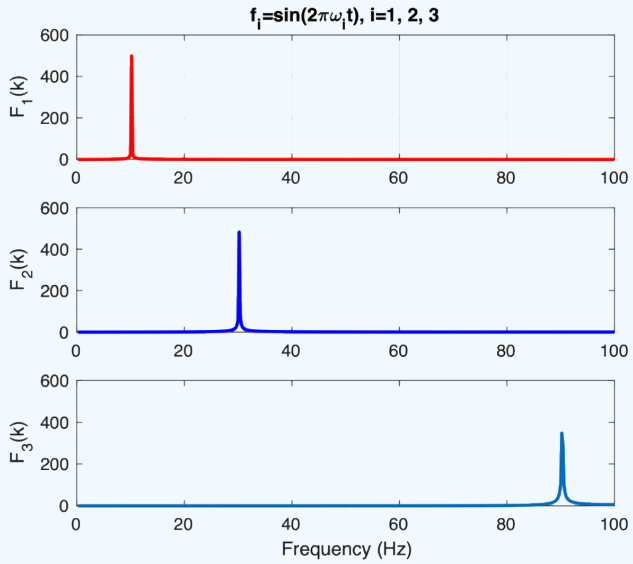


Figure 4.24: Frequency responses of the sinusoidal functions $\sin(2\pi\omega_i t)$ $(\omega_i = 10,\ 30,\ 90)$.

For each sinusoidal function, there is a dominated frequency peak. For the 10 Hz sinusoidal signal $\sin(2\pi 10t)$, the peak is located at 10 Hz; for $\sin(2\pi 30t)$, the peak is at 30 Hz; and for $\sin(2\pi 90t)$, it's located at 90 Hz. Therefore, the discrete Fourier transform provides frequency domain information for a time domain digital signal.

Now, let's combine the three sinusoidal functions, as shown below in the top plot of Figure 4.25.

$$\begin{aligned} f(k) &= f_1(k) + f_2(k) + f_3(k) \\ &= \sin(2\pi(0.005k10) + \sin(2\pi(0.005k30)) + \sin(2\pi(0.005k90)) \end{aligned} \tag{4.3.32}$$

A 1001-point discrete Fourier transform of the above combined sinusoidal digital signal is conducted using the MATLAB command `fft(f)`, resulting in a 1001-point frequency response $F(n)$. The frequency response magnitude of $F(n)$ can be found in the bottom plot of Figure 4.25.

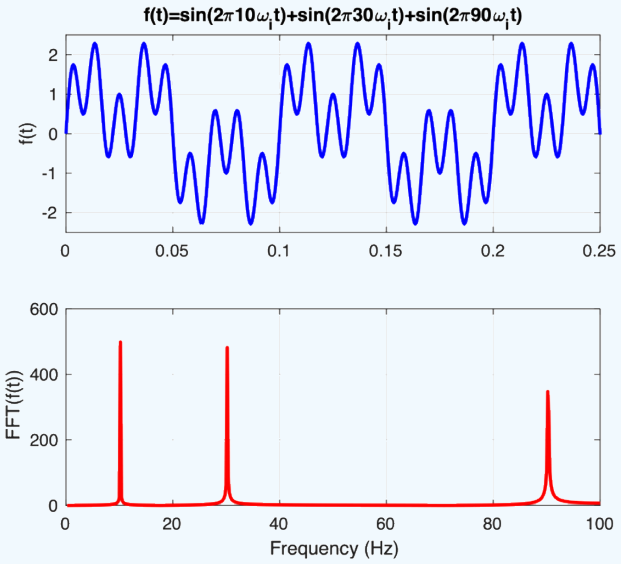


Figure 4.25: Combined sinusoidal function time and frequency responses.

From equations 4.3.24 and 4.3.25, it can be found that about N^2 multiplications are needed for either the discrete Fourier transform or the inverse discrete Fourier transform. Now let's consider the four-point ($N = 4$) discrete Fourier transform indicated below.

$$X(k) = \sum_{n=0}^{4-1} x(n) e^{-i\frac{kn\pi}{2}} = x(0) + (-i)^k x(1) + (-1)^k x(2) + (i)^k x(3) \tag{4.3.33}$$

We can see that we need 16 multiplications to complete the 4-point discrete Fourier transform. For this case, these multiplications are for 1, $-i$, -1, and i. However, for a large number of N, these multiplication coefficients could be any complex number on the unit circle; see Figure 4.22. We can expand and reorganize equation 4.3.33 as below for $k = 0,\ 1,\ 2,\ 3$.

$$\begin{aligned} X(0) &= (x(0) + x(2)) + (x(1) + x(3)) \\ X(1) &= (x(0) - x(2)) - i(x(1) - x(3)) \\ X(2) &= (x(0) + x(2)) - (x(1) + x(3)) \\ X(3) &= (x(0) - x(2)) + i(x(1) - x(3)) \end{aligned} \tag{4.3.34}$$

If we reorganize the above calculations as shown in Figure 4.26, the number of multiplications can be reduced down to eight by direct pass-through half of calculated numbers in the diagram. This is a 50% reduction.

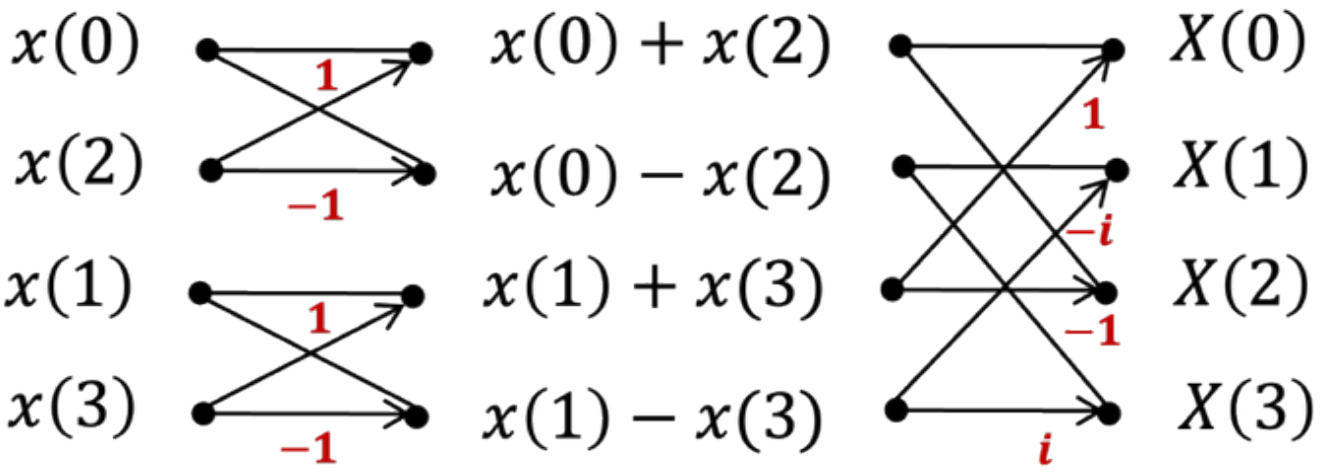


Figure 4.26: Four-point FFT diagram.

This calculation scheme can be extended to any number of points in the form of $N = 2^m$, where m is a positive integer. When used for the discrete Fourier transform (DFT), this scheme is called fast Fourier transform (FFT). As the number of DFT points increases, the percentage of calculation savings by FFT increases significantly. For most practical applications, the saving could be more than 90%, which is really important for real-time application of digital Fourier transform. Note that many digital signal processing algorithms require conducting discrete Fourier transform in real-time.

Example 4.10

For this example, a $2^m = 2^{14}$ point normal distribution random signal $f(k)$ is generated using the MATLAB command `randn` with a sample period of $T_s = 0.002$ second; see the top plot of Figure 4.27. A $2^{14} = 16384$ point fast Fourier transform (FFT) of $f(k)$ is obtained using the MATLAB command `fft(f,2^14)`; see the bottom plot of Figure 4.27.

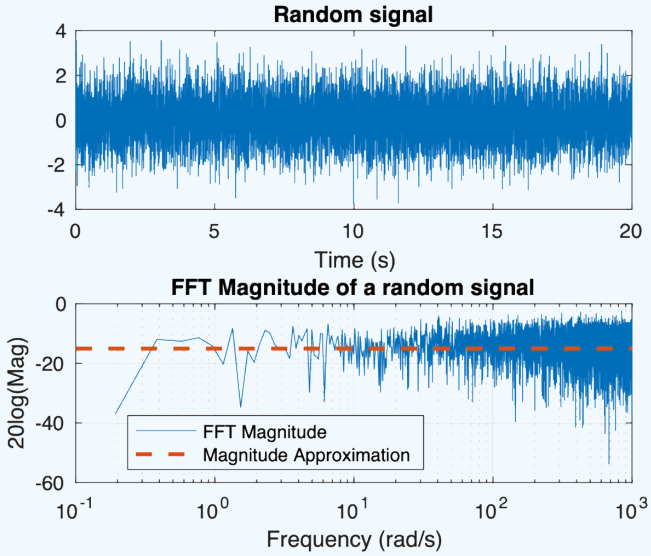


Figure 4.27: Frequency response of a Matlab generated white noise.

Note that the second entry of the MATLAB FFT command needs to be an integer of power of 2 for the fast Fourier transform. We know that the frequency response of a normal distributed random signal should be close to a straight line (constant). Figure 4.27 indicates this with a straight red-line approximation in the bottom plot. We can see that the FFT frequency magnitude oscillates around it.

Section Summary

This section addresses three main subjects. It introduces the digital signals inside a mechatronic system obtained by analog-to-digital (A/D) conversion of system analog sensor signals through a so-called sample and hold process. It then covers the discrete-time z-transform (similar to the continuous-time Laplace transform) with an example to develop a discrete-time system model in the form of a z-transform based upon the continuous-time differential equations. The section also discusses digital signal frequency analysis using the discrete Fourier transform [6], [7], [8], along with the fast Fourier transform (FFT) [7]. Note that when n (number of FFT points) is large, the FFT can reduce calculation throughput by more than 90%, compared with the conventional discrete Fourier transform. Also, note that for the FFT, the highest frequency information is limited by the signal sample frequency, and the lowest frequency information is limited by the signal length l.

Section Keywords

Practice Problems

1. Find the z-transform of the following discrete-time signals.
   a. $f(k) = 2kT$
   b. $f(k) = 4\cos 2k$
   c. $f(k) = 1 + 2e^{-2kT}\sin 2k$
2. For the discrete-time state-space transfer function defined in equation 4.3.23 with $m = 1$, $c = 0.5$, and $k = 2$, find the discrete-time z-transfer function with a sample period of 0.1 second.
3. Consider the discrete signal $f(k) = \cos(2\pi kT_s) + 0.5\sin(10\pi kT_s)$ with $T_s = 0.005$ second. Find the 2^{14} point FFT of $f(k)$. Then plot the time response in the top plot and frequency magnitude in the bottom plot with vertical axis in $20\log(|F(n)|)$ and horizontal axis in $\log(\omega)$, where ω is angular frequency in rad/s. Note that this axial choice makes a Bode liked magnitude plot.

This page titled 4.3: Digital Signals and Their Frequency Analysis is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

4.4: Analog and Digital Filter Design

Learning Objectives

After completing this section, you will be able to:

1. Provide concepts of analog and digital filtering.
2. Describe low-pass, high-pass, band-pass, and notch (band-kill) filtering concepts.
3. Design low-pass, high-pass, band-pass, and notch filters (analog and digital).
4. Provide working principles of analog filter realization circuits.

This section discusses the signal conditioning of both analog and digital signals to get rid of unwanted signal components such as sensor noise. We will first address the widely used low- and high-pass filtering concepts and the associated design methodologies, along with their realizations. We will then discuss the band-pass and band-kill (notch) filters based on low- and high-pass filters.

In general, the filter design is conducted in the frequency domain. Let $U(s)$ be the Laplace transform of the signal to be filtered, $Y(s)$ the Laplace transform of filtered signal, and $F(s)$ the continuous-time filter transfer function. Then,

$$Y(s) = F(s)\,U(s) \tag{4.4.1}$$

Similarly, let $U(z)$ be the z-transform of the signal to be filtered, $Y(z)$ be the z-transform of filtered signal, and $F(z)$ the discrete-time filter transfer function. Then,

$$Y(z) = F(z)\,U(z) \tag{4.4.2}$$

First, let's consider the case of low-pass filtering. Ideally, for a given cutoff frequency ω_c, the low-pass filter allows the signal frequency component below ω_c to pass while denying the signal frequency component above ω_c; see the left plot of Figure 4.28. That is, below the cutoff frequency, $\omega < \omega_c$, $|F(j\omega)| = 1$, allow the low frequency signal to pass. Beyond the cutoff frequency, $\omega > \omega_c$, $|F(j\omega)| = 0$, prevent the high frequency signal from passing.

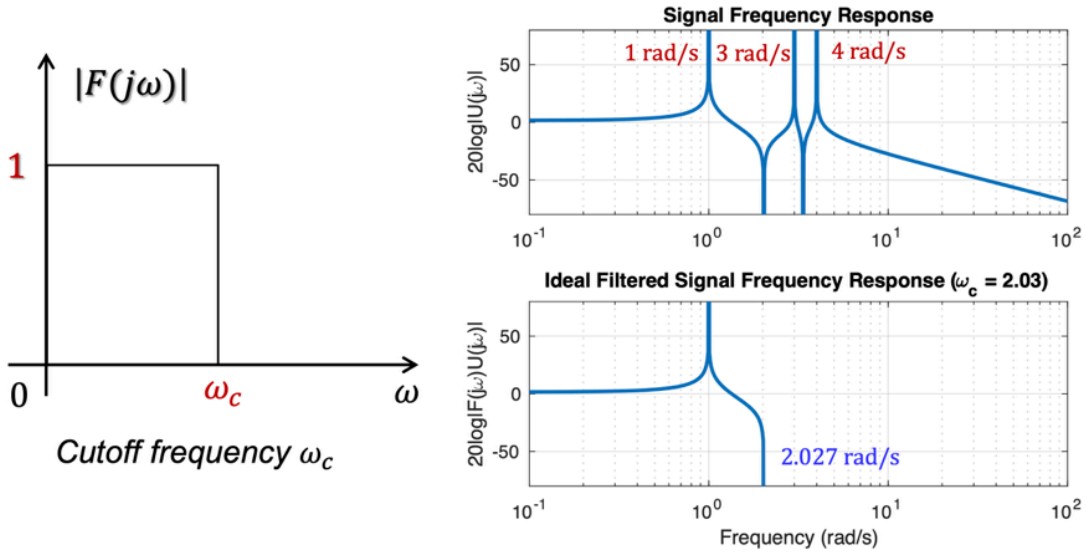


Figure 4.28: Ideal low-pass filter characteristics.

If, for a continuous-time low-pass filter transfer function, we want to ensure that the magnitude of the frequency response goes to zero as the frequency increases to infinity, then the order of the denominator's polynomial must always be higher than the numerator's. In other words, there are more poles than zeros for a low-pass filter transfer function. On the other hand, there is also

no pole nor zero at the origin of the complex plan. Therefore, the low frequency response magnitude has non-zero finite value. The right plot of Figure 4.28 illustrates the result of the signal defined in equation ???, filtered by the ideal low-pass filter with $\omega_c = 2.03$ rad/s. We can see that the frequency response was cut off after the cutoff frequency $\omega_c = 2.03$ rad/s.

However, the ideal low-pass filter is not realizable using a rational-proper transfer function that has finite order with numerator's polynomial order less than or equal to that of denominator's polynomial. The typical first order low-pass filter transfer function $F_{L1}(s)$ with unit gain is shown below.

$$F_{L1}(s) = \frac{\omega_c}{s + \omega_c} \tag{4.4.3}$$

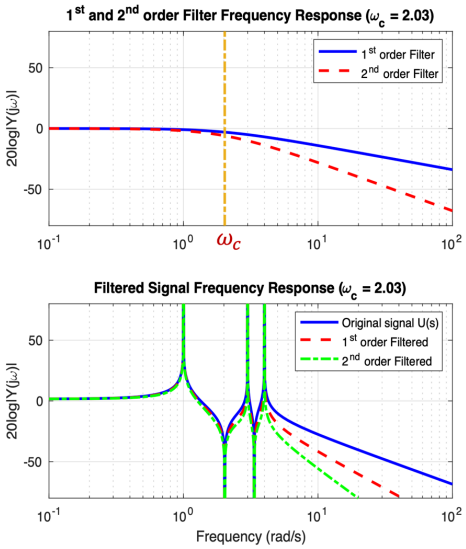


Figure 4.29: Frequency responses of the first and second order low-pass filters and filtered responses.

The typical second order low-pass filter transfer function $F_{L2}(s)$ with unit gain is presented below.

$$F_{L2}(s) = \left(\frac{\omega_c}{s + \omega_c}\right)^2 = \frac{\omega_c^2}{s^2 + 2\omega_c s + \omega_c^2} \tag{4.4.4}$$

Note that the unit gain of a transfer function can be confirmed by setting s to zero for both first and second order filter transfer functions. We can see the frequency responses of both first and second order low-pass filters in the top plot of Figure 4.29. Comparing these responses to the ideal low-pass filter characteristics in Figure 4.28, we can see that neither the first or second order low-pass filters from equations 4.4.3 and 4.4.4 kill the frequency response beyond the cutoff frequency ω_c. Instead, they only reduce the gain beyond cutoff frequency ω_c. Here, the first order low-pass filter reduces signal magnitude by 20 dB every 10 times frequency after the cutoff frequency. In addition, the second order low-pass filter reduces signal magnitude by 40 dB every 10 times frequency after the cutoff frequency.

Working from the bottom plot of Figure 4.28, for the frequency response presented in Figure 4.29 (blue line), the first order low-pass filtered frequency response is shown by the dashed-red line. The second order response is shown by the green dash-dot line. As the low-pass filter order increases, the decrement rate of frequency response magnitude after the cutoff frequency increases. Note that the high-order low-pass filter can also be implemented in the form below.

$$F_{Ln}(s) = \left(\frac{\omega_c}{s + \omega_c}\right)^n \tag{4.4.5}$$

where $n > 0$ (integer) is the order of the low-pass filter. Advanced high order low-pass filter design will be addressed in Section 4.7.

Analog filters can be implemented using analog circuits. Figure 4.30 shows two circuits that can be used to realize the first order low-pass analog filter based upon an RC (resistor and capacitor) circuit. Its first order transfer function can be derived utilizing the circuit modeling knowledge from Chapter 2. Note that high-order analog low-pass filters can also be realized by repeating the circuit shown in the right side of Figure 4.30.

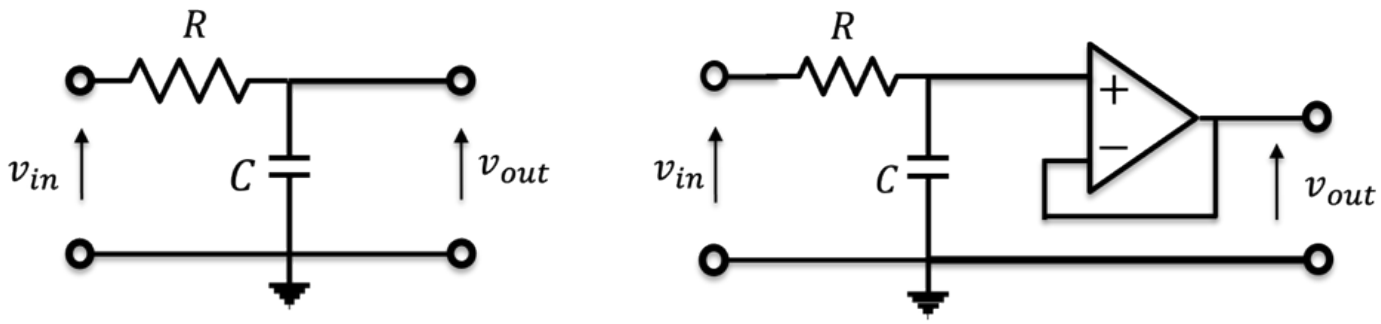


Figure 4.30: Sample realization circuit of the first order low-pass filter.

The left circuit of Figure 4.30 is called an RC circuit for low-pass filter. The right circuit provides the same transfer function with an additional isolation circuit to make the low-pass circuit output impedance equal to zero. In that way, the filter characteristics will not be altered after it is connected to the other circuit. Utilizing the knowledge learned from Chapter 2, the transfer function of this circuit can be derived below.

$$F_{L1}(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{1}{sRC+1} = \frac{1/RC}{s+1/RC} \tag{4.4.6}$$

As a result, the cutoff frequency of the low-pass filter in Figure 4.30 is $\omega_c = 1/RC$. For a given cutoff frequency, there exist infinite combinations of resistor and capacitor values, where the resistor and capacitor values need to be selected properly for feasible implementation.

Example 4.11

For the second order low-pass filter transfer function defined in equation 4.4.4, design an electronic circuit and find resistor and capacitor values to have a cutoff frequency of $\omega_c = 10$ rad/s. Note that the second order low-pass filter can be realized by repeating the first order circuit as below.

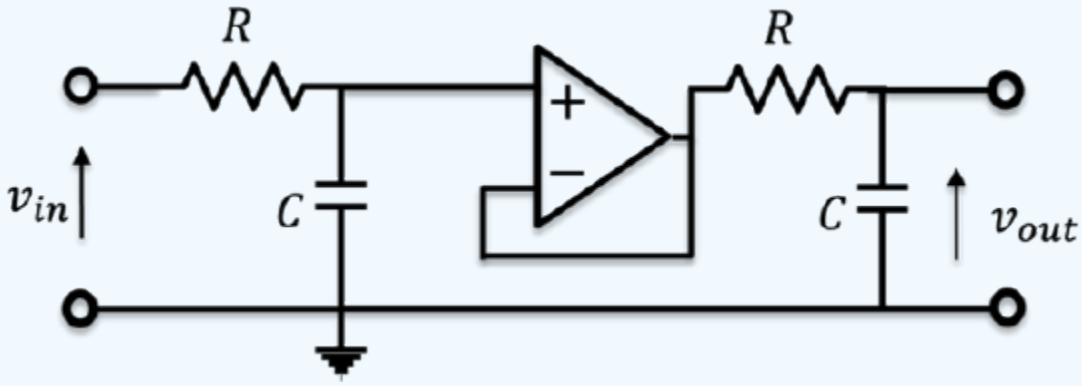


Figure 4.31: A second order low-pass filter circuit.

Due to the isolation circuit in the middle, the transfer function of the circuit in Figure 4.31 is shown below.

$$F_{L2}(s) = \frac{V_{out}(s)}{V_{in}(s)} = \left(\frac{1/RC}{s+1/RC}\right)^2 \tag{4.4.7}$$

Because the cutoff frequency is $\omega_c = 1/RC$, to make $\omega_c = 10$, let $R = 1\ k\Omega$ and we can calculate the capacitor value as below.

$$C = \frac{1}{R\omega_c} = \frac{1}{10^3 \times 10} = 100\ \mu F \tag{4.4.8}$$

Next, we will consider the case of high-pass filtering. For a given cutoff frequency ω_c, the high-pass filter ideally allows the signal frequency component beyond ω_c to pass while denying the signal frequency component below ω_c; see the left plot of Figure 4.32.

That is, below the cutoff frequency, $\omega < \omega_c$, $|F(j\omega)| = 0$, it prevents low frequency signals passing and, beyond the cutoff frequency, $\omega > \omega_c$, $|F(j\omega)| = 1$, it allows high frequency signals to pass.

Assume we want to ensure that the magnitude of the frequency response goes to one at the frequency beyond ω_c. In this case, for the continuous-time filter transfer function, the denominator's polynomial order should be equal to that of numerator. In other words, the numbers of poles and zeros need to be the same for a high-pass filter transfer function. On the other hand, at least one zero is needed at the origin of the complex plan to make it possible to have zero value at a frequency equal to zero. The right plot of Figure 4.32 illustrates the result of the signal defined in equation ??? filtered by an ideal high-pass filter with $\omega_c = 2.03$ rad/s. We can see that the frequency response was cut off before the cutoff frequency $\omega_c = 2.03$ rad/s.

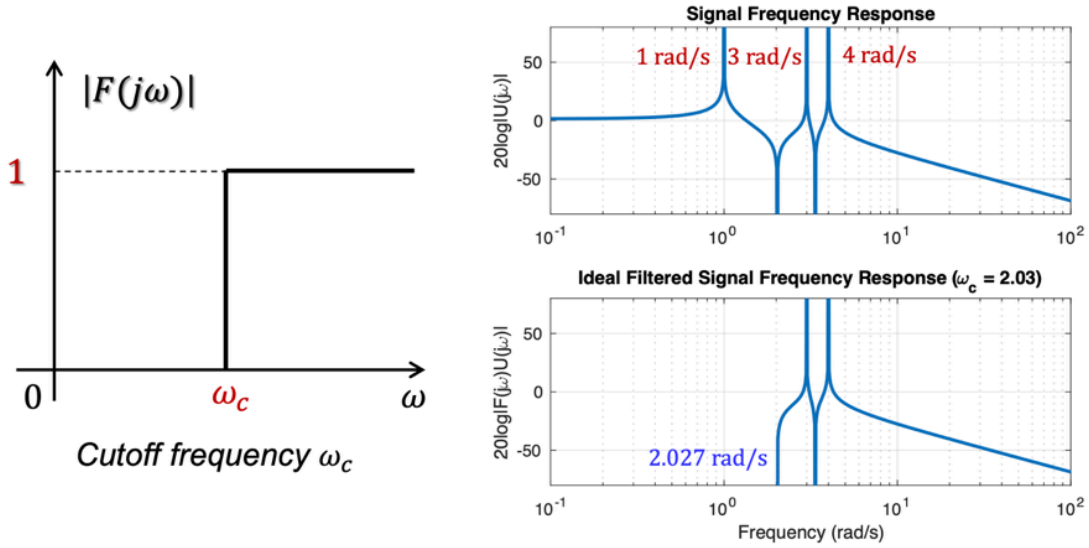


Figure 4.32: Ideal high-pass filter characteristics.

From a practical perspective, the ideal high-pass filter is not realizable using a rational-proper transfer function with finite orders of both numerator and denominator's polynomials. The typical first order high-pass filter transfer function $F_{H1}(s)$ with unit gain is shown below.

$$F_{H1}(s) = \frac{s}{s + \omega_c} \tag{4.4.9}$$

In turn, the typical second order high-pass filter transfer function $F_{H2}(s)$ with unit gain is shown below.

$$F_{H2}(s) = \left(\frac{s}{s + \omega_c}\right)^2 = \frac{s^2}{s^2 + 2\omega_c s + \omega_c^2} \tag{4.4.10}$$

The unit gain at frequencies higher than the cutoff frequency ω_c can be confirmed by letting s go to infinity for both first and second order high-pass filter transfer functions. The frequency responses of these filters can be found in the top plot of Figure 4.33. Compared with the ideal high-pass filter characteristics in Figure 4.32, we can see that, instead of killing the frequency response below the cutoff frequency, both first and second order low-pass filters from equations 4.4.9 and 4.4.10 reduce the signal magnitude. The first order high-pass filter reduces signal magnitude by 20 dB every 10-time frequency below the cutoff frequency (20 dB increment per 10-time frequency). The second order high-pass filter reduces signal magnitude by 40 dB every 10 times frequency before the cutoff frequency (40 dB increment per 10-time frequency). In addition, compared to the low-pass filter case, the characteristics of the high-pass filter are symmetric to the low-pass filter around the vertical axis at the cutoff frequency.

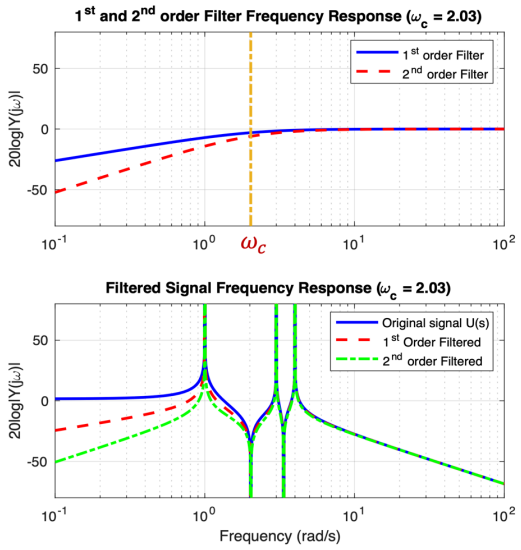


Figure 4.33: Frequency responses of the first and second order high-pass filters and filtered responses.

For the frequency response presented in Figure 4.32 (blue line), the first order high-pass filtered frequency response is shown in the dashed-red line in the bottom plot of Figure 4.33. The second order high-pass filtered frequency response is represented by the green dash-dot line. We can see that as the high-pass filter order increases, the frequency response magnitude decreases before the cutoff frequency. Also, note that high-order high-pass filters can be implemented in the form below.

$$F_{Hn}(s) = \left(\frac{s}{s+\omega_c}\right)^n \tag{4.4.11}$$

where $n > 0$ (integer) is the order of the high-pass filter. Advanced high-order high-pass filter design will be addressed in Section 4,7. We will now again address how to implement the analog high-pass filters using analog CR (capacitor and resistor) circuits based on the circuit modeling knowledge from Chapter 2. High-order analog high-pass filters can also be realized based on this circuit. The first order high-pass filter can be realized using the circuit shown in Figure 4.34. Here, the left circuit is a CR circuit for a high-pass filter; the right circuit provides the same transfer function with an additional isolation circuit to make the high-pass circuit output impedance equal to zero. Utilizing the knowledge learned from Chapter 2, the transfer function of the circuit in Figure 4.34 can be derived below.

$$F_{H1}(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{sRC}{sRC+1} = \frac{s}{s+1/RC} \tag{4.4.12}$$

As a result, the cutoff frequency of the high-pass filter in Figure 4.34 is $\omega_c = 1/RC$. Similar to the case of the low-pass filter, for a given cutoff frequency, infinite combinations of resistor and capacitor values exist.

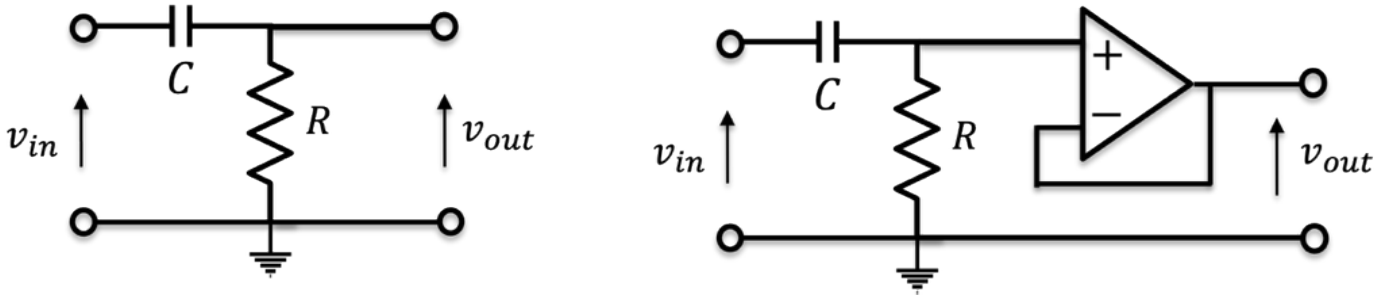


Figure 4.34: Sample realization circuit of the first order high-pass filter.

Example 4.12

For the second order high-pass filter transfer function defined in equation 4.4.10, design an electronic circuit. Then find resistor and capacitor values to have a cutoff frequency of $\omega_c = 10^4$ rad/s. Note that the second order high-pass filter can be realized by repeating the first order circuit from Figure 4.34.

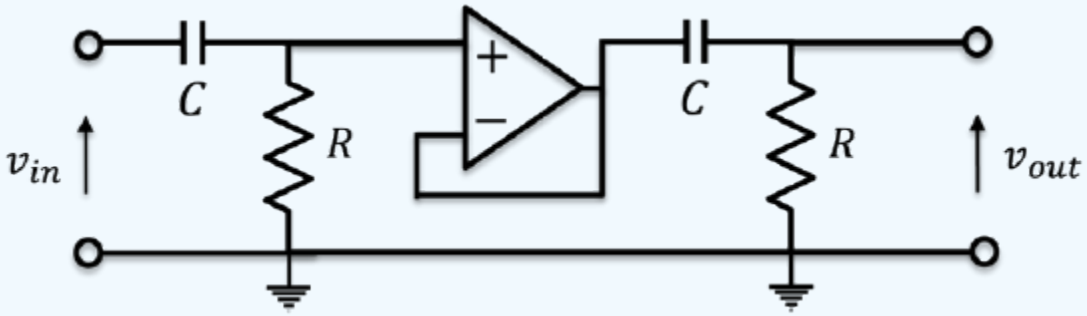


Figure 4.35: A second order high-pass filter circuit.

Given the isolation circuit in the middle, the transfer function of the circuit in Figure 4.35 is shown below.

$$F_{H2}(s) = \frac{V_{out}(s)}{V_{in}(s)} = \left(\frac{s}{s + 1/RC}\right)^2 \tag{4.4.13}$$

Because the cutoff frequency is $\omega_c = 1/RC$, to make $\omega_c = 10^4$, let $R = 1\ k\Omega$. we can then calculate the capacitor value as below.

$$C = \frac{1}{R\omega_c} = \frac{1}{10^3 \times 10^4} = 0.1\ \mu F \tag{4.4.14}$$

Next, let's consider the case of band-pass filtering. Ideally, for a pair of given cutoff frequencies ω_{c_1} and ω_{c_2}, the band-pass filter allows the signal frequency component between ω_{c_1} and ω_{c_2} to pass while denying the signal frequency component below ω_{c_1} and beyond ω_{c_2}; see the left plot of Figure 4.36. That is, below the first cutoff frequency, $\omega < \omega_{c_1}$, and beyond the second cutoff frequency, $\omega > \omega_{c_2}$, $|F(j\omega)| = 0$, preventing low and high frequency signal components from passing. At the middle frequency $\omega_{c_1} < \omega < \omega_{c_2}$, $|F(j\omega)| = 1$, allowing middle frequency signal components to pass.

In order to ensure that the magnitude of the frequency response goes to one at the frequency between ω_{c_1} and ω_{c_2}, the denominator's polynomial order should be higher than the numerator's. In other words, the number of poles should be greater than the number of zeros for a band-pass filter transfer function. On the other hand, there is also at least one zero at the origin of the complex plan to make it possible to have zero value at a frequency equal to zero. The right plot of Figure 4.36 illustrates the result of the signal defined in equation 4.4.15 filtered by the ideal band-pass filter with $\omega_{c_1} = 1.68$ and $\omega_{c_2} = 6.91$ rad/s. We can see that the frequency response was cut off before the first cutoff frequency $\omega_{c_1} = 1.68$ rad/s and after the second cutoff frequency $\omega_{c_2} = 6.91$ rad/s.

$$U(s) = \frac{1}{s^2+1} + 0.5\frac{10}{s^2+100} + \frac{3}{s^2+9} = \frac{9s^4 + 462s^2 + 1245}{s^6 + 110s^4 + 1009s^2 + 900} \tag{4.4.15}$$

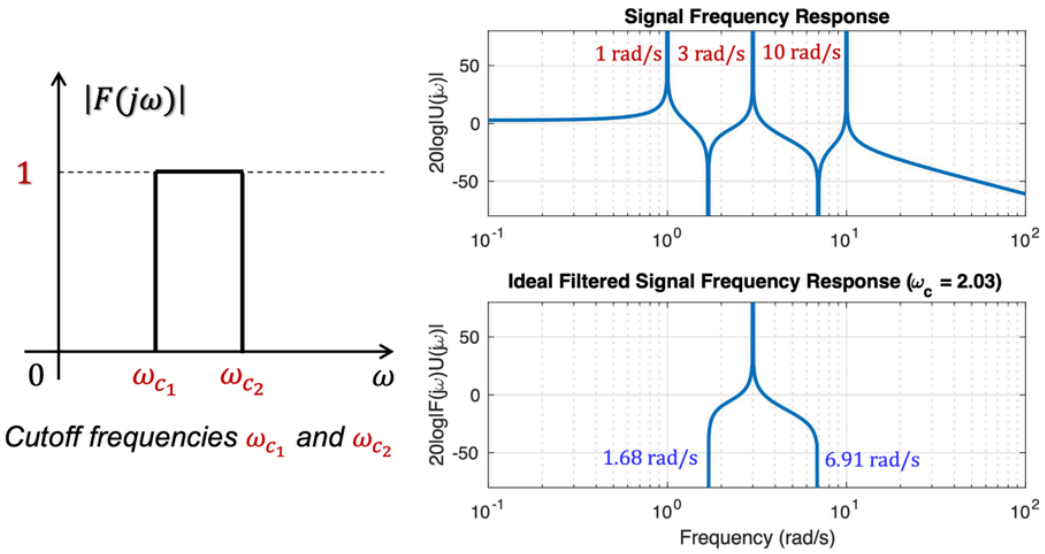


Figure 4.36: Ideal band-pass filter characteristics.

The ideal band-pass filter can be designed based on the ideal low- and high-pass filters with low-pass filter cutoff frequency ω_{c_2} greater than the high-pass filter cutoff frequency ω_{c_1}. Figure 4.37 illustrates using serially connected low- and high-pass filters to form a band-pass filter. Therefore, the band-pass filter can be formed based on the techniques discussed earlier in this section for both low- and high-pass filters.

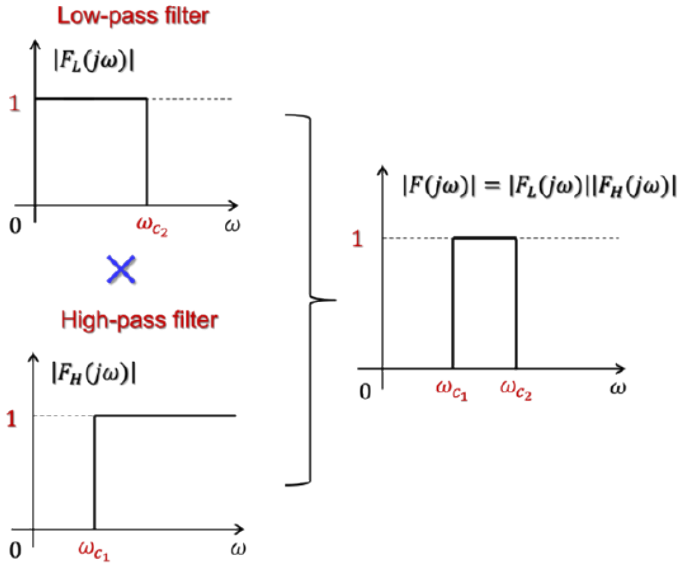


Figure 4.37: Low- and high-pass filter based band-pass filter concept.

Again, the ideal band-pass filter is not realizable using a rational-proper transfer function with finite orders of both numerator and denominator's polynomials. The typical first order band-pass filter transfer function $F_{B1}(s)$ with unit gain is shown below by serially connecting the low- and high-pass filters.

$$F_{B1}(s) = F_{L1}(s)\,F_{H1}(s) = \frac{s}{s+\omega_{c_1}}\frac{\omega_{c_2}}{s+\omega_{c_2}}, \omega_{c_2} > \omega_{c_1} \tag{4.4.16}$$

The typical second order band-pass filter transfer function $F_{B2}(s)$ with unit gain is shown below.

$$F_{B2}(s) = \left(\frac{s}{s+\omega_{c_1}}\frac{\omega_{c_2}}{s+\omega_{c_2}}\right)^2 = \left(\frac{s^2}{s^2+2\omega_{c_1}s+\omega_{c_1}^2}\right)\left(\frac{\omega_{c_2}^2}{s^2+2\omega_{c_2}s+\omega_{c_2}^2}\right) \tag{4.4.17}$$

The frequency responses of both first and second order band-pass filters with $\omega_{c_1} = 1.68$ and $\omega_{c_2} = 6.91$ rad/s can be found on the top plot of Figure 4.38. Compared with the ideal band-pass filter characteristics from Figure 4.36, we can see that, instead of killing the frequency response below the cutoff frequency $\omega_{c_1} = 1.68$ and beyond frequency $\omega_{c_2} = 6.91$, both first and second order band-pass filters from equations 4.4.16 and 4.4.17 reduce the magnitude before the first cutoff frequency ω_{c_1}. After the second cutoff frequency ω_{c_2}, the first order band-pass filter reduces signal magnitude by 20 dB every 10 times frequency below the cutoff frequency ω_{c_1} and beyond cutoff frequency ω_{c_2} (20 dB increment per 10-time frequency). The second order band-pass filter reduces signal magnitude by 40 dB every 10 times frequency before the cutoff frequency ω_{c_1} and beyond cutoff frequency ω_{c_2} (40 dB increment per 10-time frequency).

In addition, the characteristics of the band-pass filter have properties similar to high-pass filters below cutoff frequency ω_{c_1} and have properties similar to low-pass filters beyond cutoff frequency ω_{c_2}. In the bottom plot of Figure 4.38, the original frequency response is shown by the blue-line, the first order band-pass filtered frequency response is represented by the dashed-red line, and the second order high-pass filtered frequency response is shown in the green dash-dot line. We can see that as the band-pass filter order increases, the decrement rate of frequency response magnitude increases before the cutoff frequency ω_{c_1} and beyond the cutoff frequency ω_{c_2}.

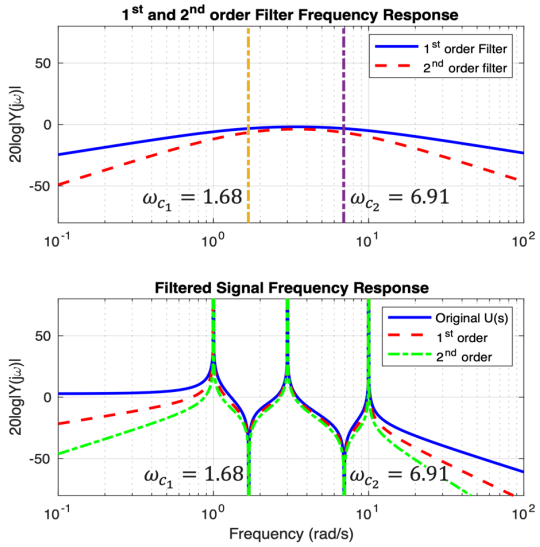


Figure 4.38: Frequency responses of the first and second order band-pass filters and their responses.

Also, the high-order band-pass filter can be implemented in the form below.

$$F_{Bn}(s) = \left(\frac{s}{s+\omega_{c_1}}\right)^n \left(\frac{\omega_{c_2}}{s+\omega_{c_2}}\right)^n \tag{4.4.18}$$

where $n > 0$ (integer) is the order of the band-pass filter. Advanced high-order band-pass filter design will be addressed in Section 4.7.

Now, we will address implementing the analog band-pass filters using analog resistor and capacitor (RC) circuits based on the circuit modeling knowledge from Chapter 2. Note that high-order analog band-pass filters can also be realized using similar circuits.

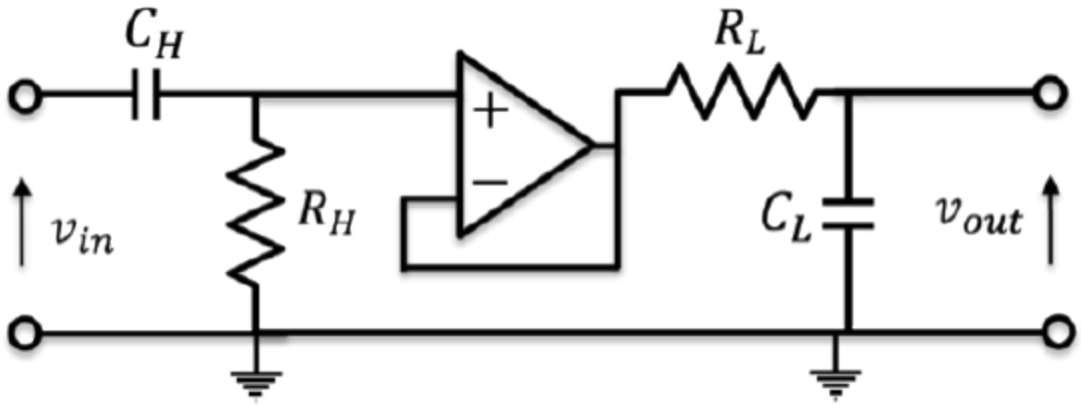


Figure 4.39: Sample realization circuit of the first order band-pass filter.

The first order band-pass filter can be realized using the circuit shown in Figure 4.39. Here, the high-pass filter is isolated from the second low-pass filter, where the order of RC circuits for both high-pass and low-pass filters can be switched. That is, we can have the high-pass filter on the left and the low-pass filter on the right, to have the same band-pass transfer function. Utilizing what we learned in Chapter 2, the transfer function of the above circuit can be derived and listed below.

$$F_{B1}(s) = \frac{V_{out}(s)}{V_{in}(s)} = \left(\frac{sR_HC_H}{sR_HC_H + 1}\right)\left(\frac{R_LC_L}{s + R_LC_L}\right) \tag{4.4.19}$$

As a result, the cutoff frequencies of the band-pass filter in Figure 4.39 are $\omega_{c_1} = 1/R_HC_H$ and $\omega_{c_2} = 1/R_LC_L$ with R_HC_H greater than R_LC_L. Similar to the case of low- and high-pass filters, for any given cutoff frequency, infinite combinations of resistor and capacitor values exist. We can realize the second order band-pass filter in a similar manner by replacing the first order low- and high-pass filters with the second order ones, respectively.

Example 4.13

For the first order band-pass filter circuit defined in Figure 4.39 with its transfer function defined in equation 4.4.19, find resistor and capacitor values to have cutoff frequencies $\omega_{c_1} = 1.68$ and $\omega_{c_2} = 6.91$ rad/s (used in the simulations in Figure 4.38) for given resistor values $R_H = R_L = 100\ k\Omega$.

Because the low cutoff frequency is $\omega_{c_1} = 1/R_HC_H = 1.68$ rad/s, to make $\omega_{c_1} = 1.68$, we can calculate the capacitor value C_H as below.

$$C_H = \frac{1}{R_H\omega_{c_1}} = \frac{1}{10^5 \times 1.68} = 5.952\ \mu F \tag{4.4.20}$$

For the second cutoff frequency of $\omega_{c_2} = 1/R_LC_L = 6.91$ rad/s, to make $\omega_{c_2} = 6.91$, we can calculate the capacitor value C_L as below.

$$C_L = \frac{1}{R_L\omega_2} = \frac{1}{10^5 \times 6.91} = 1.447\ \mu F \tag{4.4.21}$$

Next, we will consider the case of band-kill (notch) filtering. For a pair of given cutoff frequencies ω_{c_1} and ω_{c_2}, the band-kill filter ideally allows the signal frequency component below ω_{c_1} and beyond ω_{c_2} to pass while denying the signal frequency component between ω_{c_1} and ω_{c_2}; see the left plot of Figure 4.40. That is, at frequencies $\omega < \omega_{c_1}$ and $\omega > \omega_{c_2}$, $|F(j\omega)| = 1$, allowing low and high frequency signal component to pass. At the middle frequency $\omega_{c_1} < \omega < \omega_{c_2}$, $|F(j\omega)| = 0$, denying high frequency signal component to go through.

In order to ensure that the magnitude of the frequency response goes to one as frequency goes to infinity, the order of denominator's polynomial should equal the numerator's. In other words, the number of poles should be equal to that of zeros for a band-kill filter transfer function. On the other hand, there is no zero (nor pole) at the origin of the complex plan to have finite non-zero magnitude at frequency equal to zero (on infinity). The right plot of Figure 4.40 illustrates the result of the signal defined in equation 4.4.15 filtered by the ideal band-kill filter with $\omega_{c_1} = 1.68$ and $\omega_{c_2} = 6.91$ rad/s. We can see that the frequency response was cut off between the first cutoff frequency $\omega_{c_1} = 1.68$ and the second cutoff frequency $\omega_{c_2} = 6.91$ rad/s.

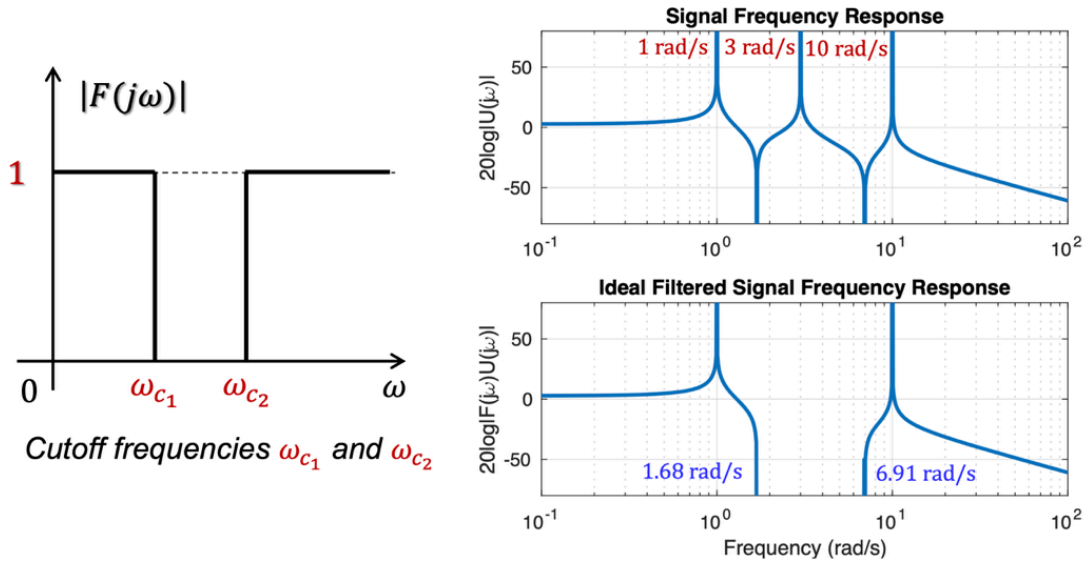


Figure 4.40: Ideal band-kill (notch) filter characteristics.

The ideal band-kill filter can be designed based on the ideal low-pass filter and high-pass filter with low-pass filter cutoff frequency ω_{c_1} smaller than the high-pass filter cutoff frequency ω_{c_2}. See Figure 4.41 for an illustration of using parallelly connected low- and high-pass filters to form a band-kill filter. As a result, the band-kill filter can be formed based on the techniques discussed earlier in this section.

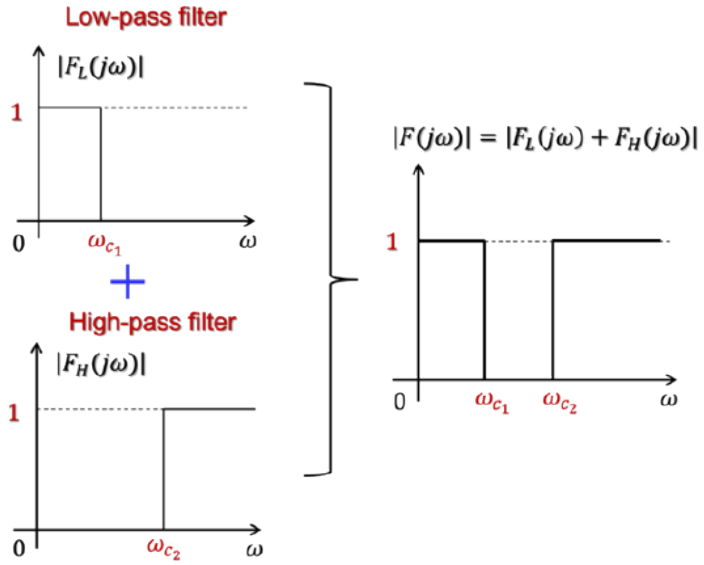


Figure 4.41: Low- and high-pass filter-based band-kill (notch) filter concept.

From a practical standpoint, the ideal band-kill filter is not realizable using a rational-proper transfer function with finite orders of both numerator and denominator's polynomials. The typical first order band-kill filter transfer function $F_{N1}(s)$ with unit gain is shown below.

$$F_{N1}(s) = F_{L1}(s) + F_{H1}(s) = \frac{\omega_{c_1}}{s+\omega_{c_1}} + \frac{s}{s+\omega_{c_2}}, \omega_{c_2} > \omega_{c_1} \tag{4.4.22}$$

The typical second order band-pass filter transfer function $F_{B2}(s)$ with unit gain is shown below.

$$F_{N2}(s) = F_{L2}(s) + F_{H2}(s) = \left(\frac{\omega_{c_1}}{s+\omega_{c_1}}\right)^2 + \left(\frac{s}{s+\omega_{c_2}}\right)^2 \tag{4.4.23}$$

The frequency responses of both first and second order band-kill filters with $\omega_{c_1} = 1.68$ and $\omega_{c_2} = 6.91$ rad/s can be found in the top plot of Figure 4.42. Compared with the ideal band-pass filter characteristics in the top plot of Figure 4.40, we can see that, instead of killing the frequency response between the cutoff frequencies $\omega_{c_1} = 1.68$ and $\omega_{c_2} = 6.91$, both first and second order band-pass filters from equations 4.4.22 and 4.4.23, where the first order band-kill filter reduces signal magnitude slightly. The second order band-kill filter reduces signal magnitude further. Both first and second order notch filters are able to reduce the magnitude between ω_{c_1} and ω_{c_2}, but are less effective for the first order filter. High-order filters are often required, which will be addressed in Section 4.7.

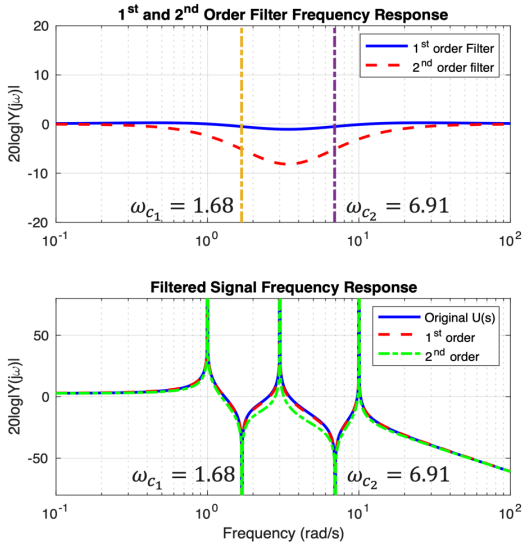


Figure 4.42: Frequency responses of the first and second order band-kill (notch) filters and their responses.

In the bottom plot of Figure 4.42, the original frequency response is plotted by the blue line, the first order band-kill filtered frequency response is shown in the dashed-red line, and the second order band-kill filtered frequency response is displayed in the green dash-dot line. The first order band-kill filter did not change the frequency response a lot, but the second order filter shows certain improvement. To design high-order band-kill filter, filtering techniques presented in Sections 4.7 shall be adopted. Note that the high-order band-kill filter can also be implemented in the form below.

$$F_{Nn}(s) = \left(\frac{\omega_{c_1}}{s+\omega_{c_1}}\right)^n + \left(\frac{s}{s+\omega_{c_2}}\right)^n \tag{4.4.24}$$

where $n > 0$ (integer) is the order of the low- and high-pass filters. Advanced high-order band-kill filters design will be addressed in Section 4.7.

Now let's address how to implement the analog band-kill filters using analog circuits consisting of resistors and capacitors based on coverage of circuit modeling in Section 2.3. High-order analog band-kill filters can also be realized using circuits.

The first order band-kill (notch) filter can be realized using the circuit shown in Figure 4.43. In the figure, high- and low-pass filters are parallelly connected with two isolators and one summation circuit of gain two. Here, ω_{c_1} is determined by R_L and C_L, ω_{c_2} is determined by R_H and C_H, and proper resistance values of R_1 and R_2 need to be selected (such as $1\ k\Omega$). Utilizing what we learned in Chapter 2, the transfer function of the above circuit can be derived below.

$$F_{N1}(s) = \frac{V_{out}(s)}{V_{in}(s)} = 2\left[\frac{sR_HC_H}{sR_HC_H+1} + \frac{R_LC_L}{s+R_LC_L}\right] \tag{4.4.25}$$

As a result, the cutoff frequencies of the band-kill filter in Figure 4.43 are $\omega_{c_1} = 1/R_LC_L$ and $\omega_{c_2} = 1/R_HC_H$ with R_HC_H smaller than R_LC_L. Similar to the case of low- and high-pass filters, for any given cutoff frequency, infinite combinations of resistor and capacitor values exist. The second order band-pass filter can be realized in a similar manner by replacing the first order low- and high-pass filters with the second order ones.

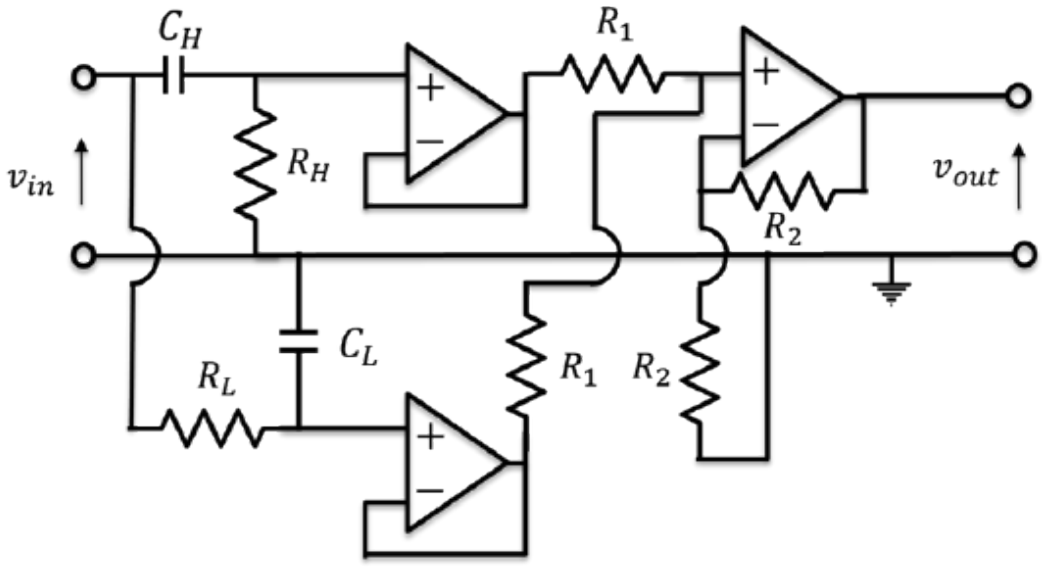


Figure 4.43: Sample realization circuit of the first order band-kill (notch) filter.

Example 4.14

For the first order band-pass filter circuit defined in Figure 4.43 with its transfer function defined in equation 4.4.25, find resistor and capacitor values to have cutoff frequencies $\omega_{c_1} = 1.68$ and $\omega_{c_2} = 6.91$ rad/s (used in earlier simulations) for given resistor values $R_H = R_L = 100\ k\Omega$. Because the cutoff frequency is $\omega_{c_1} = 1/R_L C_L = 1.68$ rad/s, to make $\omega_{c_1} = 1.68$, we can calculate the capacitor value C_L as below.

$$C_L = \frac{1}{R_L \omega_{c_1}} = \frac{1}{10^5 \times 1.68} = 5.952\ \mu F \tag{4.4.26}$$

For the second cutoff frequency of $\omega_{c_2} = 1/(R_H C_H) = 6.91$ rad/s, to make $\omega_{c_2} = 6.91$, we can calculate the capacitor value C_H as below.

$$C_H = \frac{1}{R_H \omega_{c_2}} = \frac{1}{10^5 \times 6.91} = 1.447\ \mu F \tag{4.4.27}$$

Note that calculated C_L and C_H values are complementary with those in Example 4.13.

Last, we will discuss digital filters and their implementations. Digital filters are within the information domain and implemented into a digital microprocessor. A digital filter uses microprocessors to conduct numerical calculations on sampled values of signals (see Figure 4.44). Digital filters have many advantages. They are programmable in microprocessors without changing hardware, and are easy to design, test, and implement. They eliminate hardware limitations such as resistance and capacitance. In addition, they are suitable for modern control system, signal processing, machine learning, etc.

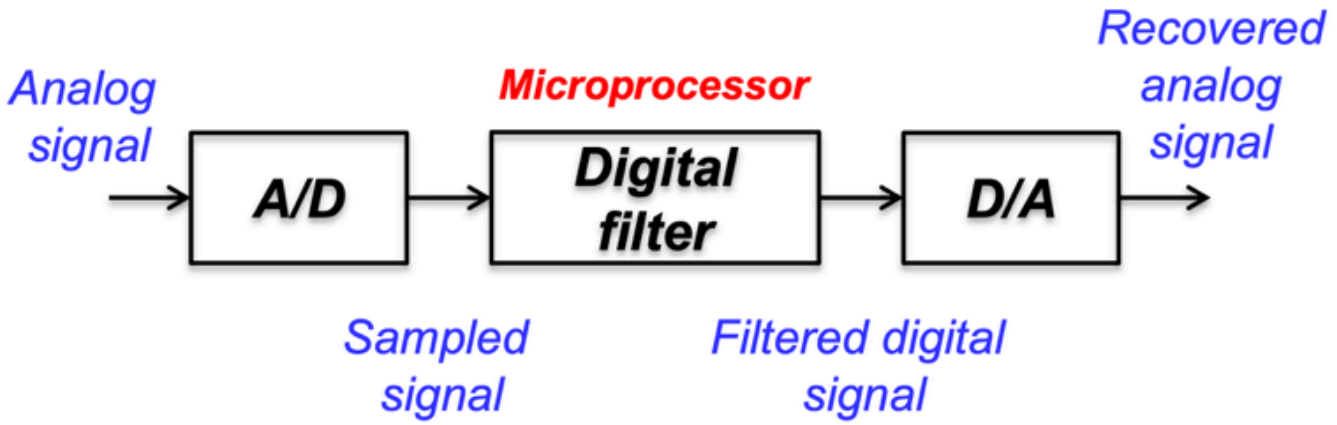


Figure 4.44: Digital filter implemented in the information domain.

We will address the low-pass digital filter first. Recall that the low-pass analog filter was introduced at the beginning of this section in equation 4.4.3. We will represent the digital filter by a z-transform and implemented as a numerical algorithm executed by the microprocessor. From the equivalence of the Laplace and z transforms in Table 4.3, we can see that the corresponding first order digital low-pass filter has the following z-transform.

$$F_{L1}(z) = \frac{Y(z)}{U(z)} = \frac{1 - e^{-\omega_c T}}{1 - e^{-\omega_c T} z^{-1}} \tag{4.4.28}$$

where ω_c is the continuous-time equivalent cutoff frequency, T is the sample period for the A/D converter, and z^{-1} is a one-step delay operator.

Note that $U(z)$ is the original signal and $Y(z)$ is the filtered signal in the z-transform. By the final value theorem of the z-transform (setting $z = 1$), we can see that $F_{L1}(1) = 1$. Recall that the final valve theorem for the Laplace transform sets "s" to zero. Therefore, this low-pass digital filter transfer function does have a DC gain equal to one. In order to implement the low-pass filter into an algorithm, we need to develop a calculation formula (algorithm). As a result, the above equation can be reorganized into the following.

$$Y(z)\left(1 - e^{-\omega_c T} z^{-1}\right) = \left(1 - e^{-\omega_c T}\right) U(z) \rightarrow Y(z) = \left(1 - e^{-\omega_c T}\right) U(z) + e^{-\omega_c T} z^{-1} Y(z) \tag{4.4.29}$$

The inverse z-transform of equation 4.4.29 yields the following.

$$y(k) = \left(1 - e^{-\omega_c T}\right) u(k) + e^{-\omega_c T} y(k-1) \tag{4.4.30}$$

Equation 4.4.30 shows an update logic (algorithm) for filtered signal $y(k)$ based on the current input signal $u(k)$ and the previous estimated signal $y(k-1)$. We can see that the update logic is a convex combination of the current input $u(k)$ and previous estimated value $y(k-1)$. As the cutoff frequency increases, more weight is on the current input $u(k)$ because $e^{-\omega_c T}$ reduces as ω_c increases, and vice versa.

One question worth answer is whether there is any correlation between the continuous- and discrete-time filters. Figure 4.45 plots out the frequency responses of both continuous- and discrete-time first order low-pass filters, where the cutoff frequency ω_c is 5 rad/s with a sample period of $T = 0.002$ (sample frequency of 500 Hz or 3141.6 rad/s) for the digital filter. Note that the blue-dashed and red-solid lines represent the continuous- and discrete-time filters, respectively.

Applying the Nyquist-Shannon theorem (see Section 4.5), the maximum frequency that can be handled by the digital filter is half of the sample frequency. Here it is 1570.8 rad/s, which is shown in Figure 4.45 as the vertical thin black-line. We can see that as the frequency approaches $\omega_c/2 = 1570.8$ rad/s, both the magnitude and phase traces diverge from the continuous-time ones, especially for the discrete-time phase response. In summary, at the frequency far away from $\omega_c/2 = 1570.8$ rad/s, the magnitude and phase responses of both continuous- and discrete-time first order low-pass filters are very close. That is, the digital filter behaves like the continuous-time one. It is important to know the limitation of the digital filter used.

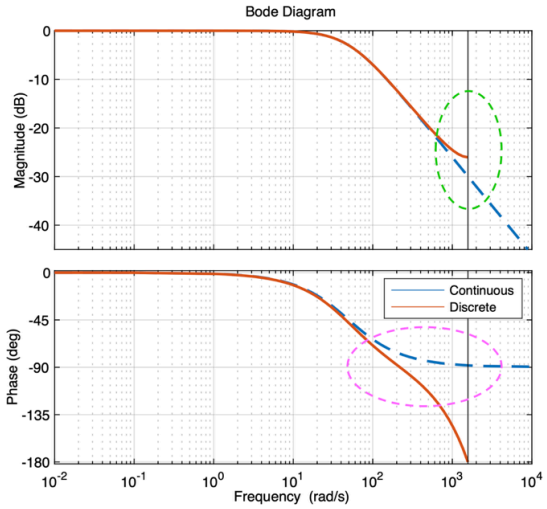


Figure 4.45: Frequency response comparison of continuous- and discrete-time low-pass filter.

The high-order low-pass filter can also be designed in a manner similar to the first order case and will not be addressed here.

Now, we will discuss digital high-pass filters. They can be considered as all-pass filtered signals minus a low-pass filtered one, as illustrated in Figure 4.46 with a sample transfer function of the first order filter.

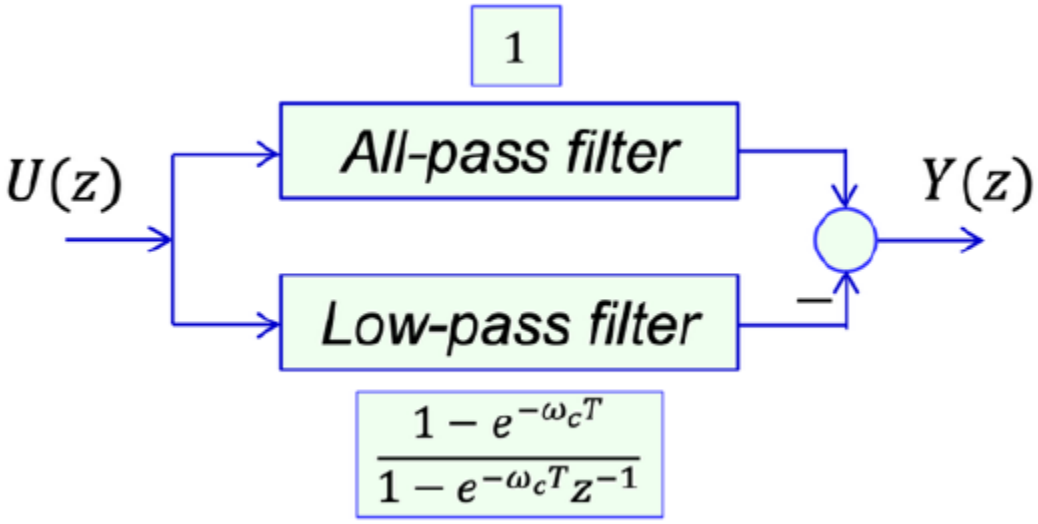


Figure 4.46: Discrete-time high-pass filter structure.

The z-transform of the first order high-pass transfer function can be derived in equation 4.4.31 based on the architecture shown in Figure 4.46.

$$F_{H1}(z) = \frac{Y(z)}{U(z)} = 1 - \frac{1 - e^{-\omega_c T}}{1 - e^{-\omega_c T} z^{-1}} = \frac{e^{-\omega_c T}(1 - z^{-1})}{1 - e^{-\omega_c T} z^{-1}} \tag{4.4.31}$$

From the filter transfer function shown in equation 4.4.31, the inverse z-transform yields the following high-pass filter calculation logic that can be implemented in the form of software.

$$y(k) = e^{-\omega_c T} y(k-1) + e^{-\omega_c T} [u(k) - u(k-1)] \tag{4.4.32}$$

However, for extremely fast applications, FPGA (field programmable gate arrays) silicon is often used for implementing digital filters with certain cost [11]. Both band-pass and band-kill filters can be designed using the low- and high-pass filters with the scheme of continuous-time filters presented in the beginning of this section. Therefore, we will not address these two-filter design

schemes again in the discrete-time case.

Example 4.15

For this example, we will study the effect of a first order digital filter on a random digital signal generated by the MATLAB command randn with 2^{14} points and variance equal to one. The original and FFT (fast Fourier transform) response are plotted in Figure 4.47, where the first 20-second time response is plotted (see the left chart) and its FFT is plotted (see the right chart). We assume that the digital signal has a sample period of $T = 0.002$ second (500 Hz), leading to the highest frequency component of 250 Hz (or 1570.8 rad/s) from FFT. We can also see that the FFT frequency response has a constant averaged magnitude with an oscillation magnitude that increases as frequency goes up.

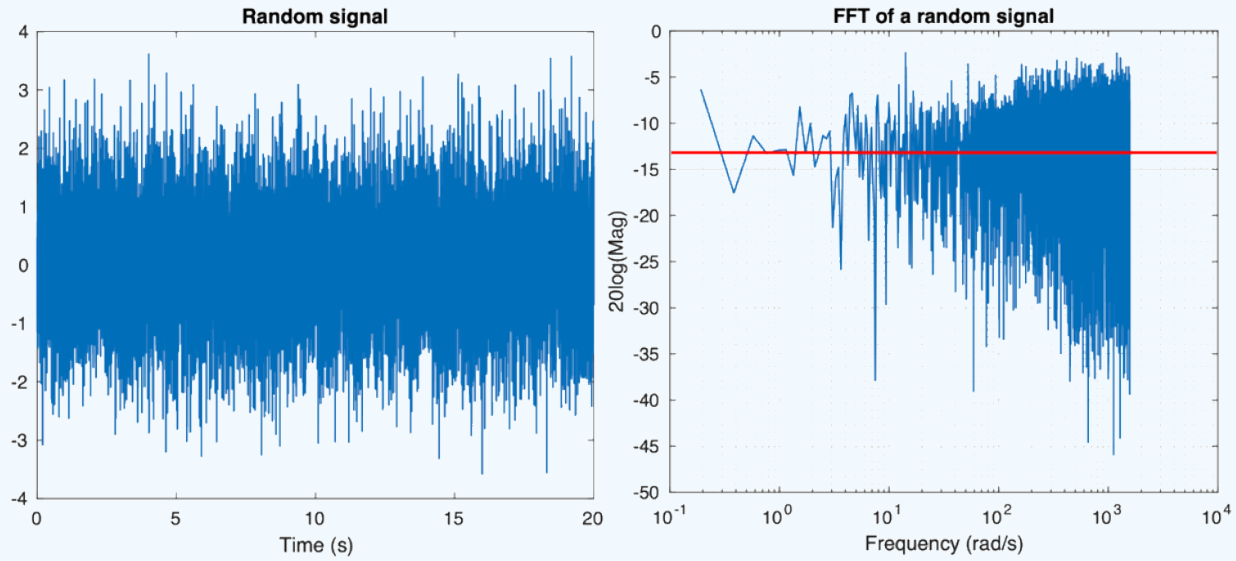


Figure 4.47: Normal random signal and its FFT magnitude.

Next, we will study the behavior of a first order digital filter equivalent to a given continuous-time filter with the transfer function below.

$$F_{L1}(s) = \frac{a}{s+a}, a = 5 \tag{4.4.33}$$

With the sample period of $T = 0.002$ second, we have the following z-transform of the digital low-pass filter.

$$F_{L1}(z) = \frac{1-b}{1-bz^{-1}}, b = e^{aT} = 0.9512 \tag{4.4.34}$$

Now, we will look at the frequency response of the first order digital filter based on FFT so that we can compare it with the result from Figure 4.45. Let $U(z)$ and $Y(z)$ be the fast Fourier transforms of the random signal generated by MATLAB and the time response of the filtered signal below.

$$U(z) = FFT\left(randn\left(2^{14}, 1\right)\right), Y(z) = FFT\left(by(k-1) + bu(k)\right) \tag{4.4.35}$$

We can estimate the filter frequency response numerically as below; see reference [23] for its derivation.

$$\hat{F}(z) = Y./U(z) \tag{4.4.36}$$

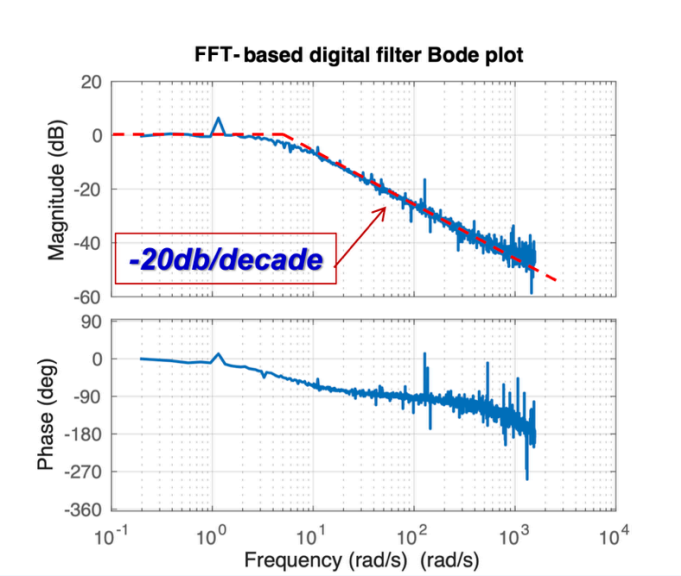


Figure 4.48: Frequency response of the first order low-pass filter calculated based on FFT.

where "./" is a point-wise division operation. Figure 4.48 above shows the frequency response calculated based on the FFT result in equation 4.4.35 with the MATLAB command sys=frd(Z, omega), where $omega = \omega$ is the frequency vector. It can be seen that the frequency response is similar to that in Figure 4.45.

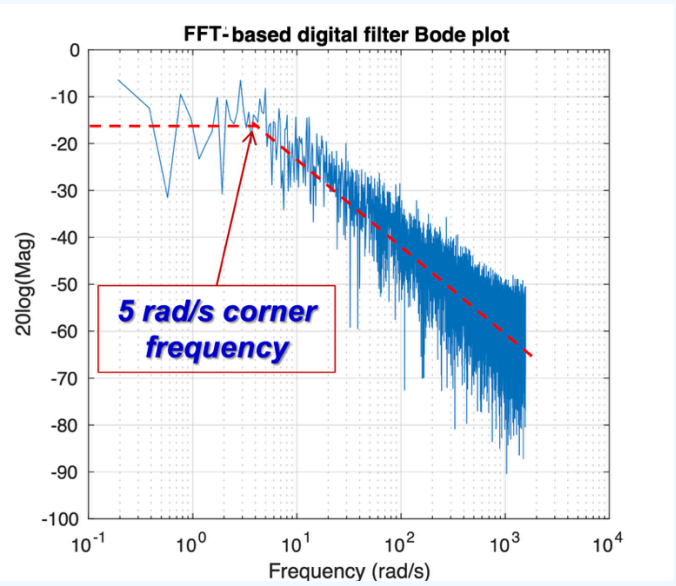


Figure 4.49: Frequency response of the first order low-pass filter calculated based on FFT.

Figure 4.49 plots the frequency response $Y(z)$ directly. The cutoff frequency is around 5 rad/s. Because the input signal is a random signal with uniform spectrum, the cutoff frequency of the filtered frequency response represents the filtered one. In addition, 20 dB decay of the frequency response is also observed, indicating a first order filter is used.

Section Summary

This section discusses analog and digital filters that are widely used to get rid of undesired information contained in the analog and digital signals, respectively. For instance, the low-pass filter is often used to get rid of high-frequency noise contained in the sensor signal. The high-pass filter could be used to eliminate the DC bias in the sensor signal, especially in the case that integration of the signal is required. There are also band-pass and band-kill (notch) filters that can be used to allow the signal within a specific frequency range to pass; they can also be used to get rid of a specific frequency component over a frequency

range.

Note that analog and digital filters [9], [10] have different characteristics. For certain applications (such as eliminating sampling aliasing, addressed in Section 4.5), only analog filters can be used. In this section, the design of both analog and digital filters is addressed in detail. We also looked at the realization of analog filters using electronic circuits and of digital filters using software algorithms without hardware. As a result, it is inexpensive to implement digital filters.

Section Keywords

Practice Problems

1. For the first order low-pass filter circuit defined in Figure 4.30, letting resistor value be $R = 10\ k\Omega$, find the capacitor value to make the cutoff frequency $\omega_c = 100$ rad/s.
2. For the first order low-pass filter circuit defined in Figure 4.34, letting capacitor value be $C = 1\ \mu F$, find the resistor value to make the high-pass filter cutoff frequency $\omega_c = 1000$ rad/s.
3. For the first order band-pass filter circuit defined in Figure 4.39, letting capacitor value be $C_H = C_L = 1\ \mu F$, find the resistor values R_H and R_L to make the band-pass filter cutoff frequency $\omega_{c_1} = 10$ and $\omega_{c_2} = 50$ rad/s.
4. For the first order band-kill (notch) filter circuit defined in Figure 4.43, letting capacitor value be $C_L = C_H = 1\ \mu F$, find the resistor values R_L and R_H to make the band-kill (notch) filter cutoff frequency $\omega_{c_1} = 10$ and $\omega_{c_2} = 50$ rad/s.

This page titled 4.4: Analog and Digital Filter Design is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters

Learning Objectives

After completing this section, you will be able to:

1. Utilize the Nyquist-Shannon sample theory.
2. Calculate the aliasing frequencies generated due to not satisfying the Nyquist-Shannon sample condition.
3. Select the corner frequency of the analog anti-aliasing filter.

This section discusses how to retain the analog signal information (physical domain) inside the sampled digital signal (information domain) and how to avoid sampling aliasing frequencies. When an analog signal is sampled by an A/D converter, the intension is to use a digital signal to represent the analog one with full information contained in the analog signal. However, the sampled digital signal has only the analog signal information at these sample points; the information between the sampled points is lost during the sampling process. The question now is how to make sure that the sampled digital signal contains the required (or necessary) information to be processed in the information domain. Intuitively, the sample rate (frequency) should be large enough to extract information rich enough from the analog signal. A slow sample rate cannot represent analog signal well. We will use an example to demonstrate the importance of selecting a proper sample rate.

Example 4.16

Consider the analog signal shown in equation 4.5.1 and in a solid line in the top-right plot in Figure 4.50.

$$f(t) = 1.5 \sin 2\pi t + 0.6t + 1 \tag{4.5.1}$$

The two left plots in Figure 4.50 show the two signal sampling schemes with one at 20 Hz (top left plot with $T = 0.05$ second) and the other at 2 Hz (bottom left plot with $T = 0.5$ second). We can see that with 0.05 second sample period, the top-left plot shows that the sampled digital signal (red dots) represents the original analog signal well. However, with sample period of 0.5 second, the bottom-left plot shows that the sampled signal (red dots) is a straight line if these dots are connected, indicating that the analog signal is as below.

$$f(t) = 0.6t + 1 \tag{4.5.2}$$

This indicates that with an improperly selected sample period, the sampled digital signal may not represent the original analog signal, making selecting the sample rate extremely important.

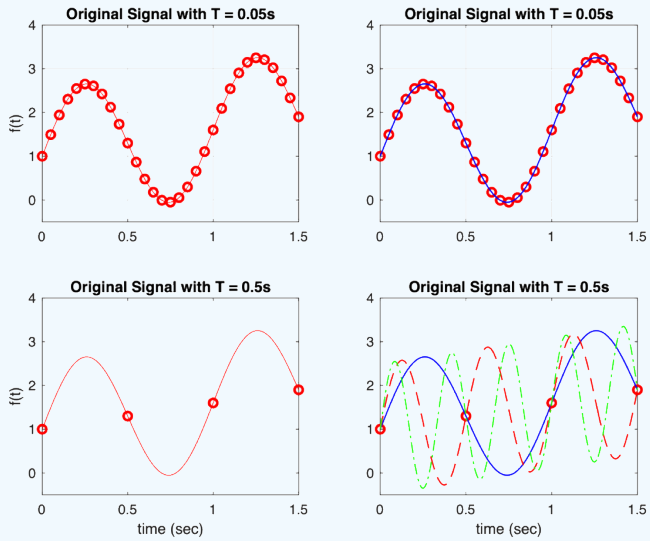


Figure 4.5.0: Effect of under-sampling a signal with missing information.

Now, let's look into the right two plots in Figure 4.5.0. The top-right plot indicates that with a sample rate of 20 Hz, the digital signal (red dots) is very close to the analog one (blue line). For the bottom-right plot, we can see that many different signals can have the same sampled digital signal such as $f_1(t)$ (red-dashed line) and $f_2(t)$ (green dash-dot line) shown below.

$$f_1(t) = 1.5\sin 4\pi t \ + 0.6t + 1 \tag{4.5.3}$$

$$f_2(t) = 1.5\sin 6\pi t \ + 0.6t + 1 \tag{4.5.4}$$

As a summary, because the analog signal between the sample points is missing, a digital signal obtained by sampling an analog signal could represent an infinite number of analog signals.

Now the question is what the proper sample rate (or condition) is so that the analog signal can be represented by sampled digital (quantized) signal in the information domain. The following Nyquist-Shannon sample theorem provides such information.

Theorem 4.1

The Nyquist-Shannon sample theorem indicates that signal sampling frequency (f_s) should be at least more than twice the highest frequency (f_h) component contained in the signal $f(t)$, or in mathematical term below,

$$f_s > 2f_h \tag{4.5.5}$$

Remark 4.2

In most of engineering applications, the sampling frequency (f_s) is often selected to be at least ten times of the highest frequency (f_h) contained in the signal, that is,

$$f_s \geq 10f_h \tag{4.5.6}$$

Remark 4.3

The Nyquist-Shannon sample theorem indicates that the sampled digital signal is able to recover the frequency components of an analog signal up to half of the sample frequency.

Example 4.17

This is an example about the telephone audio system. The maximum frequency (bandwidth) of the human voice is around 3.6 kHz. The human ear can detect a sound signal up to 15 kHz. At the phone microphone, the human voice is filtered by an anti-aliasing analog filter to substantially reduce signal magnitude at frequencies beyond 3.6 kHz. The filtered microphone signal is then sampled at 8 kHz to meet the Nyquist-Shannon sample theorem required to avoid sample aliasing. The human ear can detect sound up to 15 kHz. Because the voice signal frequency is limited to 3.6 kHz by the anti-aliasing low-pass filter, the sampled sound signal could be different from the original voice, making the sound from a telephone different from the actual sound.

Theorem 4.1 provides a guideline for selecting a proper sample rate so that the sampled analog signal characteristics can be recovered from the digital signal. As discussed in Remark 4.2, for practical applications, we recommend having the sample rate be 10 times greater than or equal to the highest frequency component contained in the sampled analog signal, even though the Nyquist-Shannon sample theorem only requires it to be two times greater than the highest frequency component. Now we will discuss what happens if the Nyquist-Shannon sample condition is not satisfied.

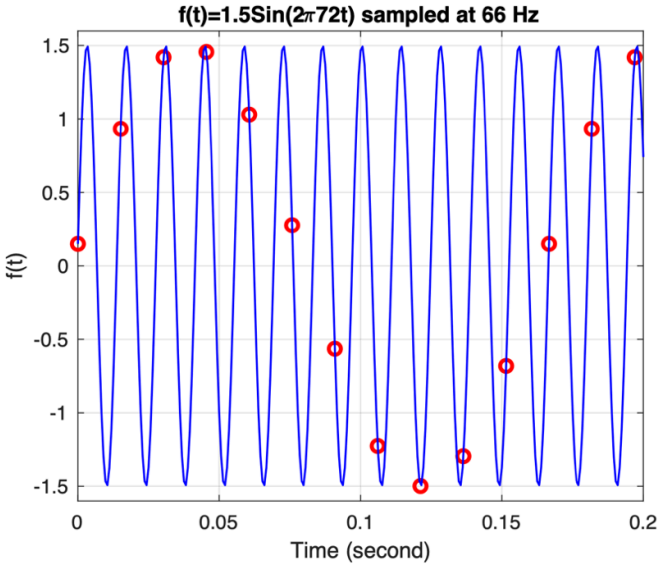


Figure 4.51: A 72 Hz $f(t) = 1.5\sin(2\pi \times 72t + 0.1)$ sampled at 66 Hz.

Figure 4.51 shows the plot of 72 Hz sinusoidal signal with a phase delay of 0.1 second in the solid-blue line and the sampled signal at 66 Hz in the red dots. We can see that the sampled analog signal has a frequency of 6 Hz that is not included in the original analog signal. That is, with the sample frequency (66 Hz) below the Nyquist-Shannon required frequency of greater than $2 \times 72 = 144$ Hz, the sampled signal shifted the signal frequency to be below the analog one. This phenomenon is called sample aliasing and the shifted frequency is called aliasing frequency. Let's look further how this happens. Consider a continuous-time sinusoidal signal.

$$f(t) = A\sin(2\pi f_0 t + \varphi) \tag{4.5.7}$$

Let the sample frequency to be $f_s = 1/T_s$ such that

$$f(n) = A\sin(2\pi f_0 n T_s + \varphi) \quad , t = 0, T_s, 2T_s, \cdots \tag{4.5.8}$$

Note that

$$f(n) = A\sin(2\pi f_0 n T_s + \varphi) = A\sin(2\pi(f_0 \pm l f_s) n T_s + \varphi) \tag{4.5.9}$$

for $l = 0, 1, 2, \cdots$. Furthermore, $\pm 2\pi l f_s n T_s = \pm \ln\{2\pi\}$ because $f_s T_s = 1$. This means that the sampled signal $A\sin(2\pi(f_0 \pm l f_s) n T_s + \varphi)$ with $l = 0, 1, 2, \ldots, \infty$ results in equivalent sequences for any integer l.

Remark 4.4

Based upon the discussion above, if the Nyquist-Shannon sampling condition is not satisfied, the aliasing frequencies of a signal sampling system can be calculated using the following formula.

$$f_a = |f_0 \pm l f_s|, l = 1,\ 2,\ \ ... \tag{4.5.10}$$

Note that there could be more than one aliasing frequency such that \(|f_0 \pm lf_s |

The next example demonstrates the case with more than one aliasing frequency.

Example 4.18

For this example, we will use three sinusoidal signals, as shown in equation 4.5.11, with their frequencies at 30, 170, 230 Hz. Our plan is to study the digital signal characteristics when these signals are sampled at 200 Hz. Figure 4.52 shows the analog signals plotted using blue lines.

$$\begin{aligned} f_1(t) &= \sin(2\pi 30t + 0.1) \\ f_2(t) &= -\sin(2\pi 170t - 0.1) \\ f_3(t) &= \sin(2\pi 230t + 0.1) \end{aligned} \tag{4.5.11}$$

The digital signals sampled at $T_s = 0.005$ second (200 Hz) are shown in equation 4.5.12, with sampled points marked with red dots in Figure 4.52.

$$\begin{aligned} f_1(k) &= \sin(2\pi(30)kT_s + 0.1) \\ f_2(k) &= -\sin(2\pi(170)kT_s - 0.1) \\ f_3(k) &= \sin(2\pi(230)kT_s + 0.1) \end{aligned} \tag{4.5.12}$$

Let's check analytically if any aliasing frequencies are generated during the sampling process. First, consider 30 Hz analog signal $f_1(t)$ sampled at 200 Hz. Because the sample frequency $f_s = 200$ Hz is greater than two times of the highest signal frequency ($2f_h = 2 \times 30 = 60$ Hz), there is no aliasing frequency. Second, consider 170 Hz analog signal $f_2(t)$ sampled at 200 Hz. Because the sample frequency $f_s = 200$ Hz is smaller than two times of the highest signal frequency ($2f_h = 2 \times 170 = 340$ Hz), there is at least one aliasing frequency. Using the formula in equation 4.5.10, we can calculate the aliasing frequency as below.

$$f_a = |f_0 \pm l f_s| = |170 \pm l \times 200|, l = 1,\ 2,\ \ ... \tag{4.5.13}$$

The only aliasing frequency below the signal frequency $f_0 = 170$ Hz occurs when $l = -1$, yielding $f_a = 30$ Hz. From the middle graph of Figure 4.52, we can see that the sampled signal does have a frequency of 30 Hz. Last, consider 230 Hz analog signal $f_3(t)$ sampled at 200 Hz. Because the sample frequency $f_s = 200$ Hz is again smaller than two times of the highest signal frequency ($2f_h = 2 \times 230 = 460$ Hz), there is at least one aliasing frequency. Again, using the formula in equation 4.5.10 , we can calculate the aliasing frequency as below.

$$f_a = |f_0 \pm l f_s| = |230 \pm l \times 200|, l = 1,\ 2,\ \ ... \tag{4.5.14}$$

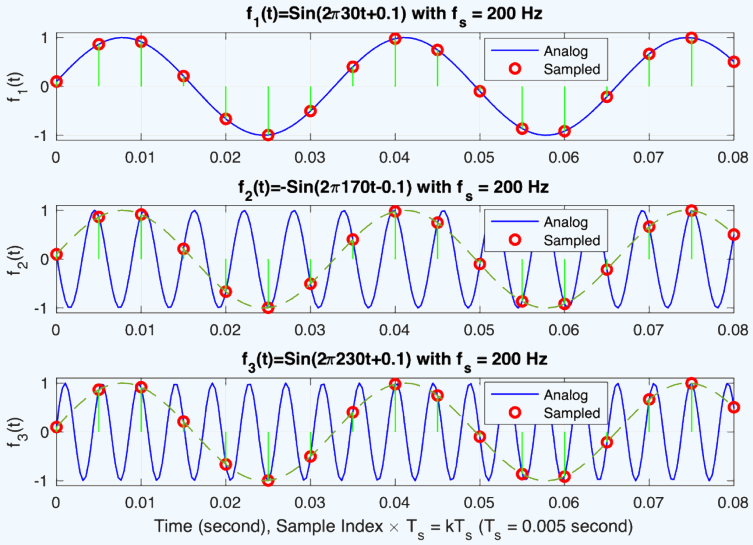


Figure 4.52: Three sinusoidal signals sampled at 200 Hz resulting signals with the same frequency.

The aliasing frequencies below the signal frequency $f_0 = 230$ Hz occur when $l = -1, \ -2$, leading to $f_a = 30, \ 170$ Hz. From the bottom graph of Figure 4.52, we can see that the sampled signal does have a frequency of 30 Hz. As a summary, for the three analog signals ($f_1(t)$, $f_2(t)$, and $f_3(t)$) sampled at 200 Hz, all of the resulting digital signals have a frequency at 30 Hz. This study indicates that for a 170 or 230 Hz sinusoidal function sampled at 200 Hz sample rate will result in a 30 Hz sinusoidal signal due to aliasing because the Nyquist-Shannon sample condition is not satisfied. Note that for $f_3(t)$ sampled at 200 Hz, the highest recoverable frequency is less than 100 Hz. As a result, the 170 Hz aliasing frequency cannot be seen from the simulation result shown in Figure 4.52.

Example 4.19

For this example, imagine a spinning wheel (with a fixed-dot on its edge) rotating counterclockwise (CCW) at a speed of 1, 2, 4, 6, 7, and 8 rotations per second. In other words, it rotates at 1, 2, 4, 6, 7, and 8 Hz. A strobe light is used to shine the dot on the wheel to find the wheel position; see Figure 4.53 for an illustration. The strobe light flashes at a fixed frequency of $f_s = 8$ Hz.

For the cases with a rotation speed of $f_h = 1$ or 2 Hz (top to rows), and because the Nyquist-Shannon sample condition is satisfied (\(2f_h

For the cases with rotation speed of $f_s = 4$ Hz (third row in Figure 4.53), because the Nyquist-Shannon sample condition is not satisfied ($2f_h = f_s$), there is a zero Hz aliasing frequency. As a result, the rotational direction cannot be identified. That is, the Nyquist-Shannon condition needs to be satisfied to recover sampled signal information (in this case, the rotational direction).

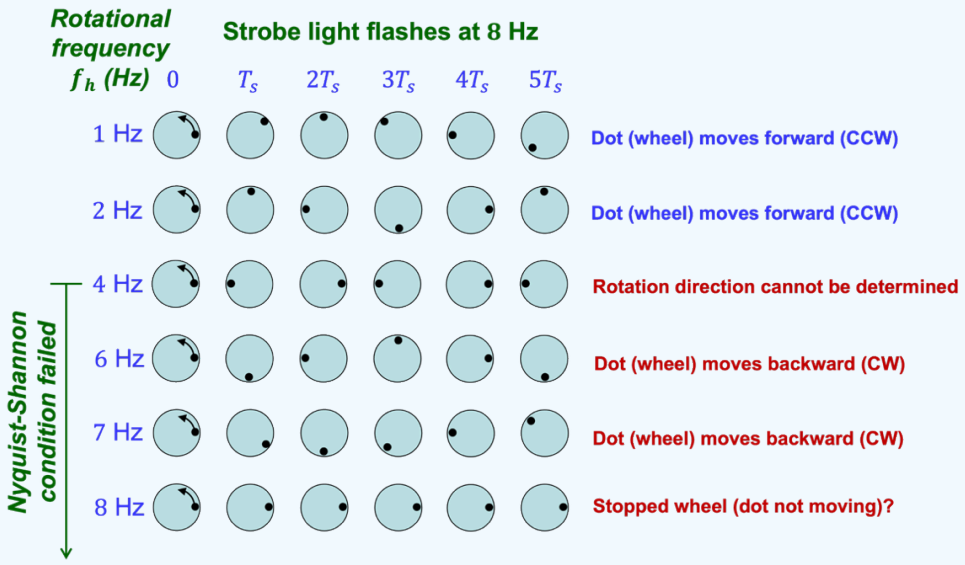


Figure 4.53: Rotational wheel position sampled by a strobe light.

For the cases with rotation speed of $f_s = 6,\ 7$, and 8 Hz (fourth, fifth and sixth rows), and because the Nyquist-Shannon sample condition is not satisfied ($2f_h > f_s$), there are aliasing frequencies of 2, 1, and 0 Hz, respectively. Especially for $f_s = 8$, the rotational direction cannot be identified. Note that 0 Hz aliasing frequency indicates that the sampled signal (in this case, wheel position) remains unchanged; see Figure 4.53.

Remark 4.5

The Nyquist-Shannon sample theorem states that to avoid aliasing frequency, the sample frequency f_s should be more than twice that of the highest frequency f_h contained in the analog signal $f(t)$, or $f_s > 2f_h$. To avoid aliasing in the sampling process, an anti-aliasing analog filter should be used (see Figure 4.54).

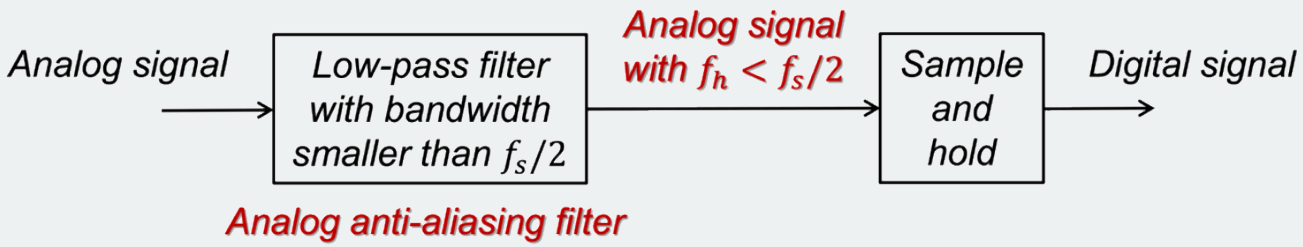


Figure 4.54: Anti-aliasing analog filter to eliminate aliasing frequency.

In practice, the anti-aliasing filter corner frequency should be at least 5 times smaller than the sample frequency.

Remark 4.6

The low-pass anti-aliasing filter must be an analog filter located before the A/D converter. A digital filter located after the A/D converter cannot eliminate any aliasing frequency.

Section Summary

This section discusses how to use analog-to-digital (A/D) converters for the sampled digital signal to retain the information contained in the original analog signal. It is both desired and important that the information contained in the analog signal from

the physical domain is retained in the sampled digital signal in the information domain. Otherwise, the sampled signals will not be useful in the information domain due to missing information. In addition, if the sample frequency is too low, the digital signal may contain the information in those frequencies (called aliasing frequencies) that are not contained in the original analog signals. This difference can lead to errors in the sampled digital signals and, as a result, in the control signals generated by the microcontroller. The Nyquist-Shannon sample theorem provides the sample frequency condition to avoid sample aliasing [12], [13]. This sample frequency should be at least two times higher than the highest frequency component contained in the analog signal. In order to avoid sample aliasing, once the sample frequency has been selected, an analog anti-aliasing filter should be designed and placed before the A/D converter, where the low-pass anti-aliasing filter has a corner frequency, at least, below one half of the sample frequency [14].

Section Keywords

Aliasing: This effect occurs when a sampled signal appears to have a lower frequency than its true value, causing a misrepresentation of the original signal. More specifically, it occurs when a signal is sampled at a rate lower than twice of its highest frequency, as described by the Nyquist-Shannon sampling theorem.

Anti-aliasing filter (**AAF**): This analog filter is used before a signal sampler (such as A/D converter) to restrict the bandwidth of the sampled signal so that the Nyquist-Shannon sampling theorem is satisfied over the band of interest.

Nyquist–Shannon sample theorem: This theorem is an essential principle for digital signal processing; it links the frequency range of a signal and the sample rate required to avoid a type of distortion called aliasing. The theorem states that the sample rate must be more than twice the bandwidth of the signal in order to avoid aliasing. In practice, it is used to select band-limiting filters that keep aliasing below an acceptable amount when an analog signal is sampled.

Practice Problems

1. If the highest frequency component of an analog signal is $f_h = 50$ Hz, justify whether the Nyquist-Shannon sample condition is satisfied or not for the following sample frequencies.
   a. $f_s = 100$ Hz
   b. $f_s = 80$ Hz
   c. $f_s = 120$ Hz
2. Repeat Example 4.19 for the rotation speed with clockwise (CW) rotational direction at a speed of 2 rotation per second ($f_h = 2$ Hz). With the same sample frequency of $f_s = 8$ Hz, determine if the Nyquist-Shannon sample condition is satisfied and if the wheel rotational direction can be determined.
3. Considering a continuous-time signal $f(t) = \cos(2\pi \times 400t - \pi)$, if it is sampled at a frequency of $f_s = 250\ Hz$, answer the following questions.
   a. Will $250\ Hz$ sample frequency cause aliasing and why?
   b. If aliasing exists, find all aliasing frequencies below the signal frequency (400 Hz).

This page titled 4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

4.6: Pulse-Width and Frequency Modulated Signals

Learning Objectives

After completing this section, you will be able to:

1. Describe the pulse-width modulation (PWM) signal and its applications.
2. Define and utilize the frequency modulation signals.
3. Distinguish the analog and PWM DC motor drive circuits.
4. Describe how the PWM signal is generated based on analog input.

At the beginning of this chapter, both physical and information domains were defined, along with the interface between these two domains (see Figure 4.1). The first three sections addressed converting physical domain analog signals to information domain digital signals, using analog-to-digital converters and vice versa using the digital-to-analog converters. Figure 4.1 indicated additional direct signal paths between the physical and information domains for signal conversions. This section discusses two special signals: pulse-width modulated (PWM) and frequency modulated (FM) signals. These can be used to exchange signals between the physical and information domains.

A pulse-width modulation (PWM) signal is a pulse train with fixed frequency and variable duty-cycle. Let's define the PWM signal percent duty cycle. First, let f_{PWM} represent the PWM signal frequency (fixed) and $\tau_c = 1/f_{PWM}$ be the period of the PWM signal. Also, let τ_0 be the duration for the PWM signal remaining at high voltage (see Figure 4.55). The percent of duty-cycle α_{PWM} can then be calculated by the following equation.

$$\alpha_{PWM} = \frac{\tau_0}{\tau_c} \times 100\% \tag{4.6.1}$$

Figure 4.55 provides a PWM voltage signal with voltage $V = V_h - V_l$ defined as the bottom (V_l) to peak (V_h) voltage. This signal will be useful when converting the digital signal in the information domain to the analog signal in the physical domain. Normally, V_l is very small; it can often be ignored. This will be discussed later in this section.

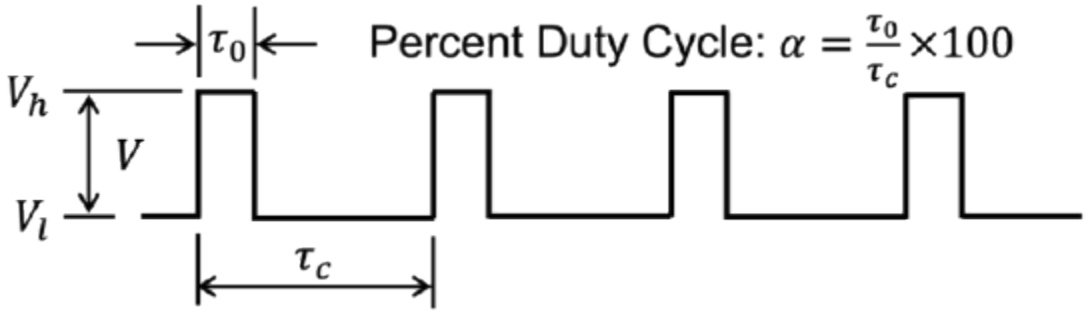


Figure 4.55: Sample PWM signal and duty-cycle definition.

The drawings of the PWM signals in Figure 4.56 have different duty cycles: 0%, 25%, 50%, 75%, and 100% are in red lines, whereas the vertical green lines are used to mark the boundary of each PWM period of $\tau_c = 0.04$ second (20 Hz). These plotted signals have a bottom-to-peak voltage of $V = 5$ volt. We can see that with 0% duty cycle, the PWM signal remains at the zero-volt level. With 100% duty cycle, the PWM signal remains at the five-volt level. For the PWM signals with 25%, 50%, and 75% duty cycle, the PWM signals are a square wave with fixed frequency but different peak duration (duty-cycle). In fact, only 0% and 100% duty cycle PWM signal remains unchanged. When the duty cycle is greater than zero and less than 100%, the PWM is a square wave with fixed frequency.

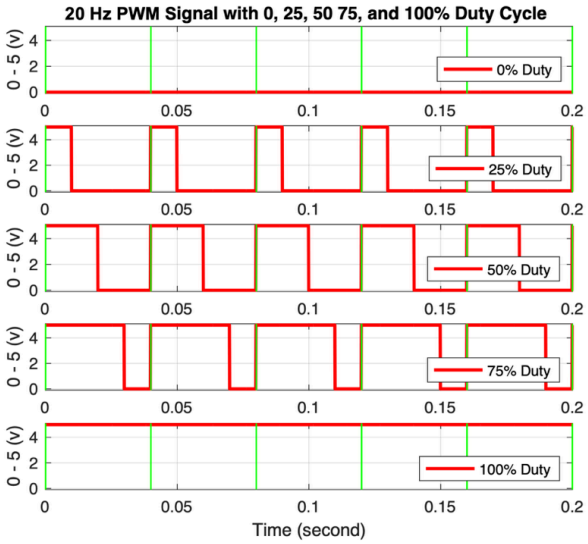


Figure 4.56: A 20 Hz PWM signal with duty cycles at 0, 25, 50, 75, and 100%.

The main advantages of using PWM signals are twofold: its high driving efficiency in energy conversion and high power density of the driving circuit. High drive circuit efficiency reduces the energy loss of the driving circuit. Hence, it reduces the driving circuit size, leading to compact drive electronics, while reducing heat generation and eliminating the required large heat sink. We will use a simple example below to illustrate the improvement of driving circuit efficiency.

For each PWM period, the averaged analog voltage, used to approximate the analog voltage signal, can be calculated below.

$$v_{analog} = duty_{cycle} \times V_h + (1 - duty_{cycle})V_l = \alpha V_h + (1 - \alpha)V_l \tag{4.6.2}$$

where duty-cycle α is defined earlier in equation 4.6.1. As a result, various analog signal values can be achieved by changing pulse width τ_o or duty cycle α.

Example 4.20

Consider the analog and PWM DC motor driving circuits illustrated in Figure 4.57 with the analog driving circuit on the left and the PWM circuit on the right. The control inputs (v_{in} and v_{PWM}) to both circuits, respectively, are to reach the target armature current i_a. We will use certain assigned circuit values for this discussion.

Let the battery voltage $V_{bat} = 12$ volt, DC-motor armature resistor $R_a = 0.5\ \Omega$, and armature current target $i_{a-target} = 8\ A$. Assume armature inductance $L_a = 0$ H and EMF voltage v_{EMF} to be 2 volts. Considering the analog drive circuit on the left of Figure 4.57, to simplify the analysis, we assume that the current flow into the base of transistor T is very small (zero); also assume the current feedback resistor R_3 is very small (ignored when calculating the power consumption). From Section 2.4, we can calculate the input analog voltage v_{in} as below. Assume that the transistor collector ("c" in Figure 4.57) current $i_a = i_c$ equals the transistor emitter (marked "e") current i_e by ignoring the current flow into R_2.

$$v_{in} = \frac{R_2}{R_2 + R_2}\left(R_3 i_{a-target}\right) = 0.5 R_3 i_{a-target} \tag{4.6.3}$$

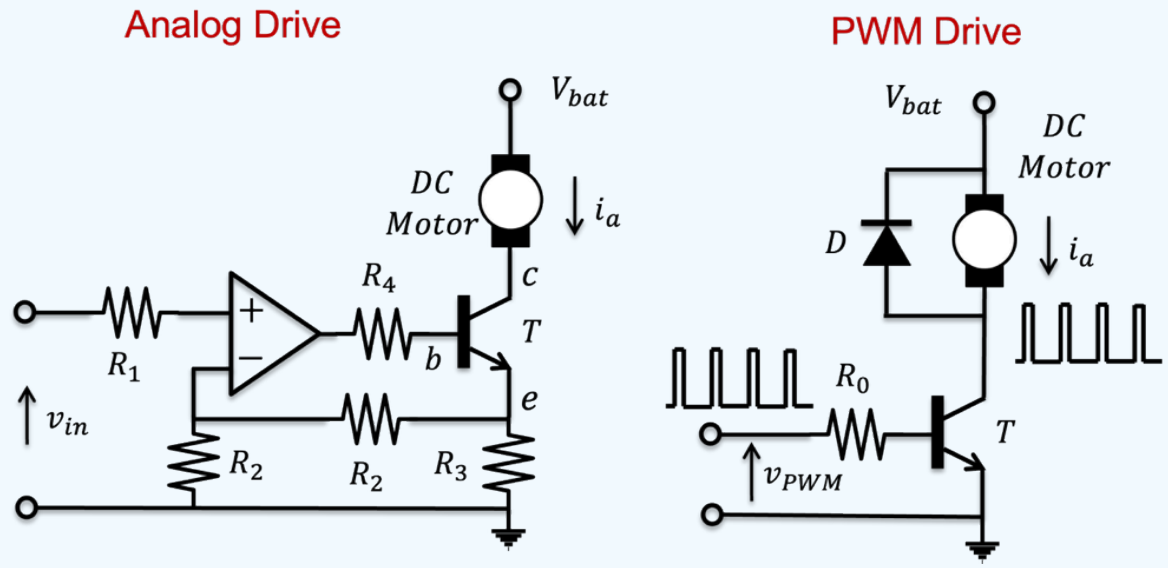


Figure 4.57: Driving a DC motor using analog and PWM signals.

With all the assumptions above, the DC-motor armature voltage is $v_a = R_a i_a + v_{EMF} = 0.5 \times 8 + 2 = 6$ volts. The associated power consumption is $P_{motor} = v_a i_a = 6 \times 8 = 48$ watts (assuming $i_a = i_{a-target}$). However, to maintain the DC-motor armature voltage at $v_a = 6$ volts, the transistor voltage from its emitter *e* to collector *c* will be approximately $v_{ce} \approx 6$ volts. As a result, the power consumption on the transistor is $P_T \approx v_{ce} i_a = 48$ watts, which will become the heat to get rid of. Therefore, a large heat sink will be needed, leading to a very large driving circuit package.

Now, consider the PWM driving circuit on the right of Figure 4.57. Assuming that the transistor saturated voltage (fully on) drop v_{ce} between transistor emitter *e* and collector *c* is $V_{ces} = V_l = 0.1$ volt, the PWM voltage is $V = V_h - V_l = V_{bat} - 0.1 = 11.9$ volts. To achieve the target armature current of $i_{a-target} = 8$ A with $v_{EMF} = 2$ volts, the average analog voltage, v_{analog}, defined in equation 4.6.2, is 6 volts, with $R_a = 0.5\ \Omega$. The PWM duty-cycle α can now be calculated below.

$$\alpha = \frac{v_{analog}}{V} \times 100\% = \frac{6}{11.9} \times 100\% = 50.42\% \tag{4.6.4}$$

In this case, the power consumption on the transistor can be calculated as below.

$$P_T = i_a V_{ces} \times \alpha = 8 \times 0.1 \times 50.52\% = 0.404\ watts \tag{4.6.5}$$

which is 0.84% of the analog driving circuit (48 watts) shown on the left of Figure 4.57. With a 99.16% waste (heat) energy reduction, the PWM driving circuit is much more efficient; it is used for most of the power driving applications.

PWM driving circuit does have its disadvantage. That is, it generates unwanted EMF (electro-magnetic field) noise that could affect other electronic devices and be harmful to human beings. The top plot of Figure 4.58 shows a 10 Hz ($\tau_c = 0.1$ second) PWM signal with a 50% duty-cycle ($\tau_0 = 0.05$ second) and bottom-to-peak voltage of 1 volt. The bottom plot of Figure 4.58 displays the discrete Fourier transform results, where horizontal axis is frequency in Hz and vertical axis is 20 times the ten-based logarithm of the magnitude of the discrete Fourier transform (similar to Bode magnitude definition). We can see that besides the DC components of the signal related to the average voltage at 0 Hz, there are significant frequency components at 10, 30 50, 70, and 90 Hz, indicating significant EMF over a wide frequency spectrum. Therefore, the other electronic circuits much be able to handle EMF noise at these frequencies.

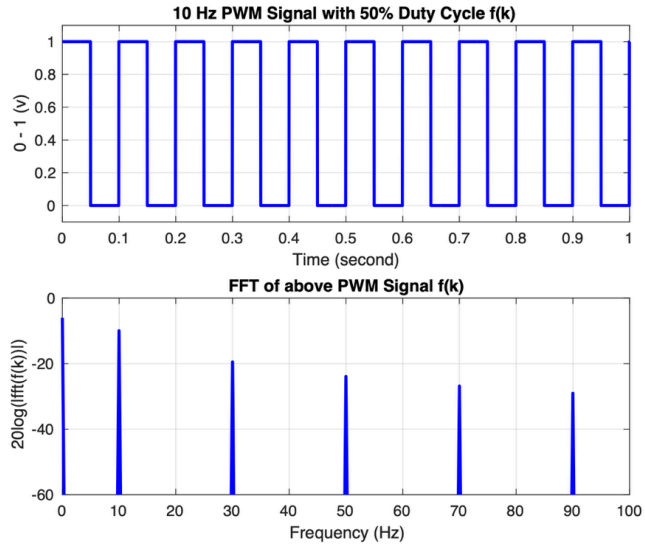


Figure 4.58: EMF effect of the PWM drive circuit.

Now that we've discussed the advantage and disadvantage of the PWM driving circuit, the question is how to generate a PWM signal based on the desired PWM duty-cycle and frequency from the information domain. There are two methods that can be used to generate the PWM signal utilizing the information in the information domain: digital and analog. The digital method uses a counter and a comparator; it outputs the PWM signal through a digital output channel of the microcontroller. The analog approach uses the D/A converter to generate an analog signal from the information domain; it uses a comparator to generate the PWM signal.

Figure 4.59 shows a diagram of how the PWM signal can be generated in the information domain digitally. The input on the left is a clock signal (for example, 1 GHz) to a digital counter (for example, a 10-bit counter). The counter output A is compared with the desired (given) duty-cycle in terms of a binary number B. When \(A

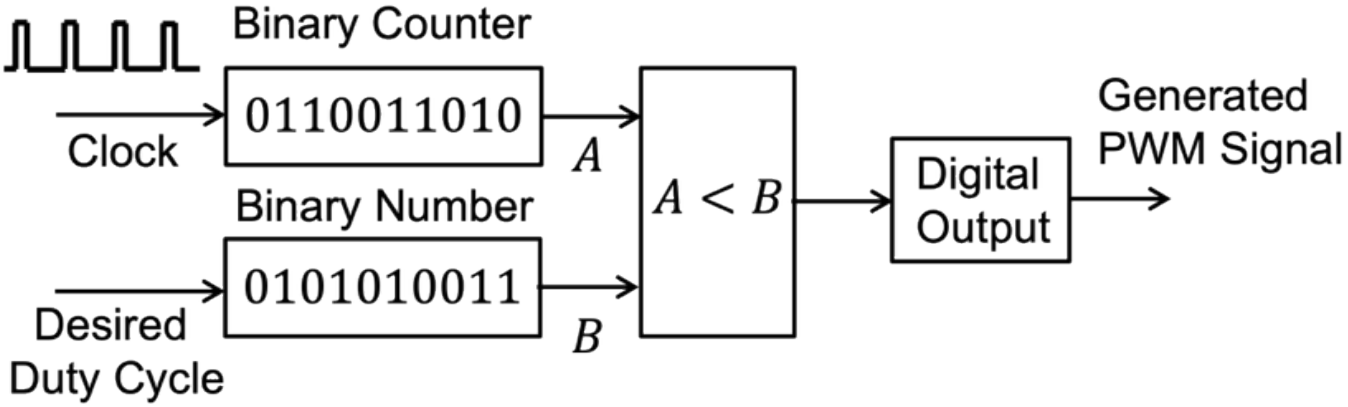


Figure 4.59: Digital PWM signal generation in the information domain.

Example 4.21

For this example, we will calculate the duty-cycle and frequency generated by the digital PWM generator shown in Figure 4.59. In this case, PWM signal frequencies are determined by the clock frequency (1 GHz) and counter size (number of bits, $N = 10$) as follows.

$$f_{PWM} = \frac{f_{clock}}{2^N} = \frac{10^9}{2^{10}} = 976,562.5\ Hz \approx 1\ MHz \tag{4.6.6}$$

Duty-cycle can be calculated using the following formula with the binary number given in Figure 4.59.

$$duty\ cycle = \alpha = \frac{\text{Binary Number}}{2^N} = \frac{2^8 + 2^6 + 2^4 + 2^1 + 2^0}{2^{10}} = \frac{339}{1024} = 33.1\% \tag{4.6.7}$$

Now, let's discuss how to generate the PWM signal using the analog method. First, the desired PWM duty cycle α is converted into an analog signal using the D/A converter in the microcontroller. If a five-volt D/A circuit is used, zero percent duty-cycle corresponds to 0 volt and 100 percent corresponds to 5 volts. If a 10-bits D/A converter is used, zero percent duty-cycle corresponds to number 0 and 100 percent corresponds to number 1023. For example, if the digital binary number is 0101010011, the D/A output is 1.655 volts. Non-zero percent duty-cycle corresponds to a number between 1 and 1023.

In Figure 4.60, a sinusoidal signal plus a constant of 2.1 is used as the reference signal for generating the PWM signal at the desired duty-cycle. To generate the PWM signal, a saw wave, which is an analog signal, needs to be generated at the frequency equal to the generated PWM signal frequency. For example, this saw wave signal can be generated using an LM555 integrated circuit. The five-volt saw wave (see the blue line on the top plot of Figure 4.60) and 1.655 volts D/A output lead to a 33.1% duty-cycle PWM signal at 10 Hz. The red line on the top plot of Figure 4.60 shows the shifted sinusoidal reference signal defined below.

$$f(t) = 2.1 + 2\sin(2\pi t) \tag{4.6.8}$$

The blue line on the bottom plot of Figure 4.60 shows the resulting PWM signal. We can see that as the shifted-sinusoidal signal level increases, the PWM signal duty-cycle increases, and vice versa. Especially, when the saw wave value is less than the sinusoidal signal, the PWM signal is high.

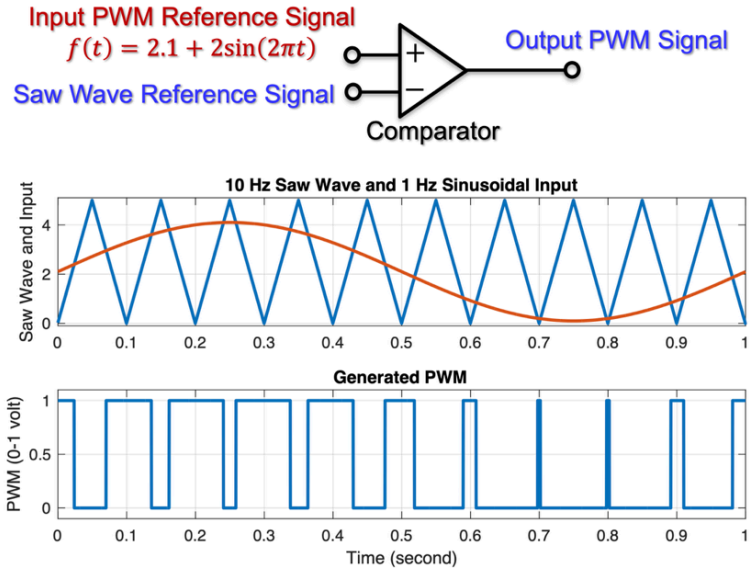


Figure 4.60: PWM signal generation using analog method.

A frequency modulated (FM) signal is a pulse train with a fixed duty cycle but variable frequency. This type of signal is often generated by digital sensors that produce high and low electronic voltage or current source signals. For example, a hall-effect rotational position sensor outputs an FM signal. Note that the current-source (with a desired current) signal is robust to ground noises. The top plot of Figure 4.61 shows a typical frequency modulated signal at 10 Hz with a bottom-to-peak magnitude of 5 volts and 50 percent duty cycle. The information contained in an FM signal is its varying frequency in the physical domain.

Now, the question is how to sample the frequency information into the information domain so that the microcontroller can process the sensor information. Two methods can be used to read the frequency information in the information domain through a digital counter: direct counting and clock counting, addressed below.

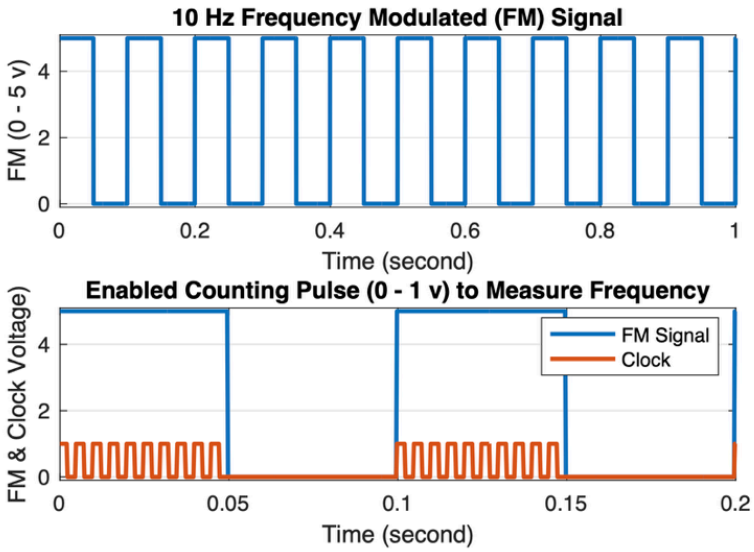


Figure 4.61: Sample frequency modulated (FM) and its counting method.

We will address the direct counting method first; see the left graph of Figure 4.62. We can see that the FM pulse train is input to a digital counter in the information domain through a digital input channel of the microcontroller. This digital counter will be read as $c(k)$ at a fixed sample period of T (second), where the FM signal period T_{FM} (second) is much smaller than the sample period T. The FM signal frequency averaged over one sample period can now be calculated using the formula below.

$$f\left(k\right)=\frac{c\left(k\right)-c(k-1)}{T} \tag{4.6.9}$$

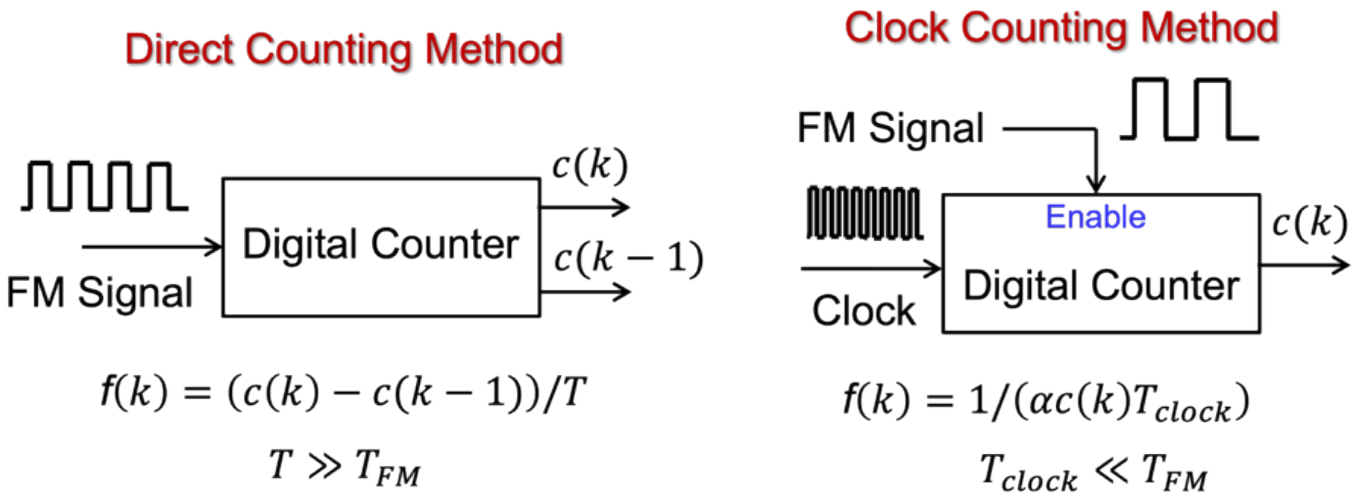


Figure 4.62: Calculating FM frequency using direct and clock counting methods.

The right graph of Figure 4.62 shows the clock counting method (see also the bottom plot of Figure 4.61). We can see that the FM pulse train is used to enable counting the clock pulse train when the signal is high. When the signal is low, the counting is disabled and counter value can be read as $c(k)$ at a fixed sample period of T_{FM} (second), where the FM signal period T_{FM} (second) should be much larger than the clock period T_{clock}. That is, the clock frequency should be much larger than the FM signal frequency. Then, we can calculate the FM signal frequency using the formula below, where α is the duty-cycle of FM signal.

$$f\left(k\right)=\frac{1}{\alpha c\left(k\right)T_{clock}} \tag{4.6.10}$$

If the duty cycle is 50%, $\alpha=0.5$.

Remark 4.7

The direct counting method has a relatively lower update rate of the FM signal than the clock counting method. If a high updating rate is required, the clock counting is recommended. However, clock counting may introduce certain signal fluctuations due to the runoff error of counting.

Example 4.22

Consider an FM signal with 40% duty cycle. For the direct counting method, assume that the sample frequency is 10 Hz ($T = 0.1$ second). If the counter outputs $c(k-1) = 90$ and $c(k) = 110$, the FM signal frequency can be calculated using equation 4.6.9 as follows.

$$f(k) = \frac{c(k) - c(k-1)}{T} = \frac{110 - 90}{0.1} = 200\ Hz \tag{4.6.11}$$

For the clock counting method, assume that the clock frequency is 10 kHz ($T_{clock} = 0.0001$ second). If the counter output $c(k) = 20$, the 40% duty-cycle FM signal frequency can be calculated using equation 4.6.10 as follows.

$$f(k) = \frac{1}{\alpha c(k) T_{clock}} = \frac{1}{0.4 \times 20 \times 0.0001} = 1250\ Hz \tag{4.6.12}$$

Section Summary

This section addresses two classes of signals: pulse-width modulated and frequency modulated signals. They are digital in nature but represented by analog signals in the physical domain. Here, the pulse-width modulated signal is a square pulse train with fixed frequency, but with varying duty-cycle [15]. The frequency modulated signal [16] is a fixed duty-cycle signal (such as square waive or sinusoidal signal) with varying frequency. Note that the frequency modulated sinusoidal signal is often used to transmit the radio signal. For square-wave pulse-width modulated (PWM) signals, converting the signal from the physical domain into the information domain can be realized by counting the clock pulse activated by the square-wave. Transferring the PWM signal from the information domain to the physical domain can be completed using a fixed frequency saw-wave generator and a comparator. Note that the pulse-width modulated (PWM) signal is often used to control DC motors to reduce the energy consumption of the amplifier circuit.

Section Keywords

Practice Problems

1. For a 1 Hz pulse-width modulated (PWM) signal with 0.3 second high (5 volts) and 0.7 second low (0 volt), what is its duty-cycle from the list below? Select the correct answer.
   1. $\alpha = 0.3/0.7\% = 42.86\%$
   2. $\alpha = 30\%$
   3. $\alpha = 70\%$

1. For the digital PWM generator shown in Figure 4.59, if the clock frequency is 0.1 GHz with an 8-bit counter ($N = 8$), what is the PWM signal frequency?
2. For the clock counting method, assume that the clock frequency is 20 kHz ($T_{clock} = 0.00005$ second). If the counter output $c(k) = 40$, find the FM signal frequency for a given duty-cycle of 20%.

This page titled 4.6: Pulse-Width and Frequency Modulated Signals is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

4.7: Advanced Analog and Digital Filter Design and Implementation

Learning Objectives

After completing this section, you will be able to:

1. Utilize the Butterworth low- and high-pass filter design techniques.
2. Provide the realization circuit for the analog Butterworth filter.
3. Understand the Chebyshev filter design technique.
4. Describe the characteristics of the moving-average filter.

In Section 4.4, we introduced both analog and digital filter concepts along with their realizations. In this section, we will address advanced analog filter design techniques such as Butterworth and Chebyshev filters. We will also conduct frequency domain analysis for these filters along with their implementation. We will in turn discuss the bilinear transformation from s domain to z domain so that the discrete-time equivalent filters can be generated. Finally, we will address the moving-average filter and its frequency response.

We'll start with the analog Butterworth filter and its design. A Butterworth filter is a type of signal processing filter designed to have a frequency response as flat as possible in the passband. Hence, the Butterworth filter is also known as a "maximally flat magnitude filter." It was invented in 1930 by the British engineer and physicist Stephen Butterworth [22]. Note that any filter design is optimal in a certain sense. The choice of optimality determines the nature of the design. Butterworth filters are optimal in the sense of having a maximally flat amplitude response, measured by Taylor expansion.

We will discuss the low-pass and high-pass Butterworth filters only in this section because both band-pass and band-kill (notch) filters can be formed using serially and parallel connected low-pass and high-pass filters, respectively (see Section 4.4). The low-pass Butterworth filter with cutoff frequency ω_c is an all-pole filter without any zero in its transfer function. The Nth odd order transfer function of the Butterworth filter is in the following format.

$$F_{BL}(s)=\left(\frac{\omega_c}{s+\omega_c}\right)\prod_{i=1}^{\frac{N-1}{2}}\frac{\omega_c^2}{s^2+2\cos(\theta_i)\ \omega_c s+\omega_c^2},\theta_i=i\frac{\pi}{N}\left(i=1,\ 2,\ \cdots,\ \frac{N-1}{2}\right) \tag{4.7.1}$$

The Nth even order transfer function of the Butterworth filter is in the following format.

$$F_{BL}(s)=\prod_{i=1}^{\frac{N}{2}}\frac{\omega_c^2}{s^2+2\cos(\theta_i)\ \omega_c s+\omega_c^2},\theta_i=\frac{(i-0.5)\pi}{N}\left(i=1,\ 2,\ \cdots,\ \frac{N}{2}\right) \tag{4.7.2}$$

We can see that poles of a low-pass Butterworth filter are those stable ones evenly distributed on the circle of the complex plane with radius of ω_c. Figure 4.63 shows pole locations of both third order and forth order low-pass Butterworth filters with $\omega_c=1$. The red-star represents the Butterworth filter $F_{BL}(s)$ pole locations (the third order on the left and fourth order on the right). The green-star represents the filter $F(-s)$ pole locations. These poles for both $F_{BL}(s)$ and $F_{BL}(-s)$ are evenly distributed on the circle with radius of $\omega_c=1$. The poles for the Butterworth filter $F_{BL}(s)$ are stable on the open left half plane (OLHP) and those for $F_{BL}(-s)$ are unstable on the open right half plane (ORHP).

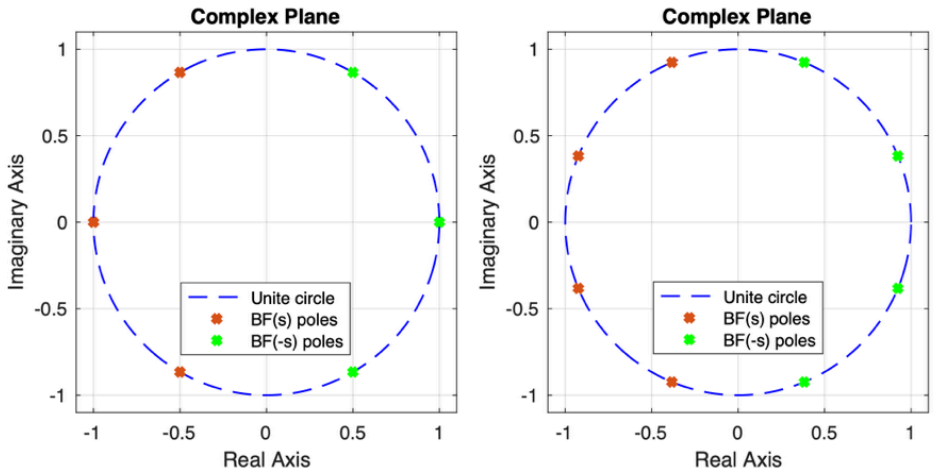


Figure 4.63: The 3rd and 4th order Butterworth filter pole locations.

With some derivations based on the pole locations of the Butterworth filters, we can show that the magnitude of the Nth low-pass Butterworth filter with cutoff frequency ω_c is in the form below.

$$|F_{BL}(j\omega)| = F_{BL}(j\omega)\,F_{BL}(-j\omega) = \frac{1}{1+\left(\frac{\omega}{\omega_c}\right)^{2N}} \tag{4.7.3}$$

which is due to the following fact.

$$F_{BL}(s)\,F_{BL}(-s) = \frac{1}{1+\left(\frac{-s^2}{\omega_c^2}\right)^{N}} \tag{4.7.4}$$

By setting the denominator's polynomial equal to zero, we have the following.

$$\frac{-s^2}{\omega_c^2} = (-1)^{\frac{1}{N}} = e^{j\frac{(2i+1)\,\pi}{N}}, i = 0,\ 1,\ 2,\ \cdots,\ N-1 \tag{4.7.5}$$

Hence,

$$s_i = \omega_c e^{j\frac{\pi}{2}} e^{j\frac{(2i+1)\,\pi}{2N}}, i = 0,\ 1,\ 2,\ \cdots,\ N-1 \tag{4.7.6}$$

Pole location equation 4.7.6 validates transfer functions 4.7.1 and 4.7.2 of odd- and even-order low-pass Butterworth filters, along with the magnitude equation 4.7.3.

Figure 4.64 shows the low-pass Bode magnitudes of Butterworth filter frequency responses as a function of order $N = 1,\ 2,\ 3,$ 4, and 5 with a cutoff frequency $\omega_c = 1$. Due to the characteristics of the Bode magnitude plot, the horizontal axis is in a scale of 10-based logarithm while the vertical axis is in a scale of 20 times 10-based logarithm magnitude (dB). When the frequency is below the cutoff frequency $\omega_c = 1$, the magnitude is very close to one. At the cutoff frequency, magnitude equals $\sqrt{2}/2$ (or $-3\ dB$). When the frequency is above the cutoff frequency $\omega_c = 1$, the magnitude decays at a rate of $20N\ dB$ per 10 times of frequency, where N is the order of the Butterworth filter. Therefore, as the order of the low-pass Butterworth filter increases, its frequency response approaches to the ideal low-pass filter (see Figure 4.28).

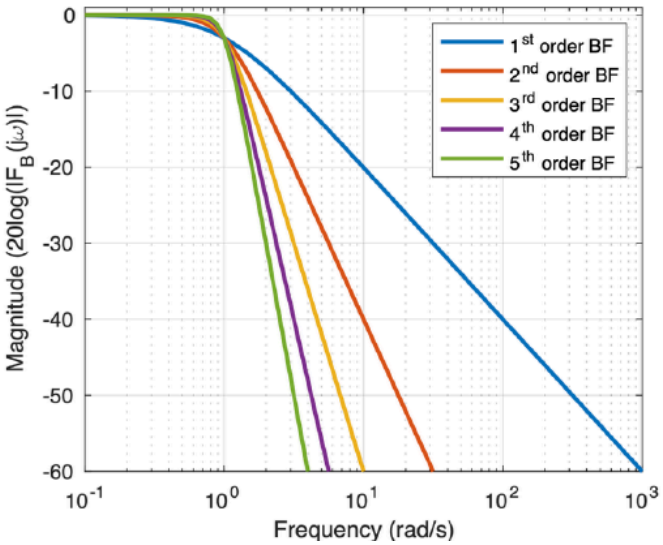


Figure 4.64: Bode magnitude plots for the 1st, 2nd, 3rd, 4th, and 5th order low-pass Butterworth filters.

The high-pass Butterworth filter transfer functions are provided in equation 4.7.7 for odd-order and equation 4.7.8 for even-order below. The high-pass Butterworth filter has the same pole locations as the low-pass one. However, the number of zeros equals the order of the Butterworth filter located at the origin of the complex plane.

$$F_{BH}(s) = \left(\frac{s}{s+\omega_c}\right) \prod_{i=1}^{\frac{N-1}{2}} \frac{s^2}{s^2 + 2\cos(\theta_i)\ \omega_c s + \omega_c^2}, \theta_i = i\frac{\pi}{N} (i = 1,\ 2,\ \cdots,\ \frac{N-1}{2}) \tag{4.7.7}$$

$$F_{BH}(s) = \prod_{i=1}^{\frac{N}{2}} \frac{s^2}{s^2 + 2\cos(\theta_i)\ \omega_c s + \omega_c^2}, \theta_i = \frac{(i-0.5)\pi}{N} (i = 1,\ 2,\ \cdots,\ \frac{N}{2}) \tag{4.7.8}$$

The pole locations of the high-pass Butterworth filter are the same as these for the low-pass one with ω_c in the low-pass Butterworth filter numerator replaced by s. As a result, we can show that the magnitude of the Nth order high-pass Butterworth filter with cutoff frequency ω_c is in the form below.

$$|F_{BH}(j\omega)| = F_{BH}(j\omega)F_{BH}(-j\omega) = \frac{\left(\frac{\omega}{\omega_c}\right)^{2N}}{1 + \left(\frac{\omega}{\omega_c}\right)^{2N}} \tag{4.7.9}$$

Figure 4.65 shows the Bode magnitudes of high-pass Butterworth frequency responses as order $N = 1,\ 2,\ 3,$ 4, and 5 with a cutoff frequency $\omega_c = 1$. When the frequency is above the cutoff frequency $\omega_c = 1$, the magnitude is very close to one. At the cutoff frequency $\omega_c = 1$, magnitude equals to $\sqrt{2}/2$ (or $-3\ dB$). When the frequency is below the cutoff frequency $\omega_c = 1$, the magnitude decays at a rate of $20N\ dB$ per one-tenth of frequency. Therefore, as the order of the high-pass Butterworth filter increases, its frequency response approaches the ideal high-pass filter (see Figure 4.32). Also, in the Bode plot scale, the high-pass Butterworth filter is a mirror of the low-pass one around the vertical axis of the cutoff frequency ω_c.

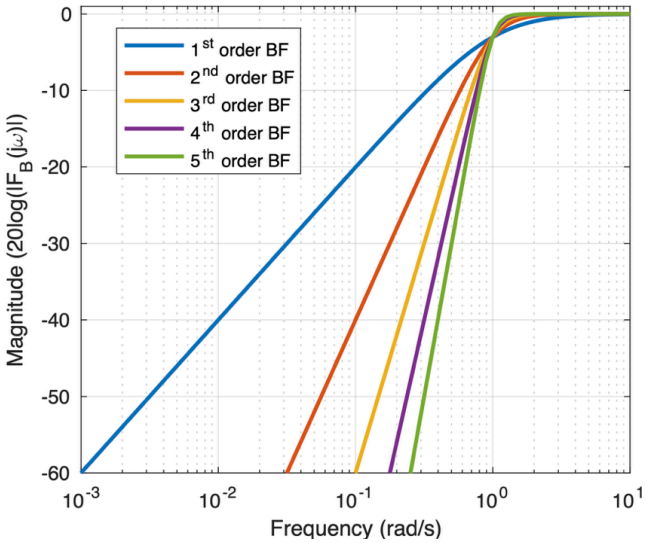


Figure 4.65: Bode magnitude plots for the 1st, 2nd, 3rd, 4th, and 5th order high-pass Butterworth filters.

Again, both band-pass and band-kill (notch) Butterworth filters can be constructed using serially and parallelly connected low-pass and high-pass Butterworth filters, respectively (see Section 4.4). Constructed band-pass filters will have the frequency characteristics of high-pass Butterworth filters at the frequency below ω_{c_1} and low-pass Butterworth filters at the frequency above ω_{c_2} with $\omega_{c_1} \ll \omega_{c_2}$, where ω_{c_1} and ω_{c_2} are defined in Section 4.4. On the other hand, constructed band-kill filters will have the frequency characteristics of low-pass Butterworth filters at the frequency beyond ω_{c_1} and high-pass Butterworth filters at the frequency below ω_{c_2} with $\omega_{c_1} \ll \omega_{c_2}$. In addition, over the frequency range between ω_{c_1} and ω_{c_2}, the magnitude of band-kill Butterworth filters will not be absolute zero, but small, and dependent on two factors: frequency difference $\omega_{c_2} - \omega_{c_1}$ and order of the Butterworth filter N, where the larger the factor, the smaller the magnitude.

Example 4.23

For this example, we will design a third-order Butterworth low-pass filter with cutoff frequency $\omega_c = 50$ rad/s. We will also provide an analog filter circuit calibrated for the designed third order Butterworth filter. Because it is an odd-order low-pass Butterworth filter, we will use equation 4.7.1 with $N = 3$, yielding the following filter transfer function, where θ_i can be calculated by equation 4.7.1.

$$F_{BL}(s) = \left(\frac{\omega_c}{s+\omega_c}\right)\frac{\omega_c^2}{s^2 + 2\cos(\theta_1)\ \omega_c s + \omega_c^2}, \theta_1 = \frac{\pi}{3} \tag{4.7.10}$$

Substituting the desired cutoff frequency yields the following low-pass Butterworth filter transfer function with cutoff frequency of $\omega_c = 5$ rad/s.

$$F_{BL}(s) = \left(\frac{50}{s+50}\right)\frac{50^2}{s^2 + 50s + 50^2} \tag{4.7.11}$$

In this case, the three poles of the Butterworth filter are $s_{1,3} = -25 \pm j43.30$ and $s_2 = -50$. We can show that the absolute values of these three poles are equal to 50. That is, all poles are on the circle with radius of $\omega_c = 50$ on the complex plane. Now the question is how to design an analog circuit with the transfer function 4.7.11. Figure 4.66 shows a typical third-order Butterworth low-pass filter using only resistors and capacitors.

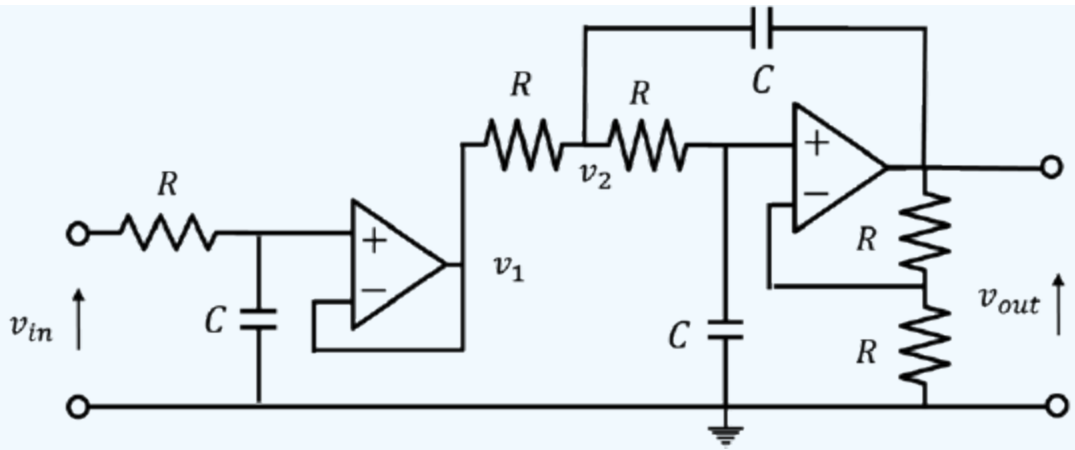


Figure 4.66: A third-order Butterworth low-pass filter circuit.

We can easily derive the transfer function from $V_{in}(s)$ to $V_1(s)$. The associated transfer function is shown below. Note that the operational amplifier on the left is an isolator with unit transfer function.

$$\frac{V_1(s)}{V_{in}(s)} = \frac{1}{1+RCs} \tag{4.7.12}$$

To derive the transfer function from $V_1(s)$ to $V_{out}(s)$, the Kirchhoff current law is applied to node v_2, resulting in the following.

$$\frac{V_1(s)-V_2(s)}{R} = \frac{V_2(s)-0.5V_{out}(s)}{R} + (V_2(s)-V_{out}(s))\,Cs \tag{4.7.13}$$

The resistance and capacitance low-pass filter at the negative input of the right operational amplifier leads to the following relationship.

$$0.5V_{out}(s) = \frac{1}{1+RCs}V_2(s) \rightarrow V_2(s) = 0.5(1+RCs)V_{out}(s) \tag{4.7.14}$$

Substituting $V_2(s)$ in equation 4.7.14 into equation 4.7.13, and reorganizing the resulting equation, leads to the following transfer function.

$$\frac{V_{out}(s)}{V_1(s)} = \frac{2}{1+RCs+(RC)^2s^2} \tag{4.7.15}$$

Now we are ready to assemble the transfer function from $V_{in}(s)$ to $V_{out}(s)$, using equations 4.7.12 and 4.7.15 as follows.

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{V_1(s)}{V_{in}(s)}\,\frac{V_{out}(s)}{V_1(s)} = \frac{V_1(s)}{V_{in}(s)} = 2\left(\frac{1}{1+RCs}\cdot\frac{1}{1+RCs+(RC)^2s^2}\right) \tag{4.7.16}$$

Note that equation 4.7.16 is a transfer function of the third order low-pass Butterworth filter with DC gain of 2 (see equation 4.7.1), where the cutoff frequency $\omega_c = 1/RC$. To achieve $\omega_c = 50$, there are an infinite number of combinations of resistance R and capacitance C values. Selecting a reasonable capacitance $C = 0.1\ \mu F = 10^{-7}\ F$, we can calculate the resistance as follows.

$$R = \frac{1}{\omega_c C} = \frac{1}{50\times 10^{-7}} = 200\ k\Omega \tag{4.7.17}$$

Now, we will briefly discuss the Chebyshev low-pass filter design [23], [24] and its high-pass filter, which is similar to the Butterworth filter case with the different denominator. Recall that the low-pass Butterworth filter has its main feature of the flattest frequency response over the passband. After the passband, the magnitude of the frequency response decays $20N\ dB$ every ten-time frequency (also called with $20N\ dB$ decay rate). The question is whether we can have a low-pass filter with the same order, but a fast decay rate. The Chebyshev low-pass filter is the one that has a fast decay rate after the cutoff frequency. The trade-off is that the frequency response magnitude is not as flat as that of the Butterworth filter over the passband. A low-pass Chebyshev filter (type I) with cutoff frequency ω_c is an all-pole filter without any zero in its transfer function. The Nth odd order transfer

function of the Chebyshev filter is in the following form.

$$F_{CL}(s) = \left(\frac{s_r}{s+s_r}\right) \prod_{i=1}^{\frac{N-1}{2}} \frac{s_i \overline{s}_i}{(s+s_i)(s+\overline{s}_i)} \tag{4.7.18}$$

and the Nth even order transfer function of the Butterworth filter is in the following format.

$$F_{CL}(s) = \prod_{i=1}^{\frac{N}{2}} \frac{s_i \overline{s}_i}{(s+s_i)(s+\overline{s}_i)} \tag{4.7.19}$$

Here, s_r is the real pole location, and s_i and $\overline{s}_i$ are the pair of complex conjugate poles. Note that all pole locations are determined based on θ_i defined in either equation 4.7.1 or equation 4.7.2 for Butterworth low-pass filters. Cutoff frequency ω_c, and the passband ripple parameter $\varepsilon > 0$ related to the oscillation magnitude of the frequency response over the passband will be discussed later.

From equations 4.7.18 and 4.7.19, we can see that the Chebyshev low-pass filter has the same structure as that of a Butterworth filter for a given filter order, but with different pole locations. Recall that the Butterworth filter poles are evenly distributed on the circle over the complex plane with its radius equal to cutoff frequency ω_c. However, these poles of the Chebyshev filter are located on the complex plane of an ellipse with its maximal radius r_1 and minimal radius r_2, satisfying the following ellipsoid equation.

$$1 = \frac{x^2}{r_2^2} + \frac{y^2}{r_1^2} \tag{4.7.20}$$

where x and y are variables associated with the real and imaginary axes, respectively. Before discussing how to find r_1 and r_2, we need to determine parameter ε, related to the maximal fluctuation of frequency response magnitude over the passband. The following equation defines the relationship between the maximal fluctuation δ_{max} and parameter $\varepsilon > 0$.

$$\delta_{max} = 1 - \frac{1}{\sqrt{1+\varepsilon^2}} \tag{4.7.21}$$

Let

$$\beta = \left[\frac{\sqrt{1+\varepsilon^2}+1}{\varepsilon}\right]^{(1/N)} \tag{4.7.22}$$

Then, maximal radius r_1 and minimal radius r_2 can be calculated using the following formulae.

$$r_1 = \omega_c \frac{\beta^2+1}{2\beta},\ r_2\ = \omega_c \frac{\beta^2-1}{2\beta}. \tag{4.7.23}$$

The Chebyshev filter complex pole locations can then be calculated by the following equations.

$$x_i = r_2 \cos\theta_i\ ,\ y_i\ = r_1 \sin\theta_i\ , i = 1,\ 2,\ \cdots,\ (N-1)/2 \text{ or } (N/2) \tag{4.7.24}$$

where θ_i are defined in 4.7.1 for odd order or equation 4.7.2 for even order Chebyshev filters. If N is odd, the real pole $s_r = r_2$.

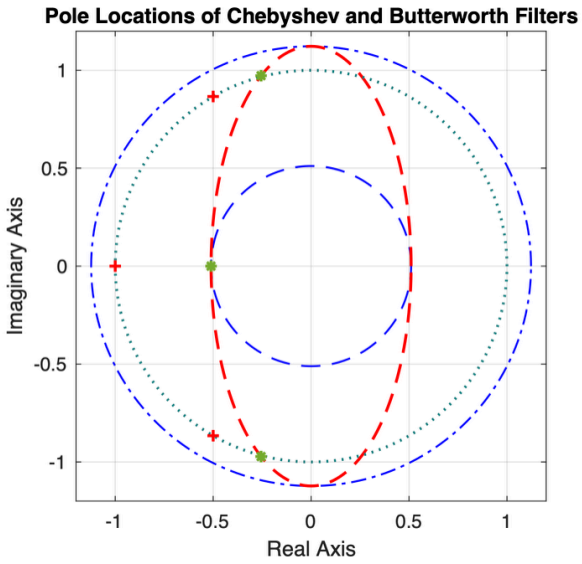


Figure 4.67: Comparison of pole locations for third-order Chebyshev and Butterworth filters.

Figure 4.67 compares the pole locations of third-order Chebyshev and Butterworth filters. The blue-dashed circle has a radius of r_2, the blue dash-dotted circle has a radius of r_1, the green-dotted circle has a radius of cutoff frequency ω_c, the red-dashed ellipse is formed by a minimal radius of r_2 and maximal radius of r_1, the red-pluses mark the pole locations of the Butterworth filter, and the green-stars mark the pole locations of the Chebyshev filter.

We can see that the pole locations of the Chebyshev and Butterworth filters are quite different. For the real axis pole, the Chebyshev pole ($-r_2$) is located closer to origin of the complex plane than the Butterworth one (ω_c). In other words, the real-axis pole of Chebyshev has a narrower bandwidth than the Butterworth pole. For the pair of complex poles, Chebyshev poles have less damping and larger frequency magnitude than the Butterworth ones. Note that the Butterworth filter complex poles have a damping ratio of 0.5 so that the frequency response will have a small peak before the cutoff frequency. However, the magnitudes of the complex Chebyshev poles are slightly greater than that of the Butterworth filter, indicating that the complex pole frequency response of a Chebyshev filter has wider bandwidth than a Butterworth filter. Example 4.24 provides detailed information for designing a third-order Chebyshev filter and comparing a Chebyshev filter frequency response with the Butterworth response.

Example 4.24

This example demonstrates the design process for a third-order Chebyshev filter with its cutoff frequency $\omega_c = 1$. Before getting into the design process, we need to determine the parameter ε associated with the allowed frequency response ripple magnitude δ_{max} over the passband. Based on equation 4.7.21, parameter ε can be calculated using the following equation for $\delta_{max} = 0.1$ (10% ripple).

$$\varepsilon = \sqrt{\frac{1}{(1-\delta_{max})^2} - 1} = \sqrt{\frac{1}{(1-0.1)^2} - 1} = 0.4843 \tag{4.7.25}$$

Then, parameter β can be calculated as below.

$$\beta = \left[\frac{\sqrt{1+\varepsilon^2}+1}{\varepsilon}\right]^{(1/N)} = \left[\frac{\sqrt{1+0.4843^2}+1}{0.4843}\right]^{(1/3)} = 1.6335 \tag{4.7.26}$$

Now, we are ready to calculate the maximal radius r_1 and minimal radius r_2 as follows.

$$\begin{aligned} r_1 &= \omega_c \frac{\beta^2+1}{2\beta} = 1 \times \frac{1.6335^2+1}{2 \times 1.6335} = 1.1229 \\ r_2 &= \omega_c \frac{\beta^2-1}{2\beta} = 1 \times \frac{1.6335^2-1}{2 \times 1.6335} = 0.5107 \end{aligned} \tag{4.7.27}$$

The pole locations of the third-order Chebyshev filter can be calculated using equation 4.7.23 with $\theta_1 = 60°$. Then we have

$$s_r = r_2 = 0.5107 \tag{4.7.28}$$

and

$$s_{1,2} = 0.2553 \pm j0.9724 \tag{4.7.29}$$

Therefore, the designed transfer function of the third-order Chebyshev filter is shown as follows.

$$F_{CL}(s) = \frac{0.5107}{s + 0.5107} \cdot \frac{\overbrace{\frac{1.0108}{s^2 + 0.5107s + 1.0108}}^{\frac{(0.2553^2 + 0.9724^2)^2}{s^2 + 2 \times 0.2553s + (0.2553^2 + 0.9724^2)^2}}}{} \tag{4.7.30}$$

Recall that the third-order Butterworth filter with cutoff frequency of $\omega_c = 1$ has the following transfer function

$$F_{BL}(s) = \frac{1}{s+1} \cdot \frac{1}{s^2 + s + 1} \tag{4.7.31}$$

Therefore, the pole locations are quite different. The frequency response magnitudes of both the Chebyshev and Butterworth filters are plotted in Figure 4.68 for comparison purpose.

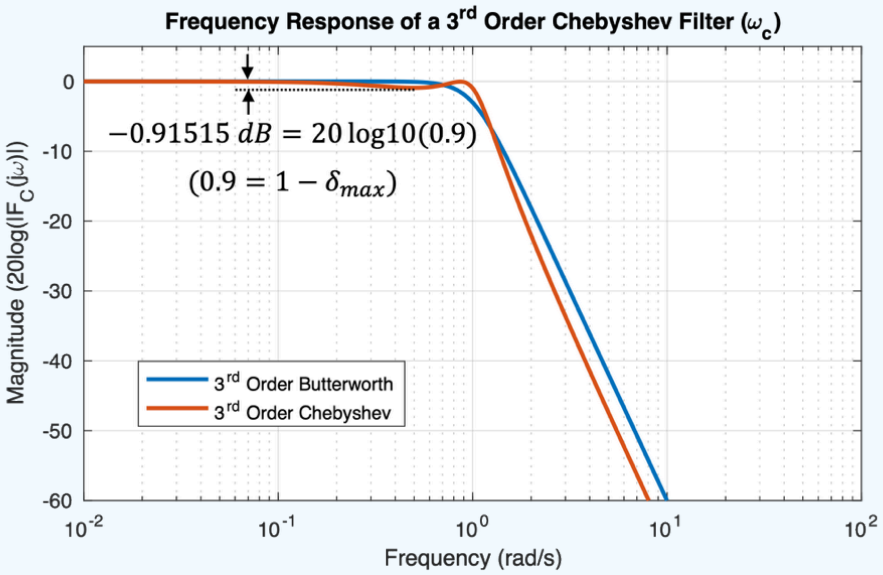


Figure 4.68: Bode frequency response comparison of third-order Chebyshev and Butterworth filters.

Figure 4.68 plots the Butterworth filter frequency response in blue and the Chebyshev filter one in red. We can see that the Chebyshev filter frequency response has a steeper decay rate after the cutoff frequency than the Butterworth one, validating the early claim. On the other hands, compared with the Butterworth filter frequency response over the passband, the Butterworth frequency response (blue line) is flat while the Chebyshev one (red line) fluctuates around a gain equal to one with a maximal dip of 10%. This validates selected design parameter ε to have the maximal fluctuation magnitude of $\delta_{max} = 0.1$ (10%). However, around cutoff frequency, the Chebyshev response ($-1\ dB$) is closer to one ($0\ dB$) than the Butterworth one ($-3\ dB$), indicating that the Chebyshev filter has a sharper cutoff frequency response than the Butterworth one.

Remark 4.8

Because the Chebyshev filter has the same structure as the Butterworth one, the analog realization circuit of the third-order Chebyshev filer is the same as the one shown in Figure 4.66 with a different choice of resistor and capacitor values. The first stage resistor and capacitor values are especially different from those in the second stage.

We've addressed the analog high-order filter design techniques of Butterworth and Chebyshev filters and compared their frequency responses. There are other filters, such as elliptic and Bessel filters, discussed in detail in references [24] and [25].

One traditional way to create the associated digital filter is by discretizing the analog filter transfer function. Using the zero-order sample and hold, a continuous-time transfer function in the s domain can be converted to its associated discrete-time transfer function in the z domain (i.e., $z = e^{st}$). Next, we will introduce another way to discretize the designed analog filter using the so-called bilinear transform.

The bilinear transform, also called Tustin's method [26], is used mainly in digital signal processing and discrete-time control applications. It helps convert an LTI (linear time-invariant) continuous-time system transfer function $F(s)$ to a discrete-time transfer function $F_d(z)$. This discrete-time transfer function can be implemented in the information domain in the form of a software algorithm, and vice versa. Note that the relationship between continuous- and discrete-time transfer functions can be approximated by the following equation based on Tayler expansion, as discussed in Section 2.5.

$$z = e^{sT} = \frac{e^{sT/2}}{e^{-sT/2}} \approx \frac{s + T/2}{s - T/2} \tag{4.7.32}$$

Its reversed relationship is shown below.

$$s \approx \frac{2}{T} \cdot \frac{z-1}{z+1} \tag{4.7.33}$$

where parameter T is the sample period, and z^{-1} is a one-step delay operator. This bilinear transform preserves the transfer function stability and maps every point on the complex plane s (continuous-time domain) Ha(jωa) to the corresponding point on the complex plane z (discrete-time domain) in the same way as the map $z = e^{sT}$. However, their frequency responses, which use $z = e^{sT}$ or $z = (s + T/2)/(s - T/2)$ Hd(ejωdT), are quite different. Example 4.25 demonstrates this point. At low frequency, both transform approaches match the original analog filter characteristics well.

Example 4.25

For this example, consider a first order analog filter with cutoff frequency $\omega_c = 50$ rad/s as shown below.

$$F(s) = \frac{50}{s + 50} \tag{4.7.34}$$

Set the sample period equal to $T = 0.002$ second with sample frequency of $1000\pi \gg 50$ rad/s, which satisfies the Nyquist-Shannon sample theorem. Using the map ($z = e^{sT}$) of zero-order sample and hold leads to the following discrete-time transfer function.

$$F_d(z) = \frac{0.09516}{z - 0.9048} \tag{4.7.35}$$

Using the bilinear transform 4.7.33 results in the following discrete-time transfer function.

$$F_{db}(z) = \frac{0.1z + 0.1}{2.1z - 1.9} \tag{4.7.36}$$

We can see that both transfer functions $F_d(z)$ and $F_{db}(z)$ have the same order (first) and DC gain (one).

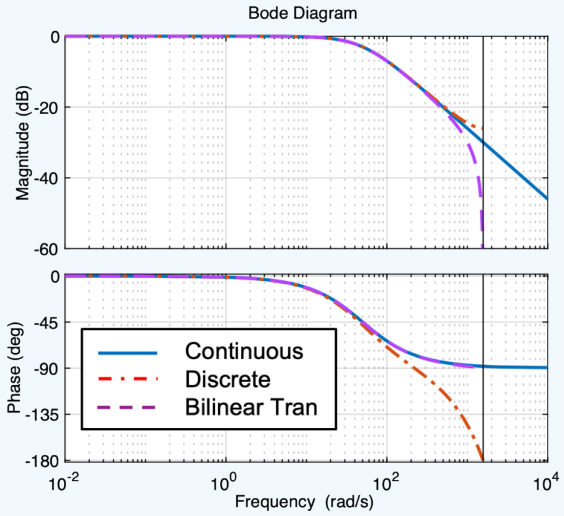


Figure 4.69: Bode frequency responses of continuous, discretized, and bilinear transform models.

Figure 4.69 plots the frequency responses of the original analog filter, discrete filters $F_d(z)$ and $F_{db}(z)$, where the solid-blue line represents the analog frequency response, the red dashed-dotted line is the frequency response of $F_d(z)$, and the purple-dashed line is the frequency response of $F_{db}(z)$. The discrete-time frequency responses for $F_d(z)$ and $F_{db}(z)$ are plotted using the MATLAB Bode plot command that transfers the discrete-time transfer function back to continuous-time one using $z = e^{j\omega T}$.

Because the sample frequency is 1000π in rad/s, based upon the Nyquist-Shannon theorem, the maximum bandwidth for the discrete-time filter is half of the sample one at $500\pi = 1571$ rad/s (represented by the vertical black line on the top plot). All three frequency magnitude responses are plotted in the top graph of Figure 4.69. Using the analog magnitude frequency response as reference, we can see that at low frequency, say four times below the sample frequency, the three frequency magnitudes are almost identical. However, at the frequency close to half of the sample frequency the magnitude of the standard discretization ($z = e^{sT}$) is slightly beyond the reference (analog) and the bilinear transform magnitude drops quickly as the frequency approaches half of the sample frequency.

Three filter phase responses are plotted on the bottom graph of Figure 4.69. Similar to the magnitude case, at frequency four times below the sample frequency, these responses match well. But at the frequency close to half of the sample frequency, the standard discretization ($z = e^{sT}$) produces a very large phase lag, while the bilinear one is almost identical to the reference one (analog).

Remark 4.9

In Example 4.25, if the system sample frequency is at least four times the highest signal frequency, both discretization approaches (sample and hold $z = e^{sT}$ and bilinear $z = (s + T/2)/(s - T/2)$) have similar performances.

We will end this section by discussing a special discrete-time filter, called moving average filter. It is widely used in many engineering applications due to its simpleness and easiness to implement. An m-point moving average discrete-time filter can be expressed below mathematically.

$$F_{dma}(z) = \frac{Y(z)}{U(z)} = \frac{1 + z^{-1} + z^{-2} + \cdots + z^{-(m-1)}}{m} \tag{4.7.37}$$

In Figure 4.70, the calculation flow of an eight-point ($m = 8$) moving average filter is plotted, where eight memory units are used as a buffer with current input $u(k)$ on the top and the oldest input $u(k-7)$ at the bottom. Outputs of a buffer with eight memory units are summed and then divided by eight (m). This dividing can be done by shifting the binary number right by 3 bits. The advantage of the moving average filter is that no multiplication is needed if the number of moving average points is in the form of $m = 2^n$ (n is a positive integer). As a result, this filter can be used in low-cost microcontrollers where multiplication is expensive.

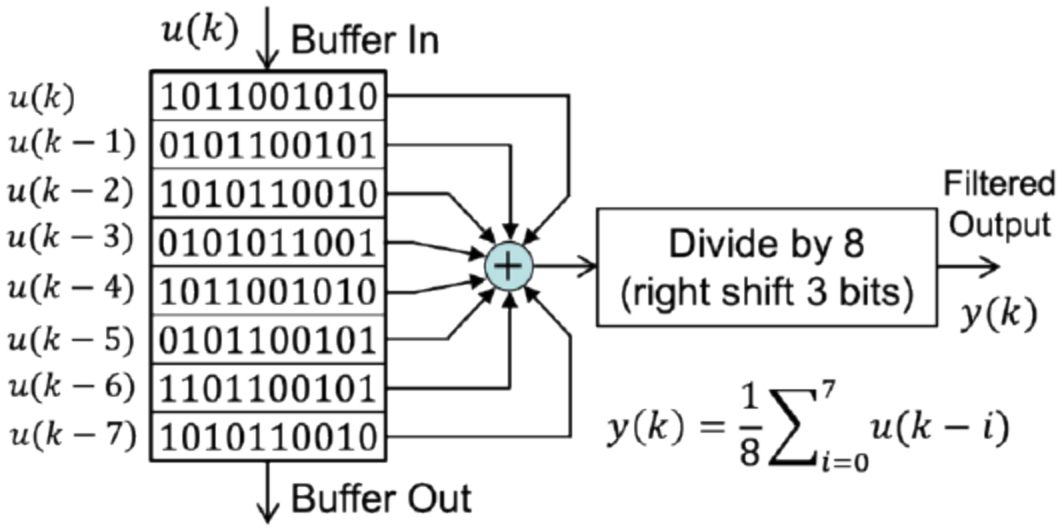


Figure 4.70: Moving average filter implantation in the information domain.

Example 4.26

This example will show the frequency response of a 24-point moving average digital filter with the following transfer function, where the system sample frequency is 500 Hz.

$$F_{dma}(z) = \frac{z^{23} + z^{22} + \cdots + z^1 + z^0}{24z^{23}} \tag{4.7.38}$$

Consider the analog filter defined in equation 4.7.34 with the same sample frequency of 500 Hz. Using the same approach as that in *Example 4.25*, the frequency response of the analog and moving average filters are plotted in *Figure 4.71* below.

We can see that the moving average filter magnitude is very close to the first order filter, with its magnitude way below the first order one at high frequency range. These valleys in the magnitude response of the moving average filter are mainly due to moving-average filter characteristics. For example, consider the sinusoidal periodic function below with its frequency at $1000\pi/24 = 130.9$ rad/s.

$$u(k) = \sin\left(\frac{2\pi k}{24}\right) \tag{4.7.39}$$

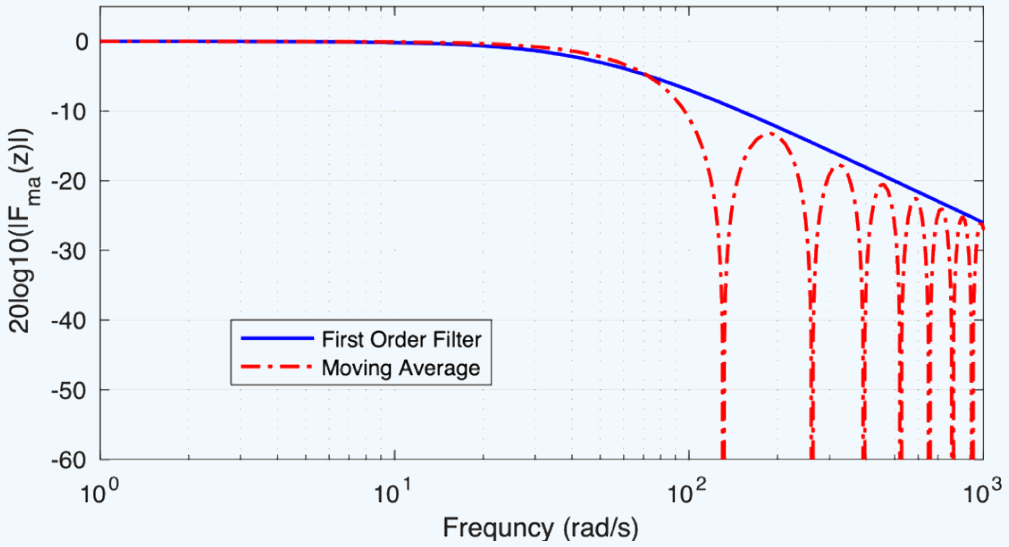


Figure 4.71: Moving average and first order filter frequency responses.

We can see that the filtered signal in this case will be zero because the sinusoidal signal above will repeat every 24 points with its 24-point average equal to zero. From Figure 4.71, we can see that at frequency of 103.9 rad/s, the magnitude of frequency response is zero. As a result, a moving average filter is especially good for getting rid of a periodic noise with fixed frequency.

Remark 4.10

To design a high order moving average filter, we may connect the moving average filters serially. We also recommended conducting frequency response analysis to understand its characteristics before using it.

Section Summary

This section addresses advanced analog and digital filter design methodologies that can be used to improve signal quality. Note that first and second order filters do not have a clean cutoff around the corner frequency. As a result, a certain amount of unwanted signal (noise) beyond or below the corner frequency passes through the filter, which is undesirable. To improve filtering technique, special filters, such as Butterworth and Chebyshev filters [17] and [23], are often designed and implemented. This section discusses how to design both analog and digital Butterworth and Chebyshev filters, along with implementation circuits for analog filters and software in z-transform (using bilinear transform). We also analyze digital filter such as the moving average.

Section Keywords

Practice Problems

1. Find a 16-point moving average digital filter transfer function.

This page titled 4.7: Advanced Analog and Digital Filter Design and Implementation is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

4.8: Chapter References

1. R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 12th ed., Englewood Cliffs, NJ, USA: Prentice Hall, 2011, ISBN-10: 0-13-602458-0.
2. M. J. M. Pelgrom, *Analog-to-Digital Conversion*, 4th ed., Helmond, Netherlands: Springer, 2022, ISBN-13: 978-3-0309-0808-9.
3. T. Waho, *Introduction to Analog-to-Digital Converters — Principle and Circuit Implementation*, Gistrup, Denmark: River Publishers, 2019, ISBN-13: 978-8-7702-2102-3.
4. D. F. Hoeschele, *Analog-to-Digital and Digital-to-Analog Conversion Techniques*, 2nd ed., Hoboken, NJ, USA: John Wiley and Sons, 1994, ISBN-13: 978-0-4715-7147-6.
5. R. van de Plassche, *Integrated Analog-to-Digital and Digital-to-Analog Converters*, Helmond, Netherlands: Springer, 1994, ISBN-13: 978-1-4613-6186-2.
6. J. O. Smith, *Mathematics of the Discrete Fourier Transformation (DFT)*, 2nd ed., Stanford, California, USA: W3K Publishing, 2007, ISBN-13: 978-0-9745-6074-8.
7. E. Chu, *Discrete and Continuous Fourier Transform: Analysis, Application and Fast Algorithm*, Boca Raton, FL, USA: CRC Press, 2019, ISBN-13: 978-0-3674-5269-8.
8. J. F. James, *A Student's Guide to Fourier Transforms with Applications in Physics and Engineering*, 3rd ed., Cambridge, MA, USA: Cambridge Univ. Press, 2011, ISBN-13: 978-0-5211-7683-5.
9. M. E. Van Valkenburg, *Analog Filter Design*, New York, NY, USA: Oxford Univ. Press, 1982, ISBN-13: 978-0-0305-9246-1.
10. T. Hinamoto and W. Lu, *Digital Filter Design and Realization*, Gistrup, Denmark: River Publishers, 2017, ISBN-13: 978-8-7935-1964-0.
11. U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays (Signals and Communication Technology),* 4th ed., Helmond, Netherlands: Springer, 2014, ISBN-13: 978-3-6424-5308-3.
12. H. Nyquist, "Certain topics in telegraph transmission theory," *Trans. AIEE,* vol. 47, no. 2, pp.617–644, Apr. 1928.
13. H. Dieter Lüke, "The origins of the sample theorem," *IEEE Commun. Mag.,* vol. 37, no. 4, pp. 106–108, Apr. 1999.
14. D. Clarke, Tobias Wrigstad, and James Noble, *Aliasing in Object-Oriented Programming: Types, Analysis and Verification (Lecture Notes in Computer Science, 7850),* Berlin Heidelburg, Germany: Springer-Verlag, 2013, ISBN-13: 978-3-6423-6945-2.
15. K. Hausmair, S. Chi, P. Singerl, and C. Vogel, "Aliasing-Free Digital Pulse-Width Modulation for Burst-Mode RF Transmitters," *IEEE Trans. on Circuits Syst. I, Reg. Papers,* vol. 60, no. 2, pp. 415-427, Feb. 2013.
16. S. Faruque, *Radio Frequency Modulation Made Easy*, 1st ed., Cham, Switzerland: Springer, 2017, ISBN-13: 978-3-3194-1200-9.
17. S. Hollos and J. Richard Hollos, *Passive Butterworth Filter Cookbook*, Longmont, CO, USA: Abrazol Publishing, 2021, ISBN-13: 978-1-8871-8742-8.
18. R. P. Agarwal and Donal O'Regan, *Ordinary and Partial Differential Equations*, New York, NY, USA: Springer, 2009, ISBN-13: 978-0-3877-9146-3.
19. IEEE Standard Association, IEEE Standard for Floating-Point Arithmetic (IEEE 754-2019), Accessed: December 29, 2025. [Online] Available: https://standards.ieee.org/ieee/754/6210/.
20. G. F. Franklin, J. David Powell, and Michael L. Workman, *Digital Control of Dynamic Systems*, 2nd ed., New York, NY, USA: Addison-Wesley, 1990, ISBN-10: 0-20-151884-8.
21. C. L. Philips and H. Troy Nagle, *Digital Control System*, 2nd ed., London, UK: Prentice Hall Int. Ltd., 1990, ISBN-10: 0-13-213596-5.
22. S. Butterworth, "On the Theory of Filter Amplifiers," *Exp. Wirel. and Wirel. Eng.,* vol. 7, pp. 536–541, Oct. 1930.
23. J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing*, New York, NY, USA: Macmillan Pub. Co., 1988, ISBN-10: 0-02-396810-9.
24. F. P. Miller, Agnes F. Vandome, and M. John, *Chebyshev Filter*, Saarbrücken, Germany: VDM Pub., Oct. 2010, ISBN-13: 978-6-1318-3469-1.
25. T. D. McGlone, Butterworth and Bessel Filters: *A Tutorial Overview,* CreateSpace Independent Publishing Platform, May 2016, ISBN-13: 978-1-5331-7280-8.
26. A. Tustin, "A method of analysing the behaviour of linear systems in terms of time series," *J. of Inst. of Elect. Eng. — Part IIA: Automat. Regulators and Servo Mech.*, vol. 94, no. 1, pp. 130–142, May 1947, doi: 10.1049/ji-2a.1947.0020.

Solutions to Practice Problems

1. The Laplace transform is

$$\mathcal{L}\left[f(t)\right]=\frac{s}{s^2+1}+0.5\frac{4}{s^2+16} \tag{4.8.1}$$

$$\mathcal{L}\left[f(t)\right]=\frac{s^3+2s^2+6s+1}{s^4+17s^2+16} \tag{4.8.2}$$

Using MATLAB, we can plot the magnitude of the frequency response below.

1.
2. The answer is 256 (c).
1. The answer is 4.805 volts (b).
2. The answer is 1 10000010 00010100...0 (c). Because the number is negative, leading to bit 1 equal to 1; the power factor is 2 ($-4.625=-2^2\times 1.15625$). Resulting bits 2–9 are 10000010 (binary representation of $127+2$). The first 6 binary digits of 0.15625 are 001010. Indicating bits 10–32 are 00010100...0.
1. The following are the z-transforms.
2. $\frac{2Tz}{(z-1)^2}$
3. $\frac{4z\left(z-\cos 2\ \right)}{z^2-2z\cos 2\ +1}$
4. $\frac{z}{z-1}+\frac{2e^{-2T}z\sin 2}{z^2-2e^{-2T}z\cos 2\ +e^{-4T}}$
1.

when $U\left(z\right)=1$, $Y\left(z\right)=\frac{z}{z-0.5}\ \rightarrow aT=0.6931$. Therefore,

$$y\left(k\right)=\mathcal{Z}^{-1}\left[Y(z)\right]=e^{-0.6931k} \tag{4.8.3}$$

1.

The discrete-time system matrices are shown below using MATLAB command c2d.

$$A_d=\left[\begin{array}{cc}1.0000 & 1.1052\\ 0.8187 & 0.9512\end{array}\right],\ \ B_d=\left[\begin{array}{c}-0.2519\\ 1.1052\end{array}\right],\ \ C_d=\left[\begin{array}{cc}1 & 0\end{array}\right] \tag{4.8.4}$$

and the discrete-time z-transfer function is shown below using MATLAB command ss2tf.

$$G\left(s\right)=C_d(zI-A_d)^{-1}B_d=\frac{-0.2519z+1.4610}{z^2-1.9512z+0.0464} \tag{4.8.5}$$

1. The time domain and frequency domain plots can be prepared using MATLAB. The following figure shows both time and frequency (magnitude) plots for P4.12. Two frequency magnitude peaks near 6.28 and 31.42 rad/s correspond to the time domain signal $\sin(2\pi kT_s)$ and $\cos(10\pi kT_s)$ with $T_s=0.005$ second. Note that the continuous-time signals $\sin(2\pi t)$ and $\cos(10\pi t)$ contain signals at angular frequencies of 6.28 and 31.42 rad/s.
2. The answer is $C=1\ \mu F$.
3. The answer is $R=1\ k\Omega$.
4. The answers are $R_H=100\ k\Omega$ and $R_L=200\ k\Omega$.
1. The answers are $R_L=100\ k\Omega$ and $R_H=200\ k\Omega$.
1.
2. Neither a) nor b) satisfy the Nyquist-Shannon sample condition; however, c) does because $f_s=120>2f_h=100$ Hz.
3. Because $f_s=8>2f_h=4$ Hz, the Nyquist-Shannon sample condition is satisfied. The rotational direction can be identified due to the following captured image.
1. a) aliasing exists because $f_s=250\ Hz<800\ Hz$., and b) there are two aliasing frequencies below sample frequency, and they are at 100 and 150 Hz with $l=-2$ and -1, respectively.
2. The answer is b) 30%.
1.
2.
3.

with poles located at $s_{1,2}=-70.71\pm j70.71$ and $|s_{1,2}|=100$.

1.

As a result, we have the following conditions.

$$\omega_c = \frac{1}{RC}, \quad \frac{2R_2 - R}{R_2} = \sqrt{2} \tag{4.8.6}$$

With $C = 0.1\ \mu F$ and $\omega_c = 100$ rad/s, R and R_2 can be calculated as follows.

$$R = \frac{1}{\omega_c C} = \frac{1}{100 \times 10^{-7}} = 100\ k\Omega, \quad R_2 = \frac{R}{2 - \sqrt{2}} = 170.71\ k\Omega \tag{4.8.7}$$

1.

So, we have

$$R_1 = \frac{1}{0.5107 \times C} = \frac{1}{0.5107 \times 10^{-5}} = 195.81\ k\Omega \tag{4.8.8}$$

For the second part of the circuit, we have

$$\frac{V_{out}(s)}{V_1(s)} = \frac{R + R_2}{R_2} \cdot \frac{1}{1 + \left(\frac{2R_2 - R}{R_2}\right) RCs + (RC)^2 s^2} \tag{4.8.9}$$

As a result, we have the following conditions.

$$R = \frac{1}{\sqrt{1.0108} \times C}, \quad \frac{2R_2 - R}{R_2} = 0.5107 \tag{4.8.10}$$

With $C = 10\ \mu F$, R and R_2 can be calculated as follows.

$$R = \frac{1}{\sqrt{1.0108} \times C} = \frac{1}{1.0054 \times 10^{-5}} = 100.27\ k\Omega, \quad R_2 = \frac{R}{2 - 0.5107} = 67.33\ k\Omega \tag{4.8.11}$$

The DC gain for the filter is

$$\frac{R + R_2}{R_2} = \frac{100.27 + 67.33}{67.33} = 2.489. \tag{4.8.12}$$

1.
2.

This page titled 4.8: Chapter References is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

5: MECHATRONIC SYSTEM COMPONENT- ACTUATORS

Learning Objectives

1. Describe various mechatronic actuation systems and their applications such as electromagnetic and electrohydraulic valve actuators. Understand operational principles of both linear and rotational DC motors.
2. Derive the torque and back EMF (electromotive force) coefficients from the DC-motor modeling contents in Chapter 2. Provide DC-motor output torque ripple as a function of a number of rotor winding poles. Define stall torque and no-load speed as a function of input armature voltage.
3. Introduce the low/high side and H-bridge DC-motor drive circuits. Provide the benefit of using the PWM (pulse-width-modulation) to drive the DC motor with significantly reduced drive power consumption (waste).
4. Describe the discrete electromagnetic linear actuation system with an example of an internal combustion engine camless valve actuation system.
5. Introduce the discrete electrohydraulic linear actuation system with an example of an internal combustion engine camless valve actuation system.
6. Provide the working principle of permanent magnetic synchronous motor (PMSM), its drive (inverter) circuit, and vector control of PMSM.

This chapter provides an introduction to actuators used in mechatronic systems, including DC motors and their drive circuits, electromagnetic and electrohydraulic valve actuators, and permanent magnetic synchronous motors.

5.0: Prelude

5.1: Mechatronic Actuator Background

5.2: Mechatronic Actuator Background

5.3: DC-Motor Drive Circuits

5.4: Discrete Electromagnetic Valve Actuation (EMVA)

5.5: Discrete Electrohydraulic Valve Actuation (EHVA)

5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control

5.7: Chapter References

This page titled 5: Mechatronic System Component- Actuators is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

5.0: Prelude

In Chapter 1, we discussed that signals within a mechatronic system can be divided into two groups: digital signals in the information domain and analog signals in the physical information domain. The digital information domain handles digital information by microprocessors whereas the physical domain deals with the analog information in the physical system dynamics, including the sensors and actuators. In this chapter, we will address these actuation subsystems in the physical domain, along with their driving electronics (amplifiers). Note that Section 4.6 looked at the communication between the information and physical domains through D/A (digital-to-analog) and A/D (analog-to-digital) conversions; PWM (pulse-width-modulated) signals were also discussed in detail. Now, in this chapter, we will address the application of the PWM signals to low/high-side and H-bridge drives.

Section 5.1 reviews various mechatronic actuation systems including mechanical, magnetic, hydraulic, and pneumatic systems; electromagnetic and electrohydraulic actuation will be discussed in detail. The section also provides examples of all mechatronic actuation systems, along with their typical practical applications, for DC motors, and both electromechanical and electrohydraulic valves. Actuation subsystems are key components of mechatronic systems. Their proper use is very important for achieving desired system performance in terms of response time and energy consumption.

Section 5.2 introduces DC-motor working principles. It addresses how to calculate the DC-motor torque coefficient (K_τ), a proportional gain of motor armature current, and the back EMF (electromotive force) coefficient (K_{EMF}), which is a proportional gain of motor speed; see Chapter 2 and reference [1] for DC-motor modeling. The section also covers concepts of maximum motor brake torque (stall torque) and maximum motor speed (no-load speed) as a function of input armature voltage. In addition, we introduce the effect of the gearbox connected to the DC motor in terms of geared output torque and speed. In general, assuming that there is no loss to gearbox efficiency, the energy outputs of both motor and gear box outputs remain the same. These can be used to derive torque and speed before and after gearbox.

Section 5.3 addresses the methodologies of driving the DC motor efficiently, including low- and high-side drive, PWM (pulse-width-modulated) low-side drive, and the widely used H-bridge drive based on the PWM signal. Note that the H-bridge drive enables us to generate the motor torque in two directions—positive and negative—which is very important for closed-loop control displacement (or velocity) with the DC motor as an actuator. We also discuss the associated DC-motor drive circuits, along with the advantages and disadvantages of these drive circuits.

The remaining three sections are at the graduate level. Section 5.4 introduces electromagnetic valve systems [2], [3], [4], where the electromagnetic intake and exhaust valve systems of internal combustion engines are used as an example. Note that the electromagnetic valve system converts the electric energy to the desired valve displacement using an electromagnetic actuator. Instead of generating the desired torque from a DC motor, the electromagnetic actuator generates a linear force that moves the valve mass to achieve the desired displacement or speed. A closed-loop control system is often needed to ensure that the valve displacement or speed follows the reference signal.

Next, Section 5.5 introduces electrohydraulic valve systems [5], [6], [7], [8], where electrohydraulic intake and exhaust valve systems of internal combustion engines are used as an example. Note that the electrohydraulic valve system requires a high-pressure hydraulic fluid source; electromagnetic controlled hydraulic on-off valves are used to control the displacement (or speed). Instead of generating the desired torque or force, the electrohydraulic on-off valve generates a proper linear (or rotational) displacement. This displacement turns a mechanical hydraulic valve on or off (or moves the valve linearly) to achieve the desired valve displacement. Similarly, a closed-loop control system is often needed to regulate the valve displacement accurately.

Last, Section 5.6 discusses the permanent magnet synchronous motor actuator. This type of motor is widely used in the powertrain of electric vehicles. It has attracted intensive research and development in the automotive and other industries. A permanent magnet synchronous electric motor [9], [10] is an AC electric motor with its steady-state shaft rotation speed synchronized with the control current frequency. A permanent magnet synchronous motor (PMSM) is similar to a brushless direct current (BLDC) motor with a permanent magnet rotor and wound stators. For example, a three-phase stator layout is designed to create sinusoidal flux distribution in the air gap. It makes the back electromotive force (BEMF) sinusoidal with motor rotation period equal to an integer number of the current oscillations frequency. The motor rotor with permanent magnets or electromagnets turns in step with the stator field at the same rate. As a result, it provides the second synchronized rotating magnet field. The section also addresses the drive hardware and principles of PMSM with a sample circuit [11], [12], as well as vector control of PMSM with PWM signal outputs to the inverter drive.

Orienting Questions

1. What are the DC-motor armature circuit, its maximum torque, and its speed?
2. What is an H-bridge DC-motor drive circuit?
3. How does an electromagnetic valve actuator work?
4. How does an electrohydraulic valve actuator work?
5. What is the working principle of a permanent magnetic synchronous motor?
6. How do we regulate a permanent magnetic synchronous motor with vector control?

This page titled 5.0: Prelude is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

5.1: Mechatronic Actuator Background

Learning Objectives

After completing this section, you will be able to:

1. Describe the fundamental working principles of a number of mechatronic actuators and their applications.
2. Distinguish the rotational and linear actuators and describe their differences and similarities.

In general, a mechatronic actuator converts one kind of energy (e.g., electrical energy) to another kind of energy (e.g., mechanical energy) based upon the command signal provided by the information domain. For example, a DC motor (see Chapter 2) converts the electrical energy (armature current) into rotational mechanical energy (motor shaft torque) based upon the desired torque (or armature current) controlled by the PWM (pulse-width-modulated) with its duty-cycle regulated by the command output from the information domain. A mechatronic actuator is defined below.

Definition 5.1

A mechatronic actuator is a device that generates force, motion, heat, or other forms of physical effects based upon a control command produced in the information domain, utilizing the supplied electric, hydraulic, or other forms of energies.

Remark 5.1

In a mechatronic system, actuators are essential for the mechanical components to track the desired trajectories such as displacement, speed profile, and temperature.

The majority of mechanical actuators can be divided into two main categories: rotational and linear. Rotational actuators are the most common and simple ones. Linear actuators are more expensive and complicated. Rotational actuators typically consist of a rotary shaft connected to an armature. A typical example is the brushed or brushless DC motor or step motor. The DC motor was briefly discussed in Chapter 2 regarding its modeling technique; it will be addressed in more detail in Sections 5.2 and 5.3. The permanent magnetic synchronous motor will be discussed in detail in Section 5.6. Next, however, we will discuss the step motor.

A step motor [13], also called stepper motor, is an electric motor that rotates in a series of small angular steps, activated by control pulses, instead of rotating continuously. Step motors are often operated in an open-loop mode without position feedback. They can be set to any desired position constrained by the motor step resolution. Note that the step-motor position can be rapidly increased or decreased, creating continuous rotation. The motor can also be controlled to hold its position at a given step.

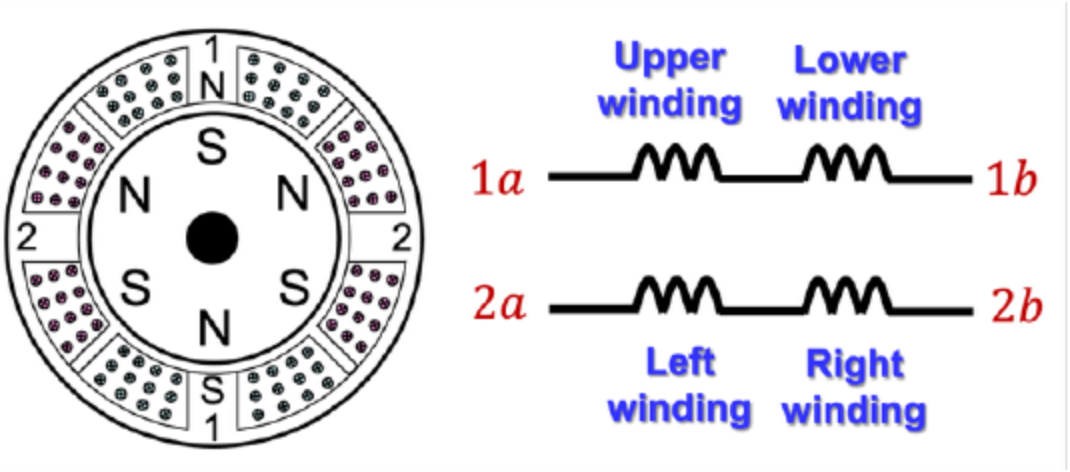


Figure 5.1: A schematic diagram of a bipolar step motor and associated windings.

Figure 5.1 shows a schematic diagram of a bipolar step motor and two associated winding pairs. These are the upper-lower (green winding #1) and left-right (red winding #2), where the upper-lower windings are energized. The generated magnetic polarities are shown just below and above the winding number of 1. Note that other forms of the step motors can be found in reference [14]. The step-motor drive circuit for a bipolar step motor is a little bit more complicated than the single one. This is because the H-bridge drive is needed so that the directions of applied voltages to windings #1 and #2 can be reversed in order to move the step motor in the desired direction. Section 5.3 will discuss the H-bridge drive in detail.

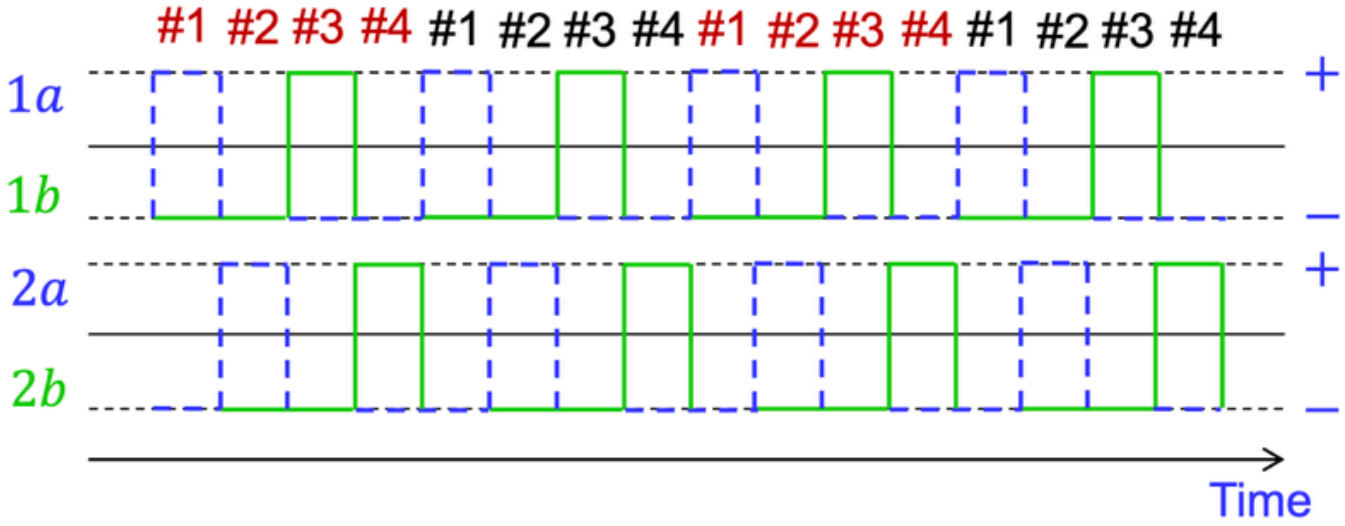


Figure 5.2: H-bridge control sequence of a single step for upper-lower and left-right winding pairs.

Figure 5.2 presents the drive control signals for a bipolar step motor with a single step. The blue-dashed lines represent the voltages applied to winding terminals *1a* and *2a*. The green lines are associated with the voltages applied to winding terminals *1b* and *2b*. Note that the horizontal axis represents the time increment from left to right.

The diagram in Figure 5.1 corresponds to the #1 location in Figure 5.3. Here, winding #1 is energized with positive voltage at *1a* and negative at *1b*, resulting in the N (north) and S (south) magnetic fields indicated in Figure 5.1. Meanwhile, winding #2 is not energized. Both terminals (*2a* and *2b*) of winding #2 are connected to the battery negative terminal, leading to no magnetic field formed.

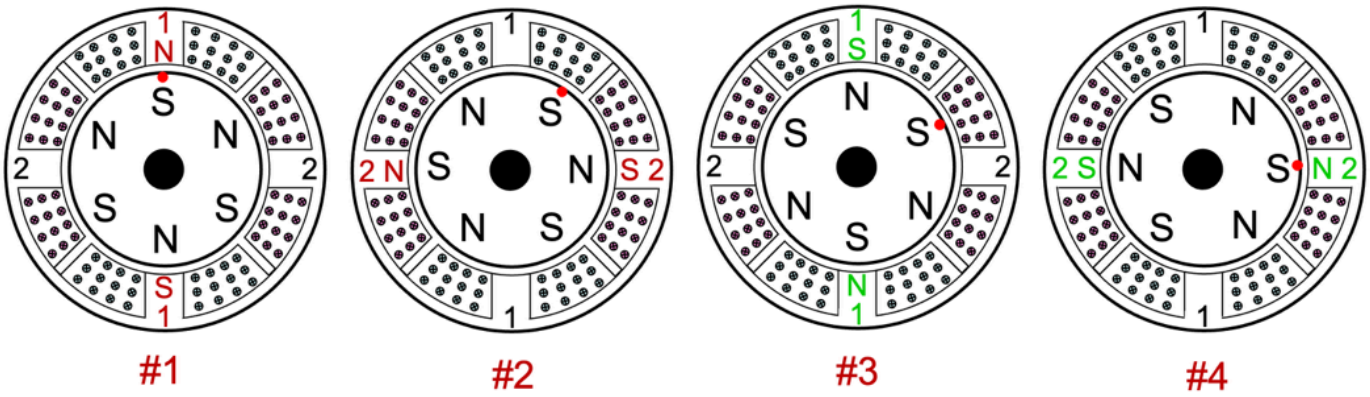


Figure 5.3: Clockwise step-motor rotation with the control signals shown in Figure 5.2.

Figure 5.3 shows a clockwise rotation of 90 degrees for the bipolar step motor when the pulse sequence #1 to #4, shown in Figure 5.2, is applied to windings #1 and #2. We can see that for the #1 plot in Figure 5.3, the upper-lower windings are energized, with positive connected to *1a* and negative to *1b*. Left-right windings are not energized, leading to a locked position shown with the red-dot on top. Next, in the #2 plot, the left-right windings are energized, with positive connected to *2a* and negative to *2b*. Upper-lower windings are not energized, rotating the rotor clockwise by 30 degrees, and resulting in a locked position shown with the red-dot turned clockwise by 30 degrees.

Then, in the #3 plot, the upper-lower windings are energized, with positive connected to *2b* and negative to *2a*. Left-right windings are not energized, rotating the rotor clockwise by another 30 degrees, and resulting in a locked position shown with the red-dot turned clockwise by another 30 degrees. Last, in the #4 plot, the left-right windings are energized with positive connected to *2b* and negative to *2a*. Upper-lower windings are not energized, rotating the rotor clockwise by 30 degrees, resulting in a locked position shown with the red-dot turned clockwise by yet another 30 degrees.

The associated control signals can be found in Figure 5.2 for the first four control pulses. By repeating these pulses, we can realize continuous rotation. The anticlockwise rotation can be realized by reversing the pulse sequence.

For this bipolar step motor with three pairs of permanent magnets, the resolution for each step is 30 degrees. To increase the step-motor step resolution, either the number of winding polarities or the number of permanent magnet pairs (or both) needs to be increased. On the other hand, a speed-down gear can also be used to improve the step-motor step resolution.

From this discussion, we can see that step motors are used mainly to control or regulate the rotational position. They should not be used for controlling the rotational speed in open-loop because it is difficult for the step motor to generate a torque to achieve a desired speed with unknown load, while a DC motor can regulate motor output torque using closed-loop speed control by regulating its armature current. One advantage of using a step motor to control rotational position is that there is no need for position feedback; the resulting rotational position can be found by counting the applied number of pulses. However, to reach the desired position with applied pulse sequence, the generated torque during the rotational transition needs to be greater than the load torque.

Example 5.1

Consider the bipolar step motor shown in Figure 5.1. Find the formula for calculating the step angle and then calculate the step motor's step angle. Note that the step angle of a given step motor can be calculated using the following formula.

$$SA = \frac{360^\circ}{2 \times n_p \times n_m} \tag{5.1.1}$$

where n_p is the number of polars and n_m is the number of permanent magnet pairs. For the step motor shown in Figure 5.1, the step angle SA can be calculated as follows, with $n_p = 2$ (bipolar) and $n_m = 3$ (three pairs of magnets).

$$SA = \frac{360^\circ}{2 \times 2 \times 3} = SA = 30^\circ \tag{5.1.2}$$

As a result, for the bipolar step motor shown in Figure 5.1, each stepping pulse leads to a thirty-degree rotation.

There are two main types of linear actuators. One uses a rotational actuator (DC motor or step motor) to drive a screw with a nut sliding on a rail (see Figure 5.4). The other employs the linear motor described in detail in this section.

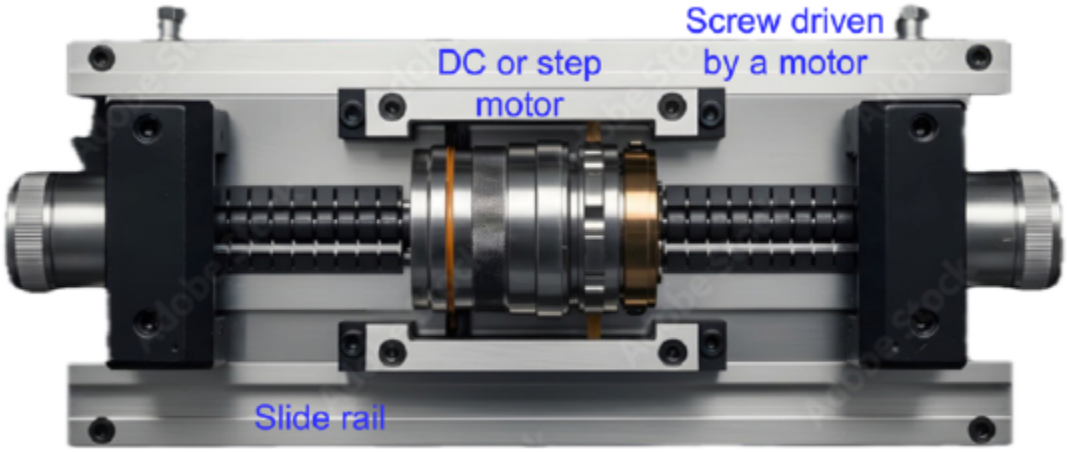


Figure 5.4: Linear actuator using a screw-nut and slider.

The advantages of the linear actuator shown in Figure 5.4 are its simple concept and the utilization of the rotational actuator that can be easily controlled for regulating speed and displacement. However, the disadvantages are twofold. First, the response speed could be slow due to the conversion of the rotational speed to a linear one. Second, the gap between the screw and nut could lead to unwanted regulation error. Example 5.2 clarifies the relationship between motor rotational angle and the linear motion of the nuts.

Example 5.2

Consider the linear actuator shown in Figure 5.4. Assuming that the screw has one start with a screw pitch p_s of one-twentieth inch (or 1.27 mm), find the gain that converts the motor rotational angle to the linear displacement. Also find the linear displacement if the motor rotates 180°. The gain can be calculated using the following formula.

$$G_{r2l} = \frac{p_s}{360^\circ} = \frac{1.27}{360^\circ} = 0.003528 \;\; mm/deg \tag{5.1.3}$$

If the motor rotates 180° (θ_r), the linear displacement d_l can be calculated as follows.

$$d_l = G_{r2l} \times \theta_r = 0.003528 \times 180 = 0.635 \; mm \tag{5.1.4}$$

Therefore, the resulting linear displacement is 0.635 (mm) if the motor rotates 180 degrees.

Let's now discuss linear motors, which we will refer to as linear actuators. A linear actuator is an electric motor that has its state and rotor "unrolled" to a straight line. As a result, instead of producing a rotational torque, it produces a linear force (displacement) along its length. To illustrate the working principle of a linear actuator, we will use a linear step motor as an example. A continuous linear actuator can be constructed in a similar way. However, to produce a smooth linear force, the control of actuation solenoids is more complicated in order to minimize the ripple of the actuation force. Linear position sensors are often required to have precise control. Details of linear actuators can be found in references [15] and [16].

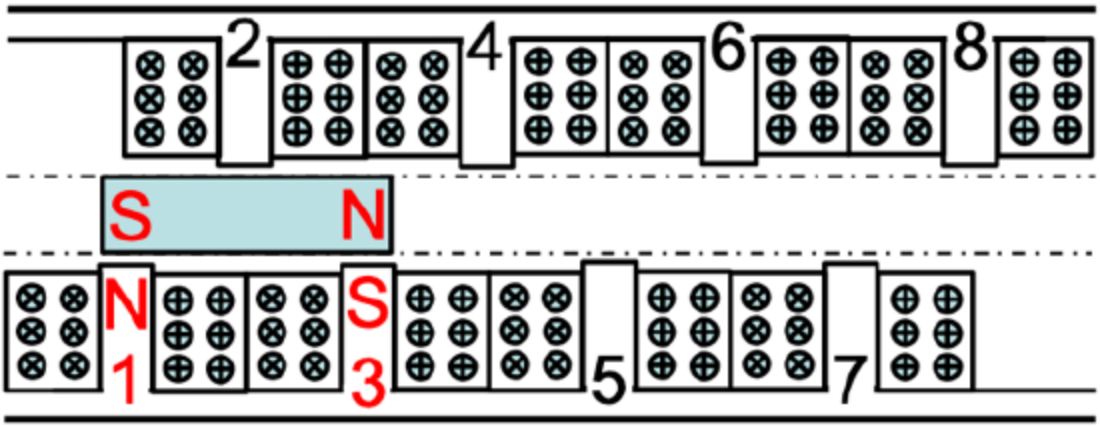


Figure 5.5: A schematic diagram of an eight-pole linear step motor.

Figure 5.5 shows an eight-pole linear step motor. It can be viewed as an unrolled rotational step motor in Figure 5.1 with double the number of poles. The rotational step motor shown in Figure 5.1 has four poles. In Figure 5.5, the magnet in the middle moves horizontally. The solenoid for each pole can be driven by an H-bridge circuit (see Section 5.3); therefore, each pole can be set to either North or South. Here, both solenoids #1 and #3 are energized (marked in red). The pole associated with solenoid #1 is set to North and the pole associated with solenoid #3 is set to South. The remaining solenoids are not energized. In this figure, the magnet in the middle (actuator moving piece) is locked in place.

Table 5.1: Five-step right movement control sequence.

Moving Steps	Energized solenoids	Resulting Polarities
Step #0	Solenoid #1 & #3	N (#1) & S (#3)
Step #1	Solenoid #2 & #4	N (#2) & S (#4)
Step #2	Solenoid #3 & #5	N (#3) & S (#5)
Step #3	Solenoid #4 & #6	N (#4) & S (#6)
Step #4	Solenoid #5 & #7	N (#5) & S (#7)
Step #5	Solenoid #6 & #8	N (#6) & S (#8)

Let's consider a control sequence that moves the actuator from the most left to the most right with a five-step movement. Because it is a linear step motor, we will move the actuator step-by-step. Table 5.1 presents the control sequence from Steps #1 to #5, where the solenoids are grouped as pairs. For example, for Step #0, solenoids #1 and #3 are paired together. Here, the positive is applied to solenoid #1, leading to a North pole for solenoid #1, while the negative is applied to solenoid #3, leading to a South pole for solenoid #3. In this case the rest of the solenoids are not energized.

Now let's consider Step #2 control with solenoids #3 and #5 energized and the rest of the solenoids not energized. In this case, the positive is applied to solenoid #3, leading to a North pole for solenoid #3, while the negative is applied to solenoid #5, leading to a South pole for solenoid #5. Note that for the controls of Steps #0 and #2, the voltage applied to solenoid #3 has an opposite sign; solenoid #3 is negative for Step #0 and positive for Step #2. As a result, an H-bridge drive circuit is needed (see Section 5.3).

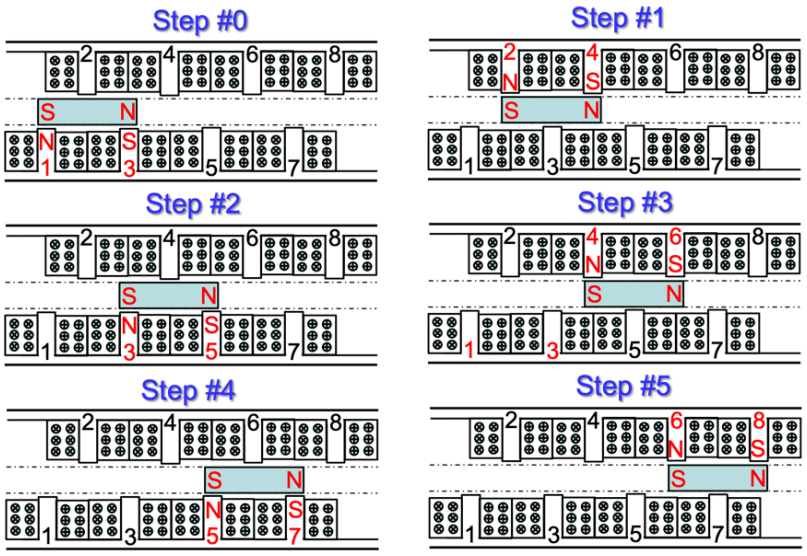


Figure 5.6: Five-step right movement process of a linear step motor.

Figure 5.6 shows the process of moving the linear actuator from its most-left position in Step #0 to its most-right position in Step #5. Note that in Step #0, the linear step motor is in the initial locked state; solenoids #1 and #3 activated and the resulting polarities of the solenoids are shown on the top-left plot of Figure 5.6 (North and South for solenoids #1 and #3, respectively). In Step #1, solenoids #2 and #4 are activated with the rest of the solenoids not energized. This results in North and South polarities of solenoids #2 and #4, respectively, moving the center magnet to right with its locked position shown on the top-right plot of Figure 5.6.

Next, in Step #2, solenoids #3 and #5 are activated with the rest of the solenoids not energized. This results in North and South polarities of solenoids #3 and #5, respectively, moving the center magnet to right with its locked position shown on the middle-left plot of Figure 5.6. In turn, in Step #3, solenoids #4 and #6 are activated with the rest of the solenoids not energized. This results in North and South polarities of solenoids #4 and #6, respectively, moving the center magnet to right with its locked position shown on the middle-right plot of Figure 5.6.

Next, in Step #4, solenoids #5 and #7 are activated with the rest of the solenoids not energized. This results in North and South polarities of solenoids #5 and #7, respectively, moving the center magnet to right with its locked position shown on the bottom-left plot of Figure 5.6. Last, solenoids #6 and #8 are activated with the rest of the solenoids not energized. This results in North and South polarities of solenoids #6 and #8, respectively, moving the center magnet to right with its locked position shown on the bottom-right plot of Figure 5.6. This completes movement from left to right.

This description of the steps summarizes the operational principle of a linear step motor (actuator). The continuous linear motor (actuator) often has the same hardware as the linear step motor, but the control of these solenoids needs to be modified to produce a continuous linear force.

Now we are ready to discuss a class of discrete linear (solenoid) actuators that can be used alone as valve actuators or be used as the base solenoid actuator for an actuation system. For example, certain engine hydraulic intake/exhaust valve systems use two solenoids to control the hydraulic valve system [5], [17]. The discrete solenoid actuator normally has two states (displacement positions). One is the solenoid not energized state associated with the resting state of the actuator. The other is the solenoid energized state associated with the activated state of the actuator. A popular application of the discrete solenoid actuator is the internal combustion engine fuel injector, where the discrete solenoid actuator is used to activate the fuel injector when the solenoid is activated [18].

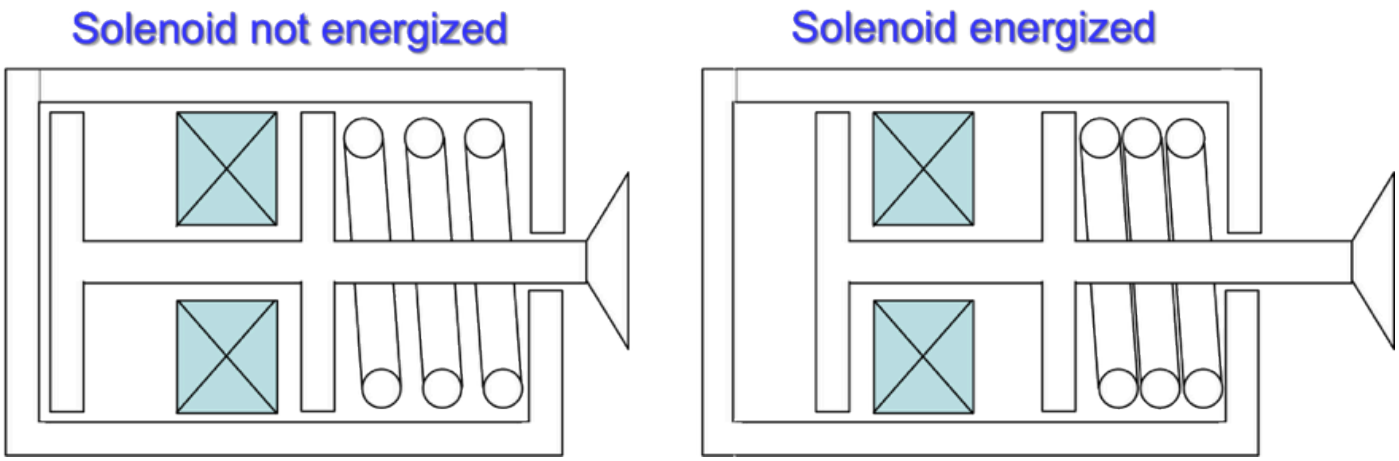


Figure 5.7: Discrete solenoid linear actuator.

Figure 5.7 shows an example of a discrete linear solenoid actuator. Here, the left plot shows the resting state of the discrete solenoid actuator, and the right plot presents the activated state. Note that the rectangular blocks with cross represent the solenoid windings. On the left plot, the spring on the right of the solenoid winding is extended, leading to a retracted (closed) valve position, whereas on the right plot, the spring on the right of the solenoid winding is compressed due to the magnetic force produced by the activated solenoid, resulting in an extended (open) valve position.

Section Summary

This section reviews various mechatronic actuation systems including mechanical, magnetic, and hydraulic systems, where the step motor, linear step motor, and discrete solenoid linear actuator are discussed in detail. This section also provides examples of mechatronic actuation systems, along with their typical practical applications, for rotational and linear actuators. Note that actuation systems are key components of mechatronic systems. Knowing how to use these actuators properly is very important for achieving desired system performance in terms of response time and energy consumption.

Section Keywords

Practice Problems

1. Which output shaft are step motors mainly used to regulate? Select the correct answer.
   a. torque.
   b. speed.
   c. angular displacement.
2. For a bipolar step motor with 6 pairs of permanent magnets, calculate the step angle SA for a single control pulse using the formula shown in equation 5.1.2.
3. Consider the linear actuator shown in Figure 5.4 and assume that the screw has one start with a screw pitch p_s of one-tenth inch (or 2.54 mm).
4. Find the gain that converts the motor rotational angle to the linear displacement.
5. Find the linear displacement if the motor rotates 90 degrees.
6. For the following movement sequence of a linear step motor from right to left, starting at step #0, find the control sequence in the form of pulse trains shown in Table 5.1.

This page titled 5.1: Mechatronic Actuator Background is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

5.2: Mechatronic Actuator Background

Learning Objectives

After completing this section, you will be able to:

1. Introduce the DC-motor operational principle.
2. Describe how the DC-motor torque is generated based upon the Lorentz principle.
3. Define the stall (maximum) torque and no-load (maximum) speed of a DC motor.

Note that there are mainly two types of motors: AC (alternating current) and DC (direct current) motors. In general, AC motors have high-power level and density; they are good for variable speed applications. DC motors have low-power, high torque, and variable speed; they are good for mechatronic applications.

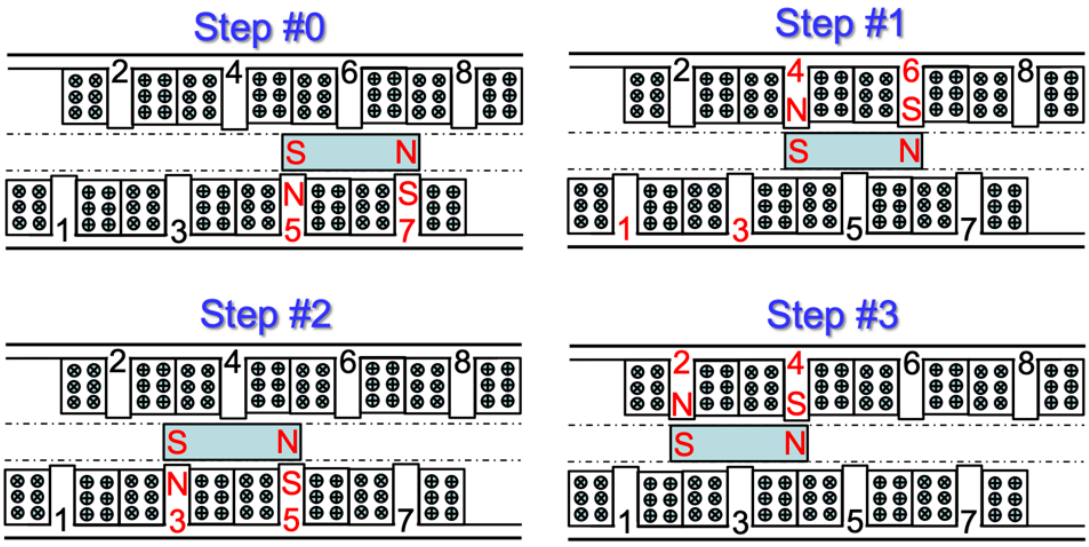


Figure 5.8: Mechatronic system signal conversion process.

In this section, we will address the DC-motor operational principle and its characteristics, especially the relationship between motor output torque and the resulting speed for a given armature voltage. Note that an electric motor converts electric energy to mechanical energy. Electric motors operate through electromagnetism, e.g. the interaction between the magnetic field and electric current. During the operational process, heat is generated because energy conversion efficiency is not 100%; see Figure 5.8 for an illustration. One of the heat sources is the armature winding resistance R_a. The armature current i_a flow into the armature winding generates the heat of $e_{heat} = R_a i_a^2$. In this section, we will focus on the permanent magnetic DC motors.

In Section 2.4, mechatronic system modeling, we described a DC motor by four differential equations associated with equations ???, ???, ???, and ???. These are listed in equation 5.2.1 with the same variables defined.

$$\begin{cases} e_a(t) = R_a i_a(t) + L_a \dfrac{di_a(t)}{dt} + e_m(t) & \tau(t) = K_\tau i_a(t) \\ e_m(t) = K_m \dot{\theta}(t) = K_m \omega(t) & J\ddot{\theta}(t) = \tau(t) - b_m \dot{\theta}(t) - \tau_l(t) \end{cases} \tag{5.2.1}$$

Recall that both DC-motor torque and back EMF coefficients, K_τ and K_m, were given without providing their derivations.

Theorem 5.1 [19], [20]

In a magnetic field with flux density B, the Lorentz magnetic force law states that the magnetic force F applied to a cable with length L and supply current i_a can be calculated by integrating over the coil length, resulting in the following equation.

$$F = i_a \cdot L \cdot B \tag{5.2.2}$$

The direction of generated force can be determined by Fleming's left-hand rule, shown in Figure 5.9. The associated variables in equation 5.2.2 are marked in Figure 5.9 and Figure 5.10.

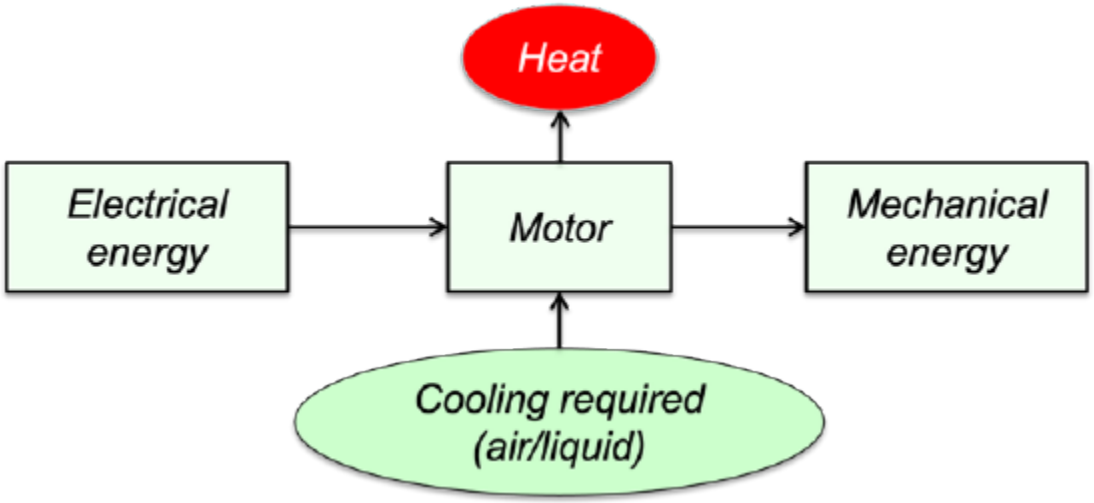


Figure 5.9: Magnetic Lorentz force law.

Figure 5.9 shows two wires moving in a magnetic field B formed by two magnets, indicated by the red (north) and green (south) bars. The dot inside the circle indicates the current i_a flowing out of the surface; the circle represents the wire. The cross inside the circle indicates the current i_a flowing into the surface, where the direction of the generated force can be determined by the left-hand rule shown in Figure 5.9.

Now, let's consider a single motor winding in a magnetic field B as shown in the left plot of Figure 5.10. Here, the winding radius is R, winding depth is L, the current flow into and out of the winging is i_a, and the generated force is represented by F. Using the left-hand rule shown in Figure 5.9, we can apply equation 5.2.2 to calculate the resulting force due to the current flow i_a inside the wire with length L, given magnetic field B with the force direction indicated on the left plot of Figure 5.10, and the associated torque generated by the single winding motor rotor.

$$\tau = 2 \cdot i_a \cdot L \cdot B \cdot R \tag{5.2.3}$$

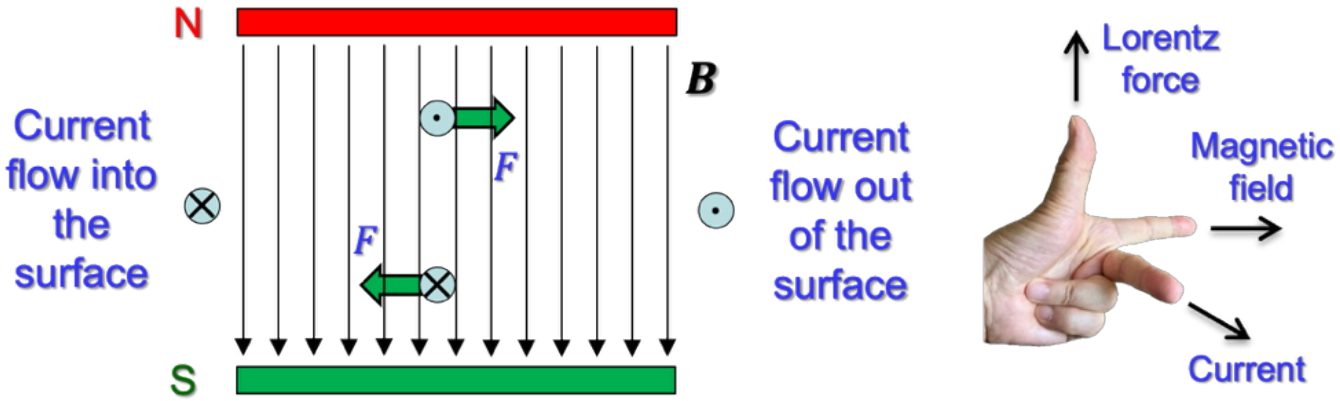


Figure 5.10: Motor winding in a magnetic field and the force generated.

For a motor rotor inside uniform magnetic field B with the number of winding equal to N, we can calculate the generated torque by the following equation.

$$\tau = 2 \cdot N \cdot i_a \cdot L \cdot B \cdot R = K_\tau i_a \tag{5.2.4}$$

where $K_\tau = 2NLBR$. However, in practice, for a given motor design, the magnetic field may not be uniform, and the individual winding radius is not the same. Therefore, coefficient K_τ often needs to be determined experimentally. The equations derived can be used as an initial calibration to determine coefficient K_τ.

Now, we are ready to address the process of deriving the back EMF coefficient K_m due to the counter EMF (electromotive force) voltage. We will first introduce the Lenz law [21] below.

Theorem 5.2 [21]

The Lenz law states that the induced electromotive force with different polarities results in a current whose magnetic field opposes the change in magnetic flux through the loop. The reason is to ensure that the original flux is maintained through the loop when current flows in it.

The process of generating counter EMF voltage on the motor winding is shown in Figure 5.11. This process starts with applying an armature voltage to the DC-motor rotor winding that generates an armature current. This generates torque and rotor motion. Because the rotor winding wire cuts through the magnetic field, an opposing voltage to the applied armature voltage, called back EMF voltage, is generated. This voltage is proportional to the DC-motor rotor speed because the back EMF voltage is proportional to the wire moving speed relative to the magnetic field.

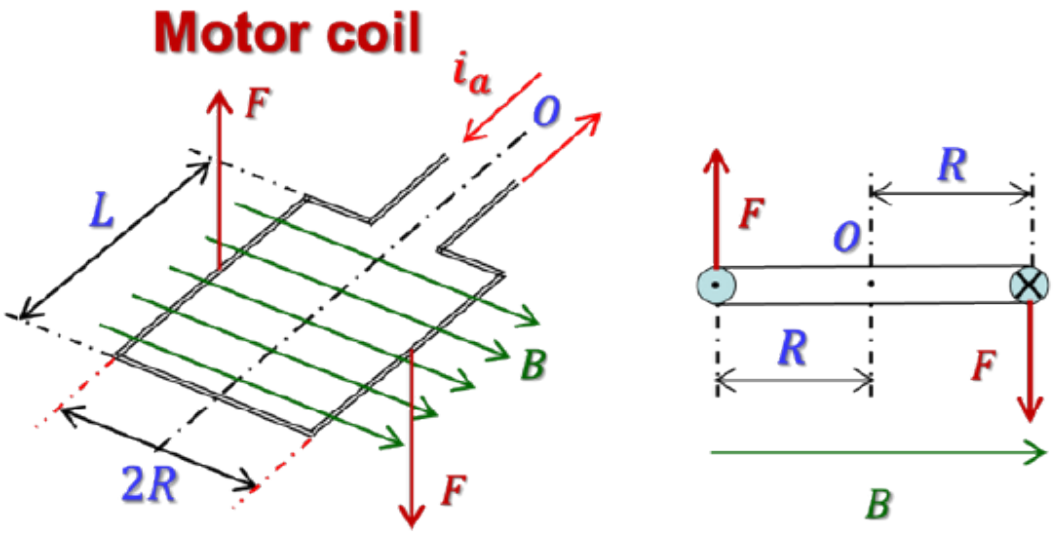


Figure 5.11: Generation of the back EMF (electromotive force) or counter EMF voltage.

Based on Lenz's law [21], after certain simplifications, the voltage generated by back EMF effect can be calculated by

$$V_m = 2 \cdot N \cdot L \cdot B \cdot R \cdot \omega = K_m \omega \tag{5.2.5}$$

where parameters N, L, B, and R were defined above; $K_m = 2NLBR$ is the DC-motor back EMF voltage coefficient; and ω is motor rotor angular speed. Because EMF increases as the motor speed increases, the motor output torque reduces, making it easy to control motor speed.

Example 5.3

As discussed above, using international units, what is the unit for the DC-motor torque coefficient K_τ?

The torque coefficient converts the armature current i_a in A (ampere) to the motor-shaft output torque τ in Nm (Newton-meter). As a result, the unit for K_τ is shown below.

$$K_\tau : Nm/A \tag{5.2.6}$$

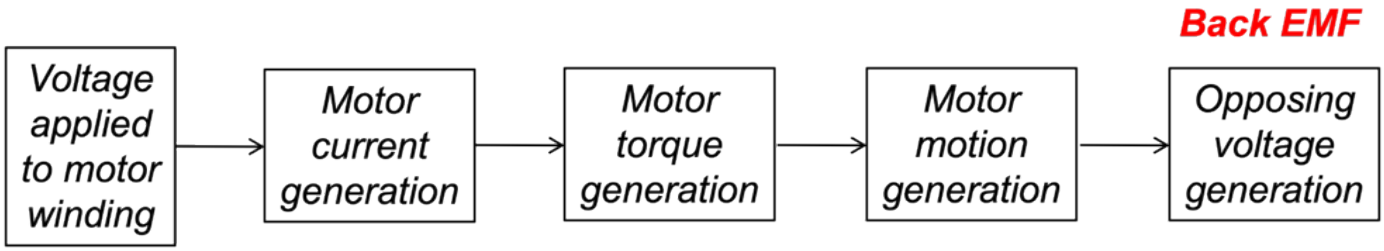


Figure 5.12: DC-motor commutator, brush, and rotor windings.

Next, we will provide a rough idea of how the DC motor works using a single pole (winding pair) rotor as an example. For a permanent magnet DC motor, its rotor assembly consists of a pair of fixed brushes (represented by black rectangular blocks) that supply DC power to the rotor windings, as well as the rotating commutator (represented by brown half circles) in the middle. In the right plot of Figure 5.12, the motor state consists of two permanent magnets denoted by *N* and *S* on the outskirt of the motor. In order to make the DC-motor rotor rotate, the power (armature voltage) applied to the rotor windings needs to be switched through the commutator in the middle.

Figure 5.13 illustrates this process for a two-pole rotor (pair of windings). Here, the battery is connected to the two fixed motor brushes. The two-pole commutator attached to the motor rotor rotates along with the rotor and windings. When the rotor rotates in order to keep the output torque remaining in the same direction, the current directions of both windings are changed by the commutator.

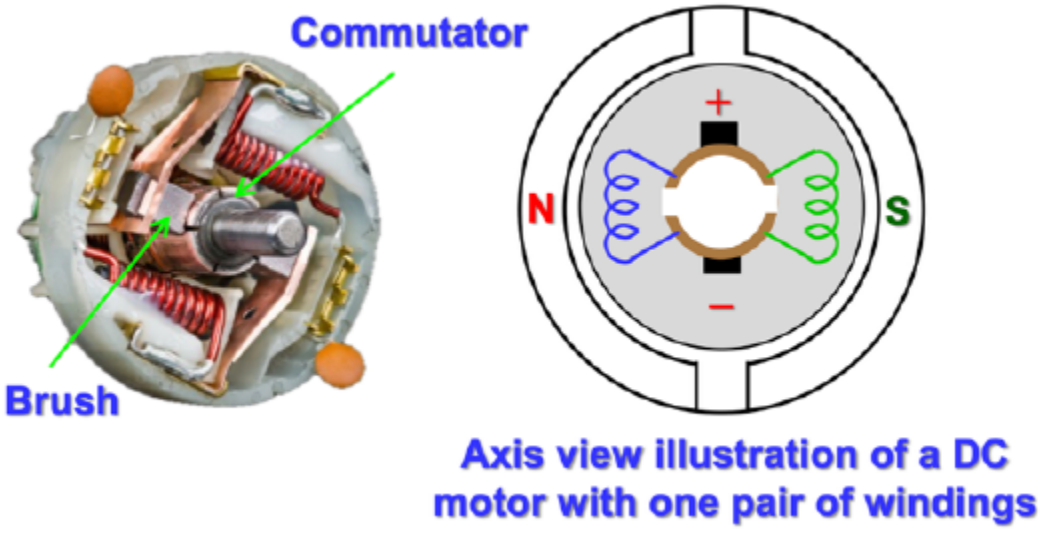


Figure 5.13: Rotation process of a single winding-pair DC motor.

Figure 5.13 uses five steps for a 180° anticlockwise rotation with an initial angle of 270°. Starting with Step #1, with rotational angle equal to 270° (see top-left plot), the power is supplied to the two windings through the commutator. It generates two forces F in the opposite directions due to the stator magnetic field. This leads to a generated torque of $T = 2RF$. This torque rotates the motor rotor in the anticlockwise direction, where R is the orthogonal distance from generated force F to the center of the axis. Note that you can find the generated torque at the #1 location on the top-left plot of Figure 5.14 with known armature current i_a and DC-motor torque coefficient K_τ.

In Step #2 with rotational angle of 225° (see bottom-left plot of Figure 5.13), we see the power is still supplied to the two windings in the same direction as Step #1. It generates two forces F in the opposite directions due to the stator magnetic field. This leads to a generated torque of $T = 2RF\,|\sin 225^\circ|$ (see "#2" mark on the bottom-left plot of Figure 5.14), which rotates the motor rotor

in the anticlockwise direction.

In Step #3 with rotational angle of 180° (see top-middle plot of Figure 5.13), the power is not supplied to the two windings because they are disconnected and generate no force. This leads to a generated torque of $T = 0$ (see "#3" mark on the top-middle plot of Figure 5.14). Even though the generated torque is zero, the motor rotor continues rotating due to its rotational moment.

In Step #4 with rotational angle of 135° (see bottom-right plot of Figure 5.13), the power is now supplied to the two windings in the opposite direction to Step #1. It generates two forces F in opposite directions due to the stator magnetic field. This leads to a generated torque of $T = 2RF\,|\sin 135^\circ|$ (see "#4" mark on the bottom-right plot of Figure 5.14), which rotates the motor rotor in the anticlockwise direction.

Finally in Step #5 with rotational angle of 90° (see top-right plot of Figure 5.13), the power is now supplied to the two windings in the opposite direction to Step #1. It generates two forces F in the opposite directions due to the stator magnetic field. This leads to a generated torque of $T = 2RF$ (see "#5" mark on the top-right plot of Figure 5.14), which rotates the motor rotor in the anticlockwise direction.

In Figure 5.14, we can see from the top-right plot that the two-pole DC motor in the top-left plot has a very large torque ripple. The motor output torque is in a form of an absolute sinusoid function varying from zero to peak torque output of $2RF$ (100% ripple).

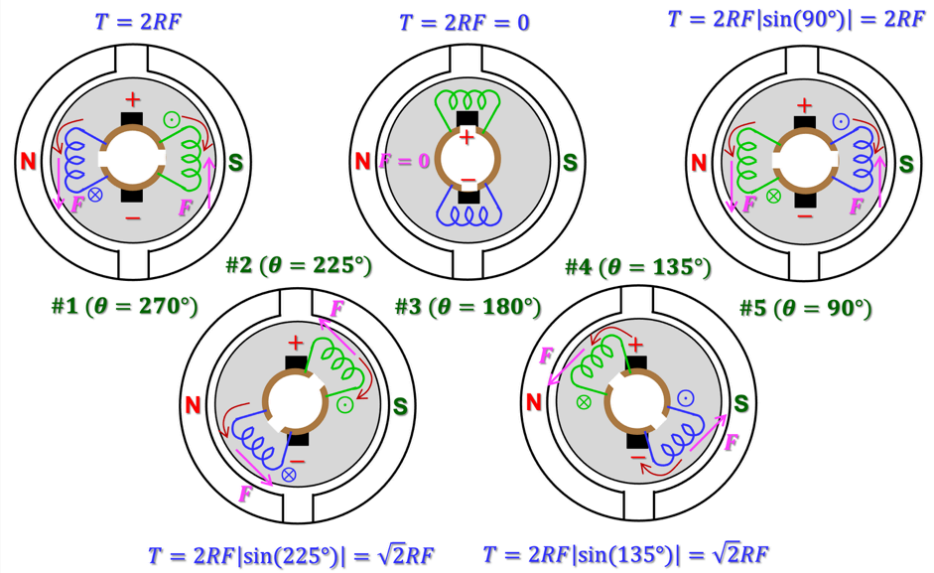


Figure 5.14: Output torque profiles of two- and four-pole permanent magnet DC motors.

To reduce the output torque ripple, we need to increase the number of motor poles (winding pairs). Again, in Figure 5.14, we can see the four-pole DC motor in the bottom-left plot with its torque output profile (red trace) in the bottom-right plot. The torque ripple reduces significantly. The blue-dashed and green dash-dot curves on the bottom-right represent the torque generated by each pair of rotor windings. Because the two torque curves complement each other, the torque ripple can be reduced. We will use Example 5.4 to demonstrate how to calculate the level of torque ripple.

Example 5.4

Consider the four-pole DC motor with two pairs of the winding shown in the bottom-left of Figure 5.14. The green and blue torque curves can be expressed using the following equations, assuming $R = F = 1$.

$$\begin{aligned} T_{blue}\,(\theta) &= |2\sin\theta| \\ T_{green}\,(\theta) &= \left|2\sin\left(\theta + \frac{\pi}{2}\right)\right| \end{aligned} \tag{5.2.7}$$

The first intersection point of the two curves for $\theta > 0$ is that $\theta = \pi/4$. In this case, we have the following.

$$T_{blue}\left(\frac{\pi}{4}\right) = 2\sin\frac{\pi}{4} = T_{green}\left(\frac{\pi}{4}\right) = 2\sin\left(\frac{\pi}{4}+\frac{\pi}{2}\right) = \sqrt{2} \tag{5.2.8}$$

Therefore, $T_{max} = 2\sqrt{2}$ and $T_{min} = 2$. As a result, the percentage ripple can be calculated below.

$$R_{\%} = \frac{T_{max} - T_{min}}{T_{max}} \times 100 = \frac{2\sqrt{2}-2}{2\sqrt{2}} \times 100 = 29.3\% \tag{5.2.9}$$

Increasing from one pair of poles to two reduces the torque ripple from 100% down to 29.3%, a reduction of 70.7%.

Now, we will introduce characteristics of the DC motor such as average peak torque and speed for a given armature voltage level. Using equation 5.2.1, and assuming a small inductor value by setting $L_a = 0$, we have the following equation.

$$e_a = \frac{R_a}{K_\tau}\tau + K_m\omega \tag{5.2.10}$$

This equation provides the relationship of the DC-motor output torque τ and resulting speed ω for a given armature voltage e_a. We can see from equation 5.2.10 the relationship of motor torque and speed for a given armature voltage is an affine function (a mathematical function of linear transformation with a nonzero offset). We will define the DC-motor stall torque and no-load speed as below.

Definition 5.2

The stall torque τ_{stall} of a DC motor for a given armature voltage e_a is defined as the motor output torque when the motor speed is zero, which is equal to $\frac{K_\tau}{R_a}e_a$. The no-load speed $\omega_{no-load}$ for a given armature voltage is defined as the motor output speed with zero load torque, which is equal to $\frac{e_a}{K_m}$.

Remark 5.2

Based on equation 5.2.10 and Definition 5.2, we can reach the following conclusions. With fixed load, motor shaft speed ω is proportional to control voltage. With fixed control voltage, shaft speed is negatively proportional to torque load. Shaft output torque is proportional to motor current, independent of motor voltage. Motor efficiency is mainly impacted by its electric resistance and mechanical friction.

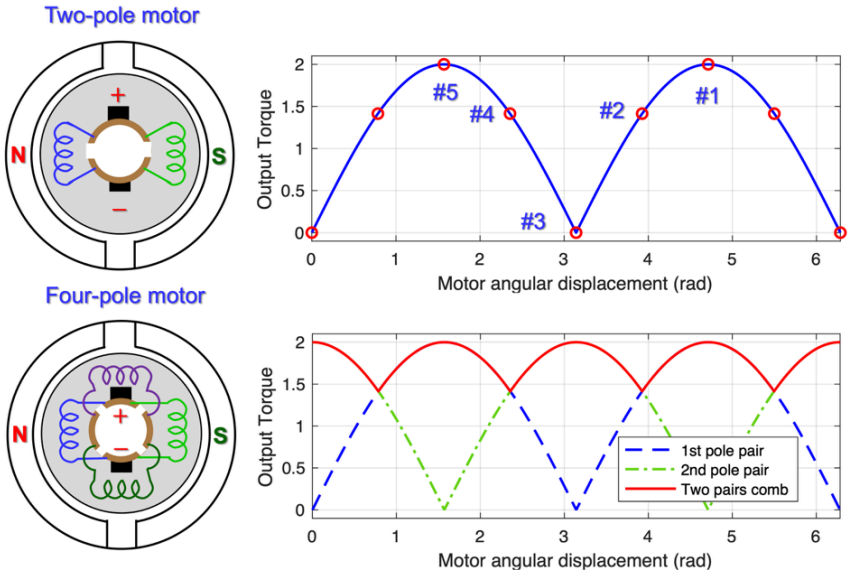


Figure 5.15: Output torque and speed profiles.

Figure 5.15 shows the relationship of DC-motor output torque τ and speed ω, where the relationship varies as the armature voltage e_a changes. As the armature voltage increases, both stall torque τ_{stall} and no-load speed $\omega_{no-load}$ (marked in purple on Figure 5.15) increase. For a given armature voltage, the motor torque and speed relationship is affine, represented by a straight line.

Example 5.5

For the DC motor with $R_a = 0.1\ \Omega$, $K_\tau = 0.5\ Nm/A$, $K_m = 0.5\ V/(rad/s)$, and $e_a = 5\ V$, find the stall torque in Newton-meter (Nm) and no-load speed in rad/s. The stall torque in Nm can be calculated using the following equation.

$$T_{stall} = \frac{K_\tau}{R_a} e_a = \frac{0.5}{0.1} \times 5 = 25\ Nm \tag{5.2.11}$$

The no-load speed can be calculated using the equation below.

$$\omega_{no-load} = \frac{e_a}{K_m} = \frac{5}{0.5} = 10\ rad/s \tag{5.2.12}$$

Therefore, the calculated stall torque is $25\ Nm$ and the calculated no-load speed 10 rad/s.

In practice, a gearbox is often used to change the range of motor output torque and speed. Assuming that the gearbox does not have power loss (normally there are a few percents), we can state that the power input to and output of the gearbox remain the same. That is, $P = \tau \cdot \omega = \tau_o \cdot \omega_o$, where τ and ω are input axis torque and speed and τ_o and ω_o are output axle torque and speed. Due to the assumption of no transmission loss, we have the following relationship.

$$\eta = \frac{N_o}{N_i} = \frac{\omega}{\omega_o} = \frac{\tau_o}{\tau} \tag{5.2.13}$$

where η is gear ratio, N_o is the output shaft number of teeth, and N_i is the input shaft number of teeth. As a result, the gearbox output torque and speed are in an inverted relationship.

Figure 5.16 demonstrates the effect of a gearbox on the output torque and speed. In general, if $\eta > 1$, i.e., $N_o > N_i$, at the gearbox output shaft, the stall torque increases, and no-load speed reduces. Thus, if one wants to increase the torque output, one needs to select a gearbox with $\eta > 1$, and vice versa.

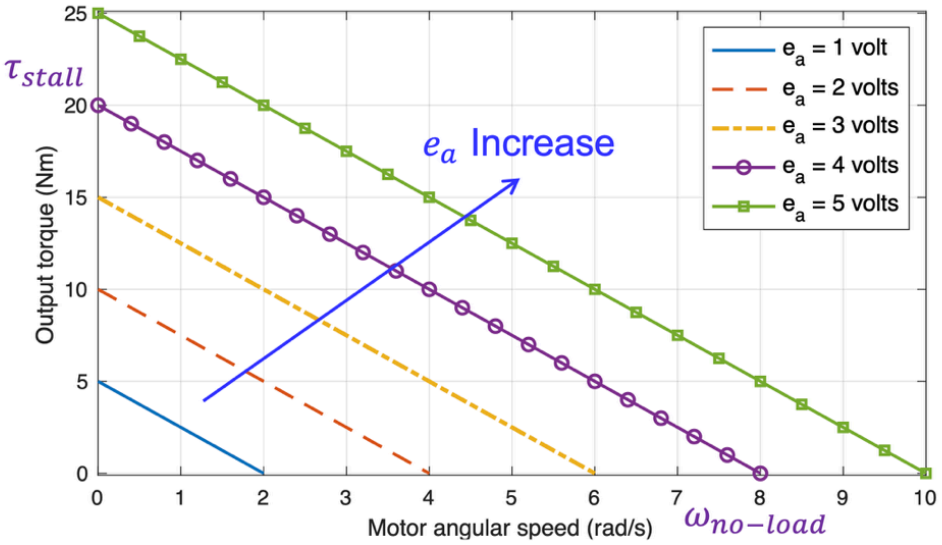


Figure 5.16: Gearbox effect to output torque and speed profiles.

Example 5.6

For a DC motor with its maximum stall torque of $25\ Nm$ and no-load speed of $10\ rad/s$, find a gearbox with its gear ratio of η so that the stall torque of gearbox output shaft is doubled with respect to the motor shaft. Also calculate the associated no-load speed at the gearbox output shaft. Using equation 5.2.13, we have

$$\eta = \frac{N_o}{N_i} = \frac{\omega}{\omega_o} = \frac{\tau_o}{\tau} = \frac{50}{25} = 2 \tag{5.2.14}$$

The no-load speed of the gearbox output shaft can be calculated using the equation below.

$$\frac{\omega}{\omega_o} = 2 \rightarrow\ \omega_o = \frac{\omega}{2} = \frac{10}{2} = 5\ rad/s \tag{5.2.15}$$

Therefore, the calculated gear ratio is $\eta = 2$ and the calculated correspondent no-load speed is $5\ rad/s$.

Below, we will discuss the transient characteristics of a DC motor. To be specific, we will investigate the acceleration process of a DC motor from zero speed to its steady-state speed with and without axle friction. Consider a DC motor described by equation 5.2.1 with the motor parameters shown in equation 5.2.16.

$$R_a = 0.1\ \Omega, L_a = 0, K_\tau = 0.5\ Nm/AK_m = 0.5\ V/(rad/s), J = 0.4\ Kgm^2, b_m = 1\ Nm/(rad/s) \tag{5.2.16}$$

First, let's study the case with $b_m = 0$, that is, there is no axle friction torque. For the given parameters in equation 5.2.16, based upon dynamic equations in equation 5.2.1, we can derive the system transfer function from armature voltage $E_a(s)$ to motor speed Ω(s) with $b_m = 0$ as below, drawing on equation ???.

$$\frac{\Omega(s)}{E_a(s)} = \frac{K}{Ts+1} = \frac{2}{0.16s+1} \tag{5.2.17}$$

With the derived transfer function in equation 5.2.17, we can simulate the step armature voltage of $E_a(s) = 5/s$ (five times of unit step, 5 volts) in MATLAB using the command step. Figure 5.17 presents the simulation results. The blue line in the top plot shows a straight line between the stall torque (25 Nm) and the no-load speed (10 rad/s) that can be calculated by equations 5.2.11 and 5.2.12. The blue curve in the bottom plot shows the time response of the DC-motor speed described by 5.2.17.

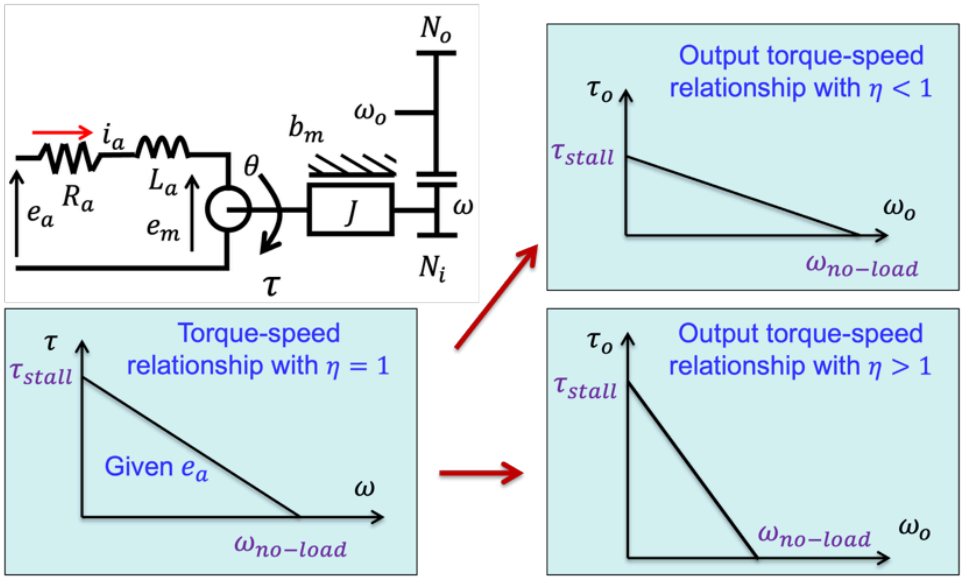


Figure 5.17: Transient response of a DC motor without axle friction.

Because the system time constant $T = 0.16$, the 2% settling time (see Chapter 3) is $t_s\,(2\%) = 4 \times 0.16 = 0.64$ second. That is, the speed response will reach 98% of the steady-state speed (in this case, no-load speed) in 0.64 second. The red dots in both the top and bottom plots show the migration process of the speed from zero to 10 rad/s corresponding to those in the bottom chart.

Because there is no axle friction ($b_m = 0$), the no-load speed of 10 rad/s is achieved at the steady state.

For the parameters given by equation 5.2.16 with $b_m = 1$, and based upon the dynamic equations in 5.2.1, we can derive the system transfer function from armature voltage $E_a(s)$ to motor speed Ω(s), as below.

$$\frac{\Omega(s)}{E_a(s)} = \frac{K}{Ts+1} = \frac{1.4286}{0.1143s+1} \tag{5.2.18}$$

We can see that both DC gain $K = 1.4286$ and time constant $T = 0.1143$ are reduced, comparing with the no friction case. This result indicates the time response will be faster than the case without friction and the achieved steady-state speed will be reduced.

With the derived transfer function in equation 5.2.18, we can simulate the step armature voltage of $E_a(s) = 5/s$ (five times of unit step) in MATLAB using the command Step. Figure 5.18 presents the simulation results. The blue line in the top plot of Figure 5.18 shows a straight line between the stall torque (25 Nm) and no-load speed (10 rad/s) that is the same as these for the no friction case. The red line is the friction torque as a linear function of motor speed ($b\omega$). The orange line shows the available torque for accelerating the rotor. Note that the available torque for acceleration is significantly reduced from the case without friction torque.

The solid blue curve in the bottom plot of Figure 5.18 shows the time response of the motor system described by 5.2.17 (no friction case); the red dash-dot line represents the time response of the motor system described by 5.2.18 (friction case). Because the system time constant is $T = 0.1143$ for the friction case, the 2% settling time (see chapter 3) is $t_s(2\%) = 4 \times 0.1143 = 0.4572$ second. That is, the speed response will reach 98% of the steady-state speed (in this case, $\omega = 5 \times K = 7.143$ rad/s) in 0.4572 second, which is faster than the no friction case of 0.64 second.

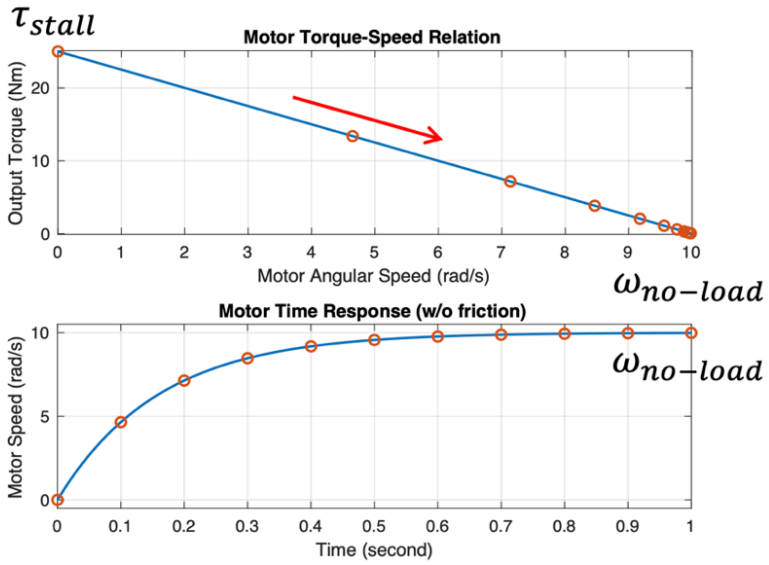


Figure 5.18: Transient response of a DC motor with axis friction.

The magenta dots in the top and bottom plots of Figure 5.18 show the migration process of the speed from zero to its steady state value of 7.143 rad/s. Due to the existence of axle friction ($b_m = 1$), the no-load speed cannot be reached, and the steady-state speed becomes 7.143 rad/s (less than the no-load speed of 10 rad/s). Even though the case with friction torque converges to its steady state faster than the case without friction torque, the rotor speed without friction torque accelerates faster than the case with friction torque.

Example 5.7

For the DC motor with $R_a = 0.1\ \Omega$, $K_\tau = 0.5\ Nm/A$, $K_m = 0.5\ V/(rad/s)$, $J = 0.4\ Kgm^2$, and $b_m = 1\ Nm/(rad/s)$, derive the system transfer function from $E_a(s)$ to output speed $\Omega(s)$. Calculate the steady-state motor speed with $e_a = 5\ V$. From equation ???, we have the following.

$$K = \frac{K_\tau}{R_a b_m + K_\tau K_m} = \frac{0.5}{0.1 \times 1 + 0.5^2} = 1.4286 \tag{5.2.19}$$

The time constant can be calculated by the following equation.

$$T = \frac{JR_a}{R_a b_m + K_\tau K_m} = \frac{0.4 \times 0.1}{0.1 \times 1 + 0.5^2} = 0.1143 \ (sec) \tag{5.2.20}$$

Therefore, the calculated $K = 1.4286 \ rad/s$ and the calculated time constant $T = 0.1143$ second. The achieved steady-state speed in this case can be calculated by the following equation.

$$\omega_{ss} = K \times e_a = 1.4285 \times 5 = 7.143 \ rad/s \tag{5.2.21}$$

In short, basic characteristics of DC motors are a) with a fixed load, the motor steady-state shaft speed is proportional to armature voltage; b) with fixed armature voltage, the motor steady-state shaft speed is negatively proportional to torque load; c) the shaft torque is proportional to the motor armature current (independent of motor armature voltage); and d) motor efficiency is mainly impacted by its armature electric resistance and mechanical shaft friction (generating heat).

Section Summary

This section introduces DC-motor working principles. It addresses how to calculate the DC-motor torque gain (K_τ), which is a proportional gain of motor armature current. It also addresses how to calculate the back EMF (electromotive force) coefficient (K_m), which is a proportional gain of motor speed; see Chapter 2 and reference [1]. The concepts of maximum motor brake torque (stall torque) and maximum motor speed (no-load speed) as a function of input armature voltage are also discussed. This section also reviews the effect of the gearbox to the DC-motor characteristics in terms of geared output torque and speed. In general, assuming that there is no gearbox efficiency loss, the energy input to and output of the gearbox remains the same.

Section Keywords

Practice Problems

1. For the DC motor with $R_a = 0.1 \ \Omega$, $K_\tau = 0.5 \ Nm/A$, $K_m = 0.5 \ V/(rad/s)$, and $e_a = 10 \ V$, find the stall torque in Nm and no-load speed in rad/s.
2. For the DC motor with its stall torque of $20 \ Nm$ and the no-load speed of 8 rad/s, find a gearbox with its ratio of η so that the stall torque of the gearbox output shaft is cut by half with respect to the motor shaft. Also calculate the associated no-load speed at the gearbox output shaft.
3. For the DC motor with $R_a = 0.1 \ \Omega$, $K_\tau = 0.5 \ Nm/A$, $K_m = 0.5 \ V/(rad/s)$, $J = 0.4 \ Kgm^2$ and $b_m = 2 \ Nm/(rad/s)$, derive the system transfer function from $E_a(s)$ to output speed $\Omega(s)$. Calculate the steady state motor speed with $e_a = 5 \ V$.

This page titled 5.2: Mechatronic Actuator Background is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

5.3: DC-Motor Drive Circuits

Learning Objectives

After completing this section, you will be able to:

1. Explain both low- and high-side DC-motor drive circuits.
2. Describe a PWM (pulse-width modulation) high-side drive circuit and its working principle.
3. Explain the working principle of an H-bridge circuit driven by the PWM signals.

This section discusses how to drive the DC motor efficiently using a PWM (pulse-width-modulated) signal in either a single or double direction. Single direction uses a high-side or low-side drive whereas dual-direction uses an H-bridge drive. Figure 5.19 illustrates the single-direction (or one-direction) drive. The figure's left plot shows a low-side drive and the right plot shows a high-side drive.

The current (or voltage) regulator is used to regulate the DC-motor armature current i_a to achieve the desired motor torque; the motor torque output is proportional to its armature current. In Section 4.6, we briefly introduced a sample low-side DC-motor drive circuit (see Figure 4.57), where an operational amplifier and a transistor are used to regulate the motor armature current. The principle of the low-side drive (or high-side) is to regulate the voltage drop v_{reg} on the voltage or current regulator (see Figure 5.19) so that the DC-motor armature voltage $v_a = V_{bat} - v_{reg}$ results in the desired armature current i_a (or DC-motor output torque τ).

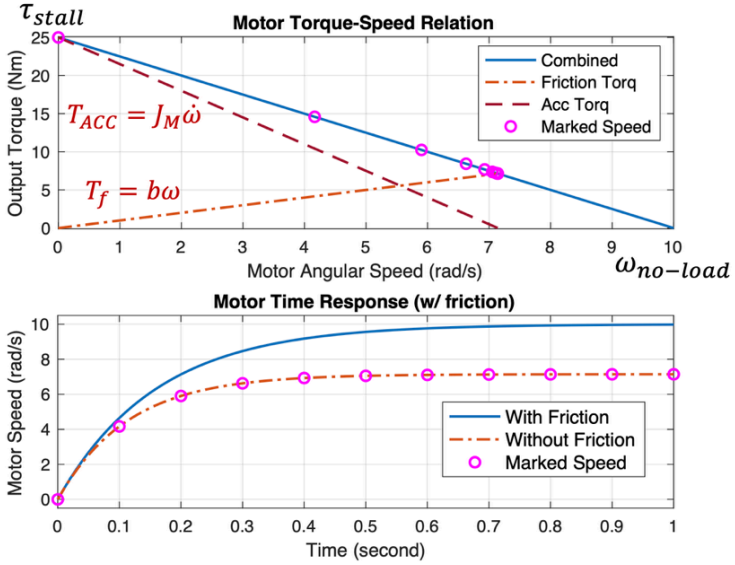


Figure 5.19: Low- and high-side single-direction drives.

The type of transistor used in the current regulation circuit is a primary factor in selecting a low-side or high-side drive. For example, an NPN (negative-positive-negative) transistor is often used to construct the low-side drive circuit (see the left plot of Figure 4.57). A PNP (positive-negative-positive) transistor is for the high-side drive circuit.

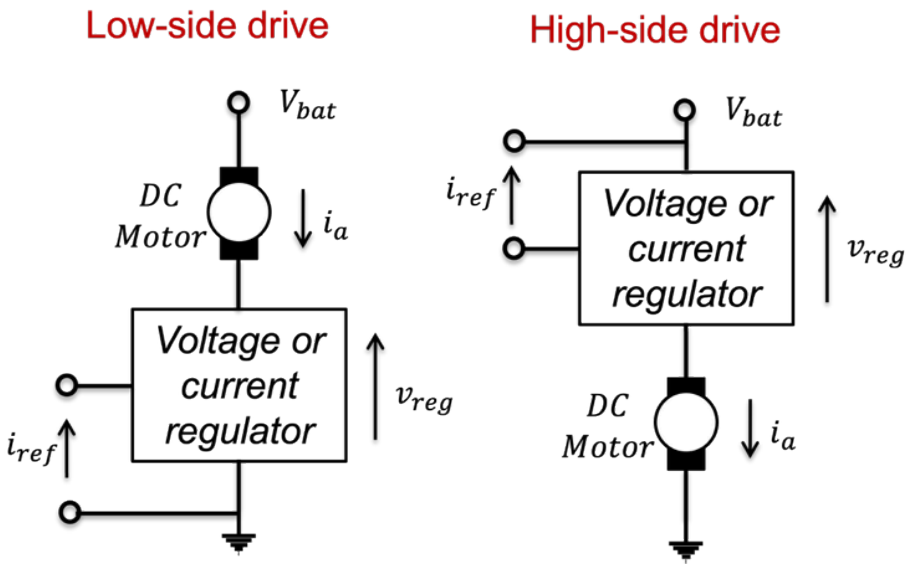


Figure 5.20: A sample high-side drive circuit.

A sample high-side drive circuit is shown on Figure 5.20. One operational amplifier (on the right side, marked “Feedback isolation circuit”) is used to generate the feedback voltage drop on resistor R_s. Another operational amplifier is used for feedback control of the motor armature current. The control circuit is intended to have the following relationship

$$v_{ref} = V_{bat} - v_e \approx R_s i_a \;\rightarrow\; i_a \approx \frac{v_{ref}}{R_s} \tag{5.3.1}$$

due to the following relationship from the feedback isolation circuit.

$$\frac{v_e - 0.5V_{bat}}{R_f} = \frac{0.5V_{bat} - v_{ref}}{R_f} \tag{5.3.2}$$

In this case, a desired armature current can be achieved by setting v_{ref} as follows.

$$v_{ref} = i_{a-desired} R_s \tag{5.3.3}$$

where $i_{a-desired}$ is the desired armature current. We are now ready to discuss the PWM drive for low-side and high-side PWM drive circuits. The low-side PWM drive circuit was previously presented in Figure 4.57; the associated benefit of a PWM drive were addressed in Example 4.20 in Section 4.6. Here, we will discuss only the high-side drive PWM circuit. To utilize the PWM signal, the transistor is often replaced by MOSFET (metal oxide semiconductor field-effect transistor) or IGBT (insulated-gate bipolar transistor) because of their fast-switching capability.

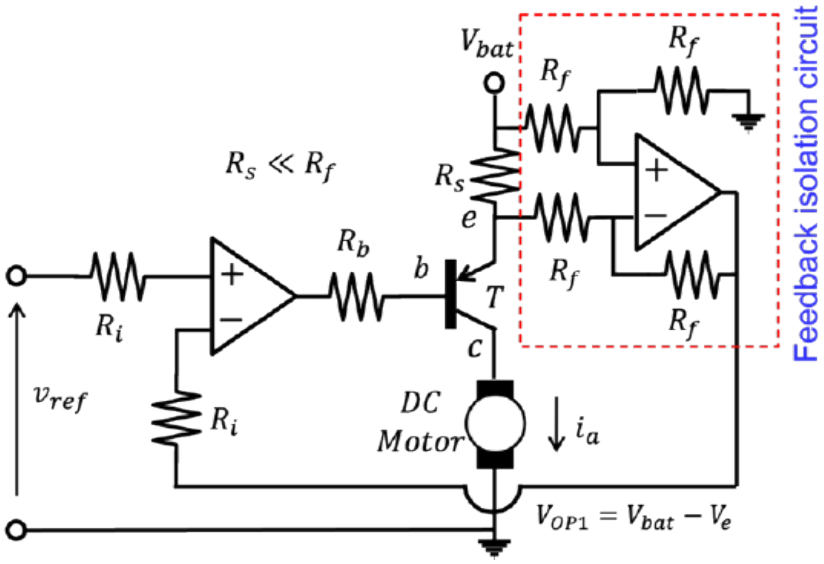


Figure 5.21: A sample high-side PWM drive circuit.

Figure 5.21 presents a sample high-side drive circuit for DC motors. Here, transistor T_i is used to inverse the PWM signal with isolation. The diode D_i is used to improve the circuit robustness to ground noise; as a result, the circuit requires about 1.4 volts to turn on transistor T_i (instead of 0.7 volt in case without D_i). After the transistor T_i is turned on, the collector voltage is reduced to about 1.0 volt; as a result, the drive transistor T_d is turned on and supplies power to the DC motor. When the input voltage v_{pwm} is low, transistor T_i is turned off and so is transistor T_d, leading to no power supplied to the DC motor. Note that the diode D_m is used to allow the motor current to flow when transistor T_m is turned off. The reason is that the motor current cannot be changed instantly due to its armature winding inductance. Examples 5.8 and 5.9 focus on analog and PWM high-side drive circuits, respectively.

Example 5.8

For the analog high-side drive circuit presented on Figure 5.20, with the sensing resistor value of $0.2\ \Omega$, find the reference voltage v_{ref} to achieve the target DC-motor armature current of $i_a = 5\ A$. From equation 5.3.3 we have the following.

$$v_{ref} = i_{a-desired} R_s = 5 \times 0.2 = 1.0\ V \tag{5.3.4}$$

Therefore, the reference voltage v_{ref} is $1.0\ V$.

Example 5.9

Assume that reference v_{pwm} is a 0 to 5 volts PWM with its duty-cycle of 40%, the battery voltage is $V_{bat} = 12.2$ volts, DC-motor armature resistance R_a is 0.1 Ω, and the voltage drop of transistor T_m in Figure 5.21 is v_{e2c}= 0.2 when it is fully turned on. Find the resulting armature current in Figure 5.21. Note that with a duty-cycle of 40% for the input PWM reference signal, the transistor T_m will be turned on with a 40% duty-cycle. When the transistor T_m is turned on, the DC-motor armature current is shown below.

$$i_{a-on} = \frac{V_{bat} - v_{e2c}}{R_a} = \frac{12.2 - 0.2}{0.1} = 120\ A \tag{5.3.5}$$

With the 40% duty-cycle, the averaged motor armature current is shown below.

$$i_a = i_{a-on} \times 40\% = 120 \times 0.4 = 48\ A \tag{5.3.6}$$

Having discussed low-side and high-side DC-motor drive methods, let's now look at the H-bridge drive for DC motors. Compared with the H-bridge drive, the main disadvantage of low- and high-side drive methods is that the applied armature voltage direction cannot be altered. That is, it is impossible to change the armature current direction due to fixed armature voltage direction. As

a result, the torque generated by the DC motor will maintain in one direction. This makes the control torque of the mechatronic system in one direction, which limits the capability of the motor speed control (to be discussed in this section).

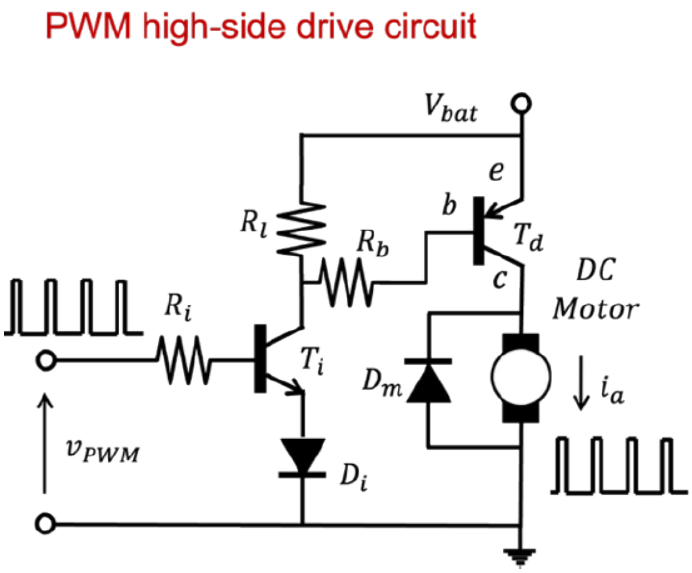


Figure 5.22: Architecture of a four-switch H-bridge drive circuit.

Figure 5.22 shows the architecture of an H-bridge drive system, where the diodes are used to protect switches right after the switches are turned off. The DC-motor armature winding has a certain inductance such that the armature current cannot be set to zero instantaneously. These diodes provide a path to allow the armature current flow so that the voltage on these switches will not increase to the level damaging the switches. When driving the DC motor in one direction, we need to activate a pair of the switches represented by a circle with contact. For example, to generate an armature current flow from left to right for the DC motor in Figure 5.22, the upper-left and lower-right switches need to be turned on. In this case, we can see that the H-bridge drive looks like the combined high-side and low-side drives discussed earlier. Note that the high-side (or low-side) drive has a switch on the high-side (or low-side).

Figure 5.23 shows how the four-switch H-bridge drive can be used to drive the DC motor in forward (clockwise) and backward (anticlockwise) directions for both rotational directions. Recall that the DC motor can be controlled by the applied armature voltage (or current) to the H-bridge circuit using pulse-width-modulation (PWM) signals. That allows the equivalent armature voltage (or current) to be regulated by PWM signal duty-cycle in the same way as that used for the high-side or low-side PWM drive.

To regulate the motor direction, a pair of the switch needs to be turned on simultaneously. To turn the DC motor forward in the clockwise direction (see the left drawing), we turn on the upper-left and lower-right switches (marked by red energized switches), resulting in the marked armature current in Figure 5.23. To turn the DC motor backward in the anticlockwise direction (see the right drawing), we turn on the upper-right and lower-left switches simultaneously (marked by red energized switches), resulting in the marked armature current in Figure 5.23. When the motor is activated, these four diodes are not activated because the current flows in the opposite direction to the diode conducting one.

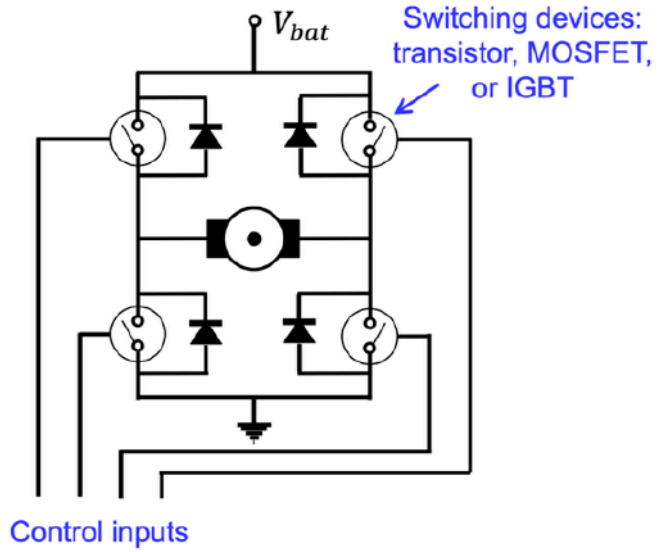


Figure 5.23: Driving the DC motor forward and backward using a four-switch H-bridge drive.

Another mode of operation is braking regeneration for the four-switch H-bridge. Figure 5.24 shows two operational modes for braking regeneration. One is when the motor is rotating forward (the left drawing) and the other is backward (the right drawing).

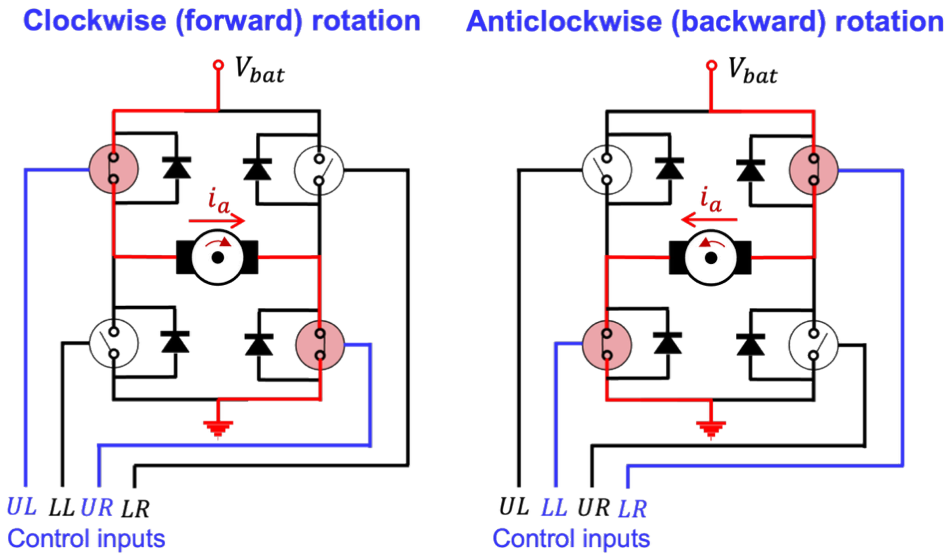


Figure 5.24: Braking regeneration of a DC motor in forward and backward directions with an H-bridge.

Consider the case of braking regeneration when the motor is rotating in the forward direction. We can see that only the lower-right switch is turned on and there is no applied armature voltage. In this case, the back EMF voltage ($K_m\omega$) is in the opposite direction of the battery. As a result, the armature current flows from right to left of motor to the battery due to activated upper-left red diode, which charges the battery. In addition, for braking regeneration, the upper-left switch cannot be turned on. If so, with both upper-left and lower-right switches turned on, the DC motor is in the drive mode.

The right drawing shows the case of braking regeneration when the motor is rotating in the anticlockwise (backward) direction. The analysis is similar to the case when the motor rotates in the clockwise (forward) direction; we will not repeat it here.

Example 5.10

Consider the braking regeneration when the motor rotates in the clockwise (forward) direction. In the left drawing of Figure 5.24, we can achieve braking regeneration by turning on only the lower-right switch. Now let's find another operational mode for braking regeneration, with the motor rotating in the clockwise (forward) direction, but different from turning on the lower-right switch. We can implement this other operational mode by turning on the upper-left switch, as illustrated in Figure 5.25.

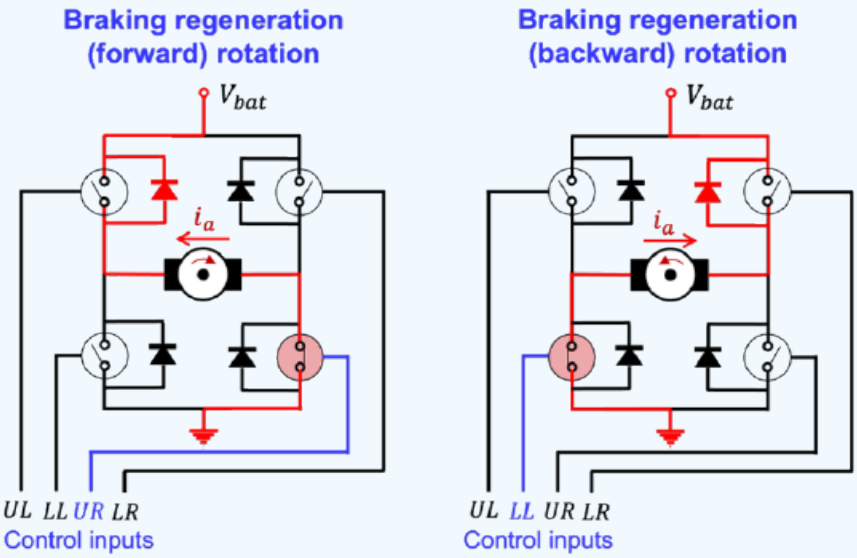


Figure 5.25: DC-motor braking regeneration in forward direction with an H-bridge.

Table 5.2 summarizes the four-switch H-bridge operational mode. Note that during the braking regeneration, a charge control circuit will be implemented to harvest the regeneration energy at low motor speed.

Table 5.2: Four-switch H-bridge operational modes.

Upper left	Upper right	Lower left	Lower right	Description
On	Off	Off	On	Forward running
Off	On	On	Off	Backward running
Off	Off	Off	On	Forward regeneration
On	Off	Off	Off	Forward regeneration
Off	Off	On	Off	Backward regeneration
Off	On	Off	Off	Backward regeneration

Figure 5.26 presents a sample four-switch H-bridge drive circuit for DC motors, where eight transistors and four diodes are used. Transistor T_s (e.g., TIP31) is used as a switch to handle a relatively large motor current. Transistor T_i (e.g., 2N3904) is used for amplifying an input PWM signal. Diode D_m is used for handling the current right after the switch transistor T_s is turned off as well as the current during braking regeneration.

Resistors R_i and R_l are input and amplifier load resistors, respectively. Note that, normally, $R_i \gg R_l$. For example, R_i could be selected as 1.2 $k\Omega$ and R_l 65 Ω. The sample circuit uses conventional transistor as the switch element. In many applications, MOSFET (metal-oxide-semiconductor field-effect transistor) and IGBT (isolated-gate bipolar transistor) are widely used to handle high-power switching.

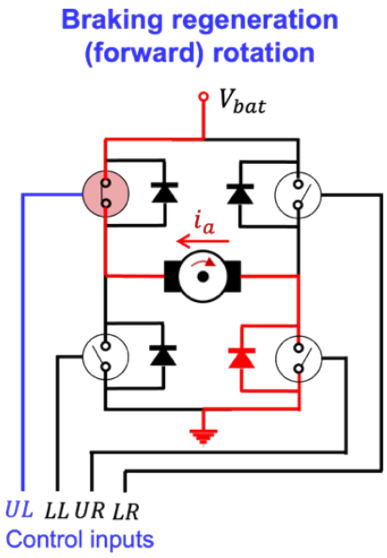


Figure 5.26: A drive circuit for a sample four-switch H-bridge DC motor.

Earlier in this section, we discussed using analog and PWM signals to drive the low- and high-side circuit of a DC motor. Now the question is whether the PWM control signal could affect the DC-motor speed or displacement response. Next, we will compare the difference between analog and PWM drive signals through a simulation study in Simulink [22].

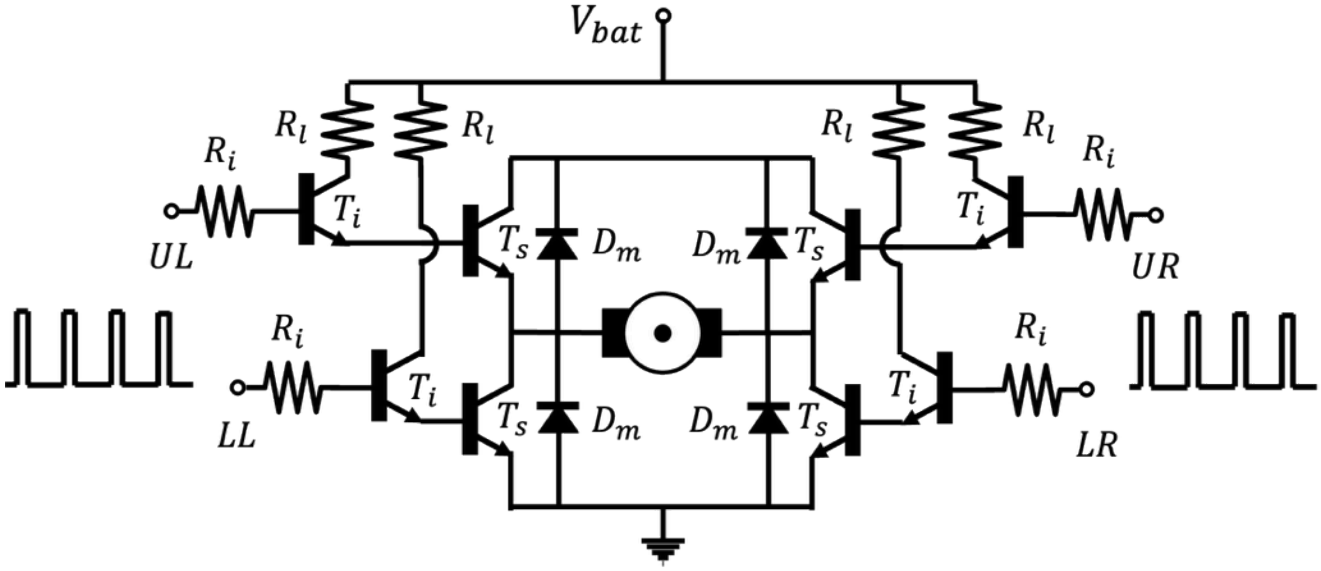


Figure 5.27: Simulink diagram for simulation of analog and PWM inputs to a DC motor.

Figure 5.27 shows two Simulink diagrams used for this study. The only difference between the top and bottom diagrams is the input (armature voltage) to the DC-motor model. The top plot has an analog voltage of 1.0 volt whereas the bottom plot has the PWM signal as input with a bottom-to-peak voltage of 5 volts and 20% duty-cycle. The equivalent voltage generated by the given PWM signal can be calculated below, using equation ??? in Section 4.6.

$$v_{analog} = v_{bottom-peak} \times duty_{cycle} = 5 \times 0.2 = 1\ V \tag{5.3.7}$$

In Figure 5.27 the motor torque coefficient $K_\tau = 0.23\ Nm/A$ and the back EMF coefficient $K_m = 0.027\ V/(rad/s)$. The motor armature winding resistance is $R_a = 4.8\ \Omega$ and the armature inductance is $L_a = 2\ mH$. For the mechanical part of model, the motor rotor inertia is $J_m = 0.0019\ kgm^2$ and the rotational friction coefficient is $b_m = 0.004\ Nm/(rad/s)$. You can find the corresponding coefficients in in Figure 5.27. To learn more about the graphic simulation software Simulink, see the MATLAB/ Simulink website or reference [22].

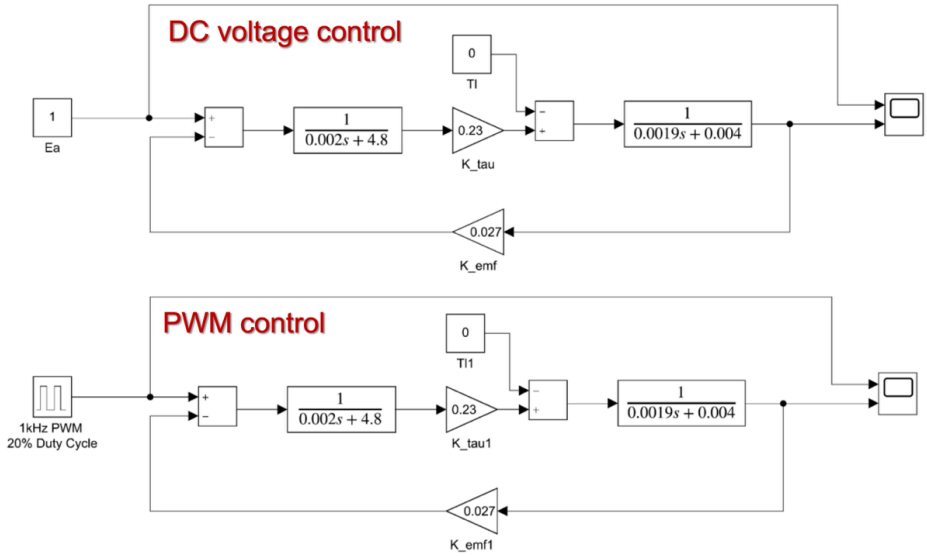


Figure 5.28: Simulink simulation results with analog and PWM inputs to a DC motor.

Figure 5.28 provides Simulink simulation results. The left plot presents the step response (blue line) with an analog step input of 1 volt (red line). The right plot presents the step response (blue line) with a PWM step input of 20% duty-cycle with bottom-to-peak of 5 volts (red line and the zoomed-in PWM signal). For the analog step input, the response shown in the left plot converges to its steady state within 1.5 seconds. We can calculate the approximated DC-motor system time constant T using equation ??? as below.

$$T = \frac{J_m R_a}{R_a b_m + K_\tau K_m} = \frac{0.0019 \times 4.8}{4.8 \times 0.004 + 0.23 \times 0.027} = 0.359 \ second \tag{5.3.8}$$

As a result, for this case, we can calculate the 2% settling time $t_s(2\%)$ of the DC-motor system as below.

$$t_s(2\%) = 4 \times T = 4 \times 0.359 = 1.436 \ second \tag{5.3.9}$$

Therefore, the simulated system response for the analog step input is very close to the analytic settling time. Now, considering the PWM response on the right plot of Figure 5.28, we can see that the motor speed response (blue line) is almost identical to the analog step response (left plot). This is mainly because a 1-kHz PWM signal is used for the PWM step signal; the motor system with a time constant of $T = 0.359$ second basically filters out the pulsation of the PWM signal. Only the mean effect of the PWM signal, equivalent to a 1.0-volt analog step, remains.

Figure 5.29 shows a Simulink simulation diagram for closed-loop motor speed control. The right red-dashed box, DC-motor model, is the model given in Figure 5.27. The middle red-dashed box, PWM-drive model, converts the analog control signal generated by the PID controller into the zero-to-five volt PWM signal (500 Hz). The left red-dashed box, PID controller, is an analog PID controller (see Section 3.4). A negative feedback loop is formed by subtracting the motor output speed from the reference speed. This obtains the speed error signal to be used as the input to the PID controller with proportional gain $K_p = 5$, integral gain $K_i = 2$, and derivative gain $K_d = 0$.

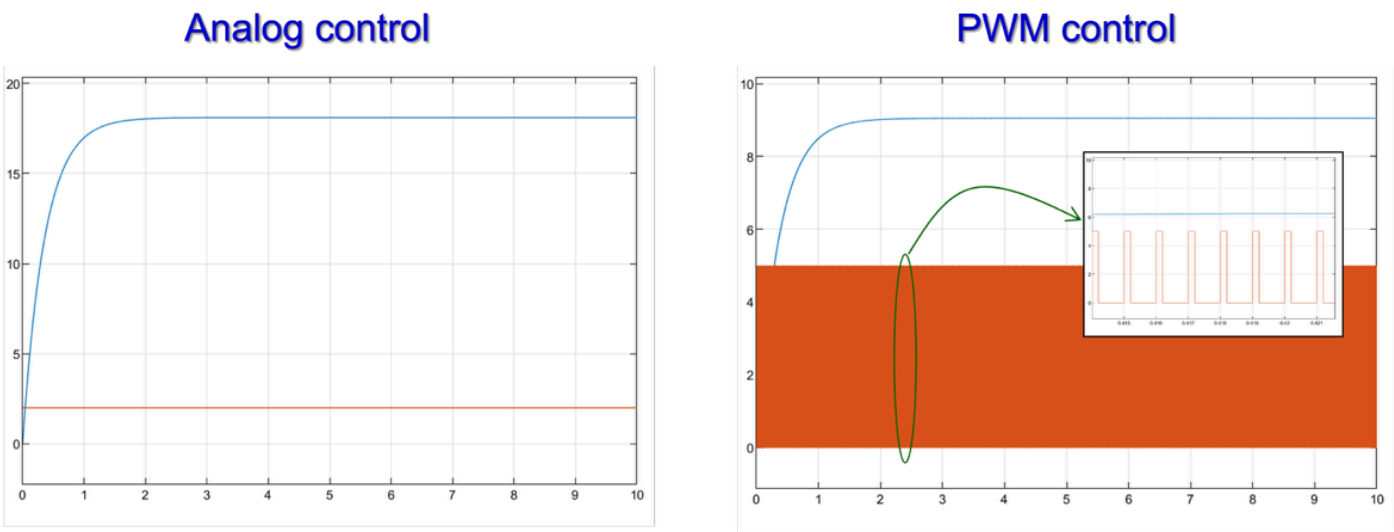


Figure 5.29: Simulink simulation diagram for closed-loop control of DC-motor speed.

Figure 5.30 presents the simulation results for the Simulink diagram shown in Figure 5.29. The top plot shows the analog control signal produced by the PID controller. The middle plot shows the PWM control signal. The bottom plot shows the time response of the DC-motor speed. The analog signal is saturated by ± 10 volts, where 10 volts corresponds to the 100% duty-cycle of zero-to-five-volt PWM signal. This results in the saturated PWM signal. From the middle plot, we can see that the PWM control signal is saturated between zero and 0.19 second to accelerate the motor speed at it maximum PWM duty-cycle of 100%. As the motor speed approaches the reference speed of 20 rad/s, the PWM duty cycle reduces to about 50%, with the analog control signal close to 5 volts. The bottom plot shows that the settling time with closed-loop control is about 0.2 second, compared with the open-loop case of about 1.5 second; see equation 5.3.9. This indicates that closed-loop control can significantly reduce the motor speed response time.

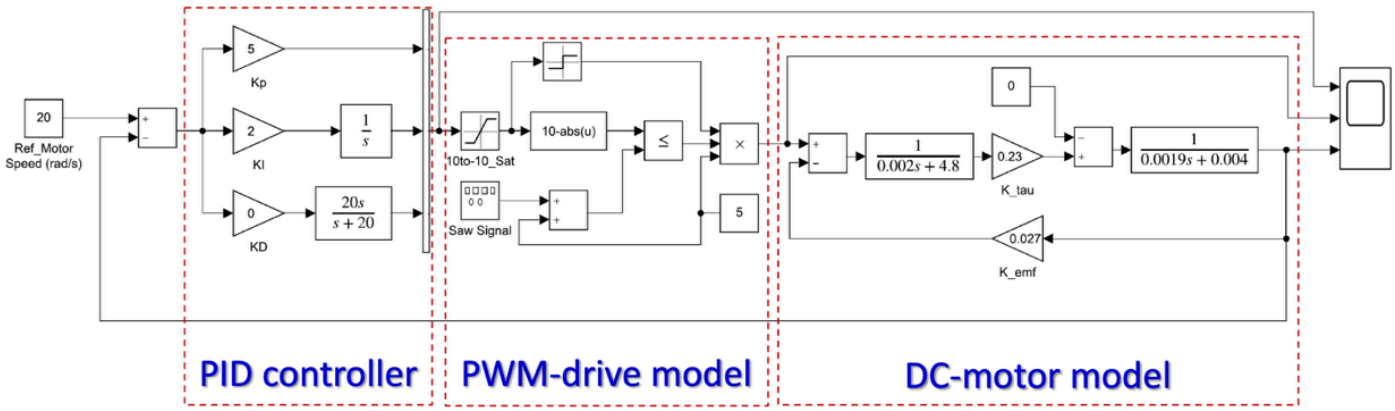


Figure 5.30: Simulink simulation results of closed-loop DC-motor speed control.

Before ending this section, let's consider a class of actuators called servomotors. A servomotor is a closed-loop angular displacement control system. It consists of a DC- (or AC-) motor actuator, a gear system to reduce motor speed, a displacement sensor (e.g., voltage divider) for measuring the angular displacement, and motor control electronics for regulating the angular displacement. Figure 5.31 shows a small servomotor system using a DC motor.

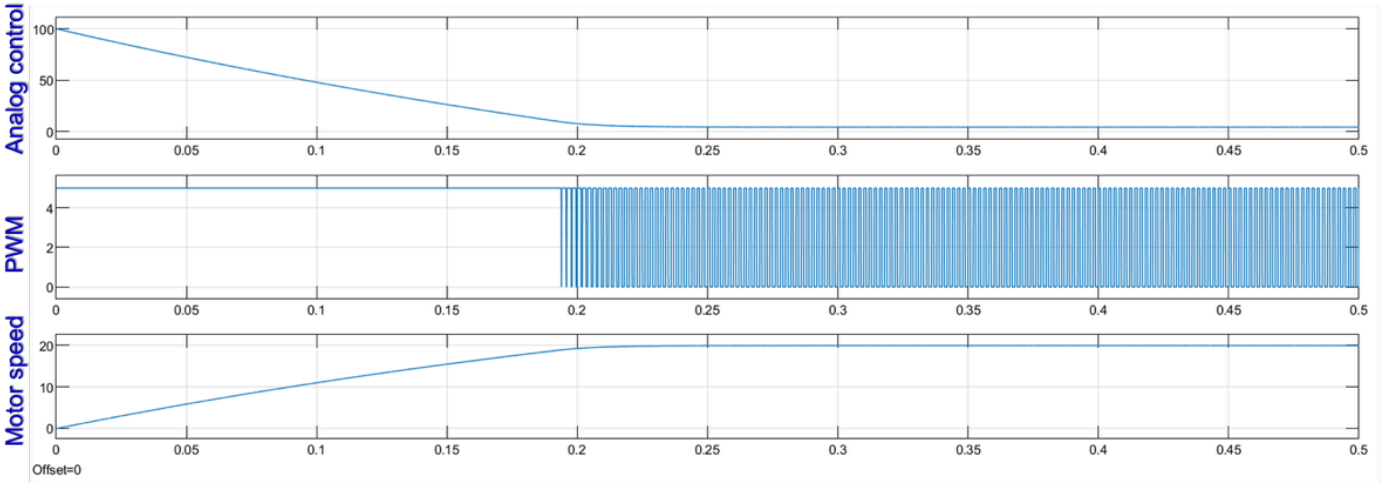


Figure 5.31: A sample servo motor actuator.

Servo motors have several advantages. They have high intermittent torque output, high torque to inertia ratio, and fast response speed. They are good for angular velocity and position control. Servo motors are available in all sizes and they have smooth rotation at low speed due to gearbox. Servo motors also have disadvantages. They are more expensive than stepper motors for displacement control. They require tuning control loop for optimal performance. Furthermore, they are not suitable for either hazardous environment or excessive power (load) applications that could result in partially damaging DC servo motor.

Section Summary

This section addresses the methodologies of driving the DC motor efficiently, including low-side and high-side drives, the PWM (pulse-width-modulated) drive, and the widely used H-bridge drive based on the PWM signal. Note that the H-bridge drive enables the generation of the motor torque in two directions: positive and negative. This is very important for closed-loop control of displacement or velocity with a DC motor as the actuator. The section also discusses the associated DC-motor drive circuits as well as the advantages and disadvantages of different drive circuits.

Section Keywords

Practice Problems

1. For the analog high-side drive circuit presented on Figure 5.20, with the sensing resistor value of $R_s = 0.1\ \Omega$, find reference voltage v_{ref} to achieve the target DC-motor armature current of $i_a = 15\ A$.
2. Assume that a 0-to-5 volts PWM reference v_{pwm} has its duty-cycle of 20%, the battery voltage is $V_{bat} = 12.2$ volts, DC-motor armature resistance R_a is 0.2 Ω, and the voltage drop of transistor T_m in Figure 5.21 is v_{e2c}= 0.2 when it is fully turned on. Find the resulting armature current in Figure 5.21.
3. Consider the braking regeneration when the motor rotates in the clockwise (forward) direction. In the right drawing of Figure 5.24, by turning on only the lower-left switch, braking regeneration can be accomplished. Find another braking regeneration operational mode with the motor rotating in the anticlockwise (backward) direction, different from the case with the lower-left switch turned on.
4. Construct the Simulink model shown in Figure 5.29. Tune the PID controller to see if the settling time can be reduced.

This page titled 5.3: DC-Motor Drive Circuits is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

5.4: Discrete Electromagnetic Valve Actuation (EMVA)

Learning Objectives

After completing this section, you will be able to:

1. Describe the working principle of an electromagnetic valve actuation (EMVA).
2. Define three states of the EMVA system.
3. Construct the Simulink model of the EMVA system and conduct associated simulation study.

This section discusses a family of electromagnetic actuation systems that have discrete states. That is, instead of producing a proportional torque, rotational displacement by the actuator, it maintains a number of discrete states based upon the input command. In this section, we will use a three-state example, an electromagnetic valve actuator (EMVA) [4], [23], to introduce this kind of an actuation system. Note that there are also electropneumatic valve actuation (EPVA) [24] and electrohydraulic valve actuation (EHVA) [6], [7] systems developed for internal combustion (IC) engines. EHVA systems will be addressed in the next section.

The EMVA system was developed for actuating the intake and exhaust valves of internal combustion (IC) engines to have flexibility such as adjusting valve timing and opening duration. The traditional IC engines have valve systems driven by camshafts with limited capability of changing the valve opening profiles; see [5] for a detailed discussion and review of IC engine valve systems. To optimize the IC engine combustion process, opening profiles of both intake and exhaust valves need to be optimized and adjusted under different operational conditions. These actions lead to the development of the so-called camless valve actuation systems. The electromagnetic valve actuator (EMVA) is one such actuation system. Other actuation systems include EPVA and EHVA.

Valve opening and closing timings of conventional IC engines are directly driven by engine camshafts that links the valve timing to the crankshaft position or through the so-called cam phaser that makes the connection between the crankshaft and cam shaft [5]. As a result, the valve timing driven by the camshaft ensures having correct valve opening and closing timing with respect to engine crankshaft position. Meanwhile, the valve timing driven by camshaft through the cam phaser is capable of changing valve opening timing, with the difference between opening and closing timing fixed.

For fixed valve timing, engine performance can be optimized only at a single operational condition with tradeoff over other operational conditions. For the cam phaser case, the engine operation can be optimized at an individual operational condition. However, the engine operation can be further optimized if the valve opening duration and lift can be adjusted. This is the main motivation of developing camless valve systems.

An EMVA system consists of an armature mounted between two sets of springs (upper and lower) and electromagnetic solenoids (upper and lower), as shown in Figure 5.32. Normally, the upper spring is part of the EMVA actuator whereas the lower spring is part of the engine valve system. Both upper and lower springs are preloaded such that the armature sits at rest equidistant from either electromagnetic solenoid.

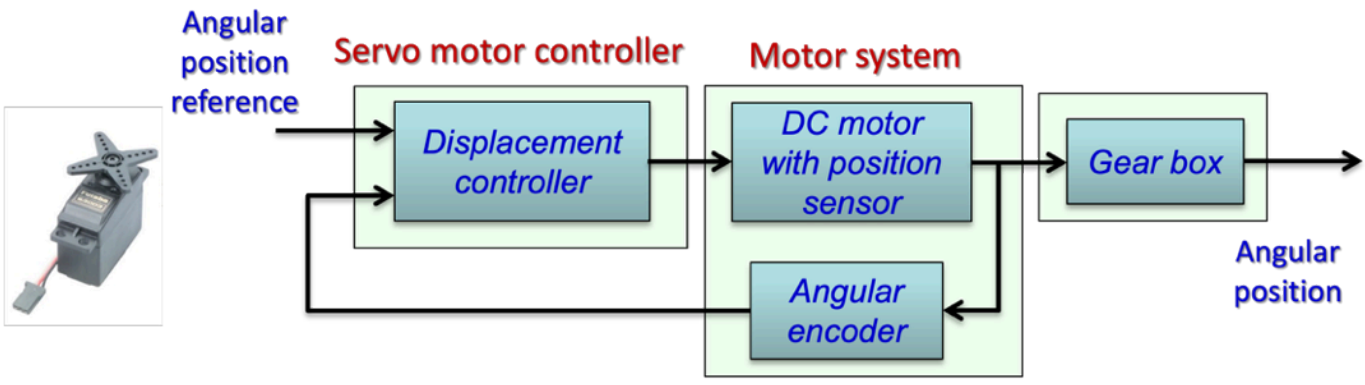


Figure 5.32: EMVA system architecture.

An EMVA has three valve states as seen in Figure 5.33. These are the top position with the upper electromagnetic solenoid activated, the middle position with no solenoid activated, and the bottom position with the lower solenoid energized. When both electromagnetic solenoids are not energized at the steady-state, the valve remains at its neutral position; it is held by both preloaded upper and lower springs (see the middle plot of Figure 5.33). When the valve's upper electromagnetic solenoid is energized at the steady-state, the valve remains at its closed position; the armature is held against the upper solenoid, causing the upper spring to be more compressed than the lower spring (see the left plot). When the valve's lower electromagnetic solenoid is energized at the steady-state, the valve remains at its open position; the armature is held against the lower solenoid, causing the lower spring to be more compressed than the upper spring (see the right plot).

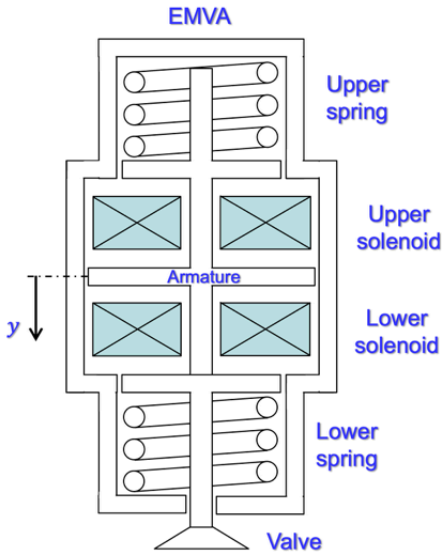


Figure 5.33: Three states of the EMVA system.

To open the valve from its closed position, we use a two-step process. First, we turn off the electric power that had been applied to the upper magnetic solenoid. This results in the armature being driven downward to the lower coil by the force imbalance in the springs, with the upper spring force greater than the lower one. Second, we apply the electric power to the lower coil to catch the armature and hold the valve in the open position. As Figure 5.33 shows, the valve opening process starts with the left plot, moves to the middle plot, and finally stops at the right plot.

The valve closing process is realized by reversing this two-step process. First, we turn off the electric power that had been applied to the lower magnetic solenoid. This results in the armature being driven upward to the upper coil by the force imbalance in the springs, with the lower spring force greater than the upper one. Second, the electric power is then applied to the upper coil to catch the armature and hold the valve in the closing position. As Figure 5.33 shows, the valve closing process starts with the right plot, moves to the middle plot, and finally stops at the left plot.

Many models have been developed for EMVA systems in the past. We will use the model developed by Wang [25] as an example. When only the lower coil is energized, the EMVA system dynamics can be described by the following equations.

$$\begin{cases} \dfrac{dy}{dt} &= v \\ \dfrac{di_{low}}{dt} &= \dfrac{u_{low} - R_{low} i_{low} - K_{emf\_low}\left(i_{low}, y\right) v}{L_{low}\left(i_{low}, y\right)} \\ \dfrac{dv}{dt} &= \dfrac{1}{m}\left(F_{mag\_low}\left(i_{low}, y\right) + k_s\left(l - y\right) - bv\right) = \dfrac{dy^2}{dt^2} \end{cases} \tag{5.4.1}$$

where y in meter is the distance between the armature and the lower coil in the direction shown in Figure 5.32; v in meter/second is the velocity of the armature, and i_{low} in ampere (A) is the current of the lower coil. In addition, m in kilogram is the combined mass of the armature and valve plus effective spring mass; $F_{mag\_low}\left(i_{low}, y\right)$ in Newton is the magnetic force generated by the lower coil; k_s in Newton/meter is the combined spring constant of upper and lower ones; l in meter is half of the total armature travel; b in Newton-second/m is the damping coefficient; u_{low} in volt is the voltage applied to the lower coil; R_{low} in Ω is the combined lower winding resistance of the wiring and magnetic coil winding; $K_{emf\_low}\left(i_{low}, y\right) v$ in volt is the back-EMF effect due to the armature motion with $K_{emf\_low}\left(i_{low}, y\right)$ being the back-EMF coefficient of the lower solenoid; and $L_{low}\left(i_{low}, y\right)$ in Henry (H) is the inductance

of the lower magnetic solenoid.

When no solenoid is energized, the equation of motion is shown below.

$$\begin{cases} \dfrac{dy}{dt} &= v \\ \dfrac{dv}{dt} &= \dfrac{1}{m}\left(k_s\left(l-y\right)-bv\right)=\dfrac{dy^2}{dt^2} \end{cases} \tag{5.4.2}$$

Last, when only the upper coil is energized, the EMVA system dynamics can be described by the following equations.

$$\begin{cases} \dfrac{dy}{dt} &= v \\ \dfrac{di_{up}}{dt} &= \dfrac{u_{up}-R_{up}i_{up}-K_{emf\_up}\left(i_{up},y\right)v}{L_{up}\left(i_{up},y\right)} \\ \dfrac{dv}{dt} &= \dfrac{1}{m}\left(-F_{mag}\left(i_{up},y\right)+k_s\left(l-y\right)-bv\right)=\dfrac{dy^2}{dt^2} \end{cases} \tag{5.4.3}$$

Here, i_{up} in ampere (A) is the current of the upper coil; $F_{mag\_up}\left(i_{up},y\right)$ in Newton is the magnetic force generated by the upper coil; u_{up} in volt is the voltage applied to the upper coil; R_{up} in Ω is the combined upper winding resistance of the wiring and magnetic coil winding; $K_{emk\_up}\left(i_{up},y\right)v$ in volt is the back-EMF effect due to the armature motion with $K_{emf\_up}\left(i_{up},y\right)$ being the back-EMF coefficient of the upper solenoid; and $L_{up}\left(i_{up},y\right)$ in Henry (H) is the inductance of the upper magnetic solenoid. The rest of variables were defined earlier.

In the EMVA operation, the lower and upper solenoids are not energized simultaneously because the lower and upper solenoids generate forces in opposite directions. System differential equations 5.4.1, 5.4.2, and 5.4.3 can be combined into single one as below.

$$\begin{cases} \dfrac{dy}{dt} &= v \\ \dfrac{di_{low}}{dt} &= \dfrac{u_{low}-R_{low}i_{low}-K_{emf\_low}\left(i_{low},y\right)v}{L_{low}\left(i_{low},y\right)},\ \dfrac{di_{up}}{dt}=\dfrac{u_{up}-R_{up}i_{up}-K_{emf\_up}\left(i_{up},y\right)v}{L_{up}\left(i_{up},y\right)} \\ \dfrac{dv}{dt} &= \dfrac{1}{m}\left(F_{mag\_low}\left(i_{low},y\right)-F_{mag\_up}\left(i_{up},y\right)+k_s\left(l-y\right)-bv\right)=\dfrac{dy^2}{dt^2} \end{cases} \tag{5.4.4}$$

This equation set describes the linear motion of the EMVA system without hard travel limits for y. For practical engine EMVA systems, there are hard travel limits: y_{min} (closed) and y_{max} (fully opened). One of the major challenges of EMVA is dealing with the hard landing when the valve travels close to its closing limit y_{min} or opening limit y_{max} because hard landing reduces the EMVA opening and closing performance and valve system durability [26].

If the EMVA is not properly controlled, the armature could accelerate toward the hard limit in an unbounded fashion, resulting in large impact speed up to 1-to-2 m/s [27]. Besides potentially damaging the actuator, these impacts generate unacceptably loud noise. Many control techniques have been deployed to significantly reduce the impact speed. These include sensor-less control [28], [29]; closed-loop control [4]; linear quadratic regulation (LQR) [26], [30]; nonlinear control [31]; and $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control [6]. Interested readers can find more details from these references.

In Chapter 2, we discussed the modeling process for the DC motor, a mechatronic actuator. Let's compare the EMVA system model of the lower solenoid described in equation 5.4.1 with the DC-motor actuator. From the middle equation of 5.4.1, we have the following after reorganization.

$$u_{low}=R_{low}i_{low}+L_{low}\left(i_{low},y\right)\frac{di_{low}}{dt}+K_{emf\_low}\left(i_{low},y\right)v \tag{5.4.5}$$

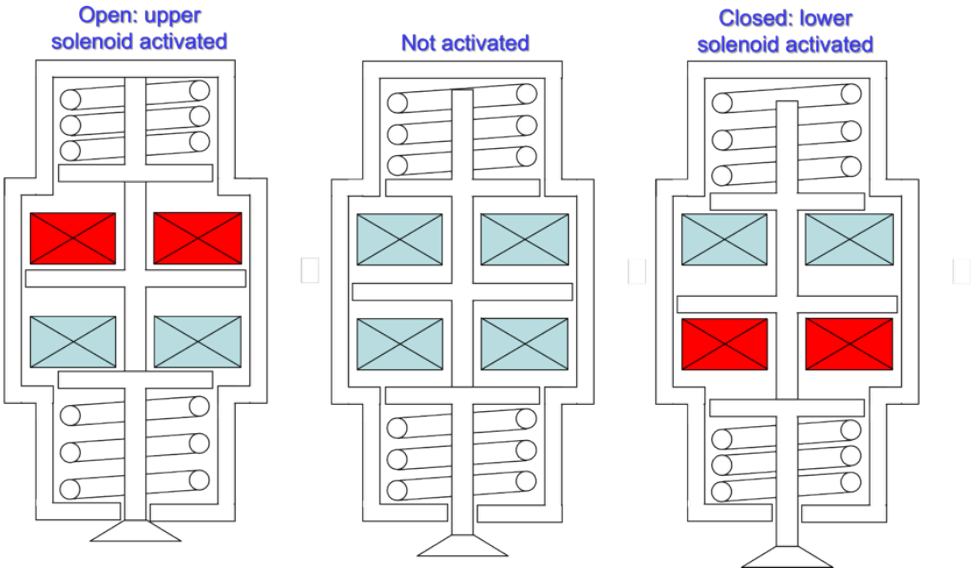


Figure 5.34: Comparison of DC-motor and EMVA electrical circuits.

Equation ??? has the same form of equation 5.4.5. The DC-motor armature resistor R_a is equivalent to the lower solenoid resistor R_{low}; the DC-motor armature inductor L_a is equivalent to the lower solenoid inductor L_{low}; and the DC-motor back-EMF coefficient K_m is equivalent to the lower solenoid back-EMF coefficient K_{emf_low}. There are three main differences between the DC motor and EMVA equivalent circuits shown in Figure 5.34. First, the lower solenoid inductance, $L_{low}\,(i_{low}, y)$, is not a constant (true for the DC-motor case), but a function of the lower solenoid current i_{low} and the valve armature displacement y. Second, the lower solenoid back-EMF coefficient is not a constant, but a function of the lower solenoid current i_{low} and the valve armature displacement y. Last, the DC-motor motion is rotational in $\dot{\theta}$ (the left plot of Figure 5.34), but the EMVA is linear in $\dot{y} = v$ (the right plot of Figure 5.34). Therefore, there is a strong correlation between the DC-motor and EMVA equivalent circuit models.

In Example 5.11, we will develop a simplified Simulink EMVA model. We will then conduct the simulation study of EMVA valve motion with a hard stop to show that it is important to tune the EMVA upper and lower solenoid control signals in order to have a soft landing. The rest of this section presents a simulation study in Example 5.11.

Example 5.11

Assume that both lower and upper solenoids have the same property. The back-EMF coefficient and solenoid inductance can then be defined using data from reference [23], assuming that valve displacement is between 0 and $2l$.

$$K_{emf\_low} = \frac{2k_a i_{low}}{(k_b + (2l - y))^2}, K_{emf\_up} = \frac{2k_a i_{up}}{(k_b + y)^2}$$
$$L_{low} = \frac{2k_a}{k_b + (2l - y)}, L_{up} = \frac{2k_a}{k_b + y} \tag{5.4.6}$$

In this case, both K_{emf_low} and K_{emf_up} are functions of the associated current (i_{low} and i_{up}) and relative displacement ($2l - y$ and y), respectively. Both L_{low} and L_{up} are functions of the associated relative displacement ($2l - y$ and y), respectively. Parameters k_a and k_b are constant. They can be calibrated using the experimental data [23]. Parameter Numerical Value Parameter Numerical Value m 0.27 kg k_a 14.96×10^{-6} Nm^2/A^2 k_s 158×10^3 N/m k_b 4.0×10^{-5} m l 4.0×10^{-3} m r 6.0 Ω b 7.53 Nm/s Table 5.3: EMVA parameters used in Example 5.11. The magnetic forces generated by both solenoids are defined below.

$$F_{mag\_low} = \frac{k_a i_{low}^2}{(k_b + (2l - y))^2}, F_{mag\_up} = \frac{k_a i_{up}^2}{(k_b + y)^2} \tag{5.4.7}$$

where, both magnetic forces F_{mag_low} and F_{mag_up} are quadratic functions of the associated current (i_{low} and i_{up}) and inverse functions of relative displacement ($2l - y$ and y), respectively. Table 5.3 defines the system parameters used to develop the EMVA model in this example, with all parameters in international units.

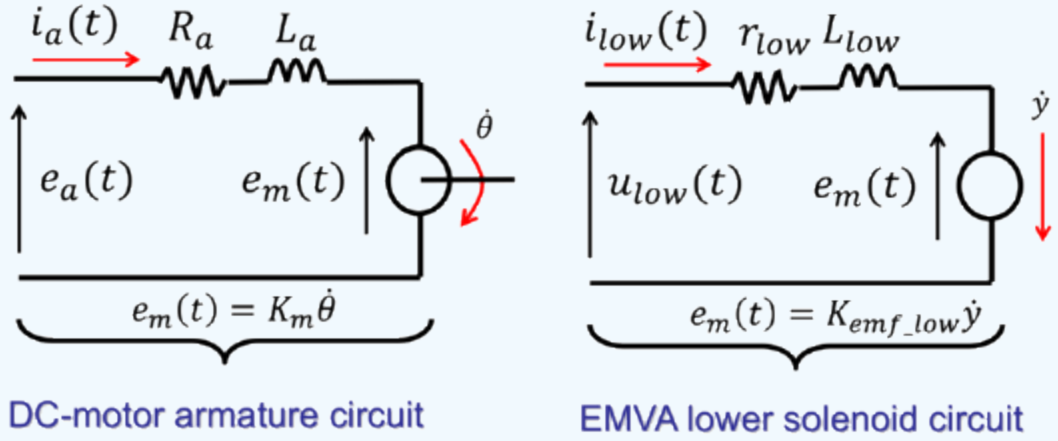


Figure 5.35: Simulink simulation model with valve dynamics and upper/lower solenoid actuators.

Based upon equations 5.4.4, 5.4.6, and 5.4.7, we can develop a Simulink simulation model. In Figure 5.35, the middle section with the red dash-dot line box describes the second-order valve mechanical dynamics—see the bottom of equation 5.4.4 —with valve input forces generated by both the lower and upper solenoids. The lower and upper solenoid dynamics are modeled above and below the red dash-dot box, respectively. Output multiplier 2000 2000 1.0 1.0 2000 2000 Input $y - l$ (m) -1000 $-l \times 0.97$ $-l \times 0.88$ $l \times 0.88$ $l \times 0.97$ 1000 Output multiplier 300 300 1.0 1.0 300 300 Input $y - l$ (m) -1000 $-l \times 0.97$ $-l \times 0.95$ $l \times 0.95$ $l \times 0.97$ 1000

Table 5.4: EMVA mechanical model multipliers for parameters b and k_s.

To model the hard limitations ($y_{min} = 0$ and $y_{max} = 2l$), we can apply the valve displacement limitation using the nonlinear spring characteristics close to both ends of the travel (y_{min} and y_{max}). Here, two one-dimension lookup tables in Table 5.4 are used as multipliers with input axis as the valve displacement $y - l$ for both the damping and stiffness coefficients (b and k_s). The reason for using the displacement as the lookup table input is to model the nonlinear characteristics of the valve travel close to the mechanical hard limit. For example, the extremely high damping coefficient multiplier is used to simulate a hydraulic damper. In addition, high stiffness close to the end of the travel is used to simulate the hard mechanical limit of the valve travel. The saturation on the displacement integration block is set to be between $y_{min} = 0$ and $y_{max} = 2l$; here the initial condition is $y_0 = 0.000199$ meter and the initial condition for the speed integration block is zero. The lower solenoid actuation Simulink model is developed based upon the middle of equation 5.4.1 and equations 5.4.6 and 5.4.7. (See the red dash-dot line box in Figure 5.35.) The current integration block has an initial condition of zero with the reset logic enabled. That is, when the logic input is high (active), the integration is disabled and set to zero until it becomes low (inactive). In practice, when the lower solenoid is inactive, the circuit is open and there is no current generated due to back-EMF effect, even with the valve movement. A lookup table with clock as input is used to generate voltage input to the lower solenoid to generate the desired movement (see the top plot of Figure 5.36 for details), where the lower solenoid lookup table value is shown in Table 5.5, along with the upper one. Output voltage 380 380 120 12 12 0 0 0 Input time (ms) 0 1.5 2.2 6 14.95 15 25 40 Output voltage 0 0 380 380 120 12 12 12 Input time (ms) 0 14.95 15 16.2 17.5 21 25 40

Table 5.5: EMVA lower and upper solenoid input voltage profile.

The upper solenoid model in Figure 5.35 (see the Simulink diagram above the red dash-dot lined box) is based upon the middle of equation 5.4.3 and equations 5.4.6 and 5.4.7. Here, the current integration block has an initial condition of zero with the reset logic enabled similar to the lower solenoid case. That is, when the logic input is high (active) the integration is disabled and set to zero until it becomes low (inactive). Table 5.5 provides two lookup tables with time as input and applied solenoid voltage as output. It generates voltage inputs to both lower and upper solenoids to achieve the desired movement (see the top plots of Figure 5.36 and Figure 5.37, respectively).

Figure 5.36 shows the simulation results of the lower solenoid for the valve opening operation. In this case, the initial condition is at the fully closed position ($y \approx 0\ mm$), where the voltage control input to the solenoid is shown in the top plot with a peak voltage of 380 volts. The solenoid current is shown in the middle plot with a peak current of 55 ampere. The force generated by the lower solenoid is shown in the bottom plot with a peak holding force of about 1050 Newtons. The close-to-zero initial condition is caused by very high stiffness modeled near the end of travel when the valve is closed. Note that the

response time of generated force is about $3.5\ ms$ whereas the valve response time is about $3\ ms$; the valve did not travel all the way to 8 mm with a small difference due to the high stiffness at the end of travel. The other observation is that the force response is slower than the current one. The major response delay is due to the large solenoid impedance (inductance). The peak solenoid inductance is about 0.6 Henry, leading to slow conversion from input voltage to current. As a result, closed-loop control of valve displacement is challenging due to this large delay.

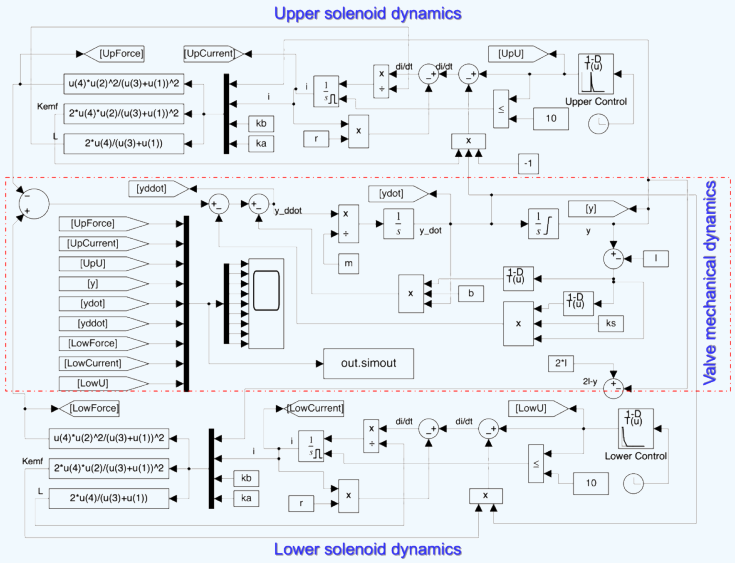


Figure 5.36: EMVA lower solenoid control voltage, current, and generated force.

Figure 5.37 shows the simulation results of the upper solenoid for the valve closing event with initial valve fully opened ($y \approx 8\ mm$). Here, the voltage control input to the solenoid is shown on the top plot with a peak voltage of 380 volts, starting at the fifteenth millisecond. The solenoid current is shown in the middle plot with a peak current of 55 ampere. The force generated by the upper solenoid is shown in the bottom plot with a peak force of about 1050 Newtons. Again, the valve did not travel to $0\ mm$ due to high spring stiffness modeled at the end of travel.

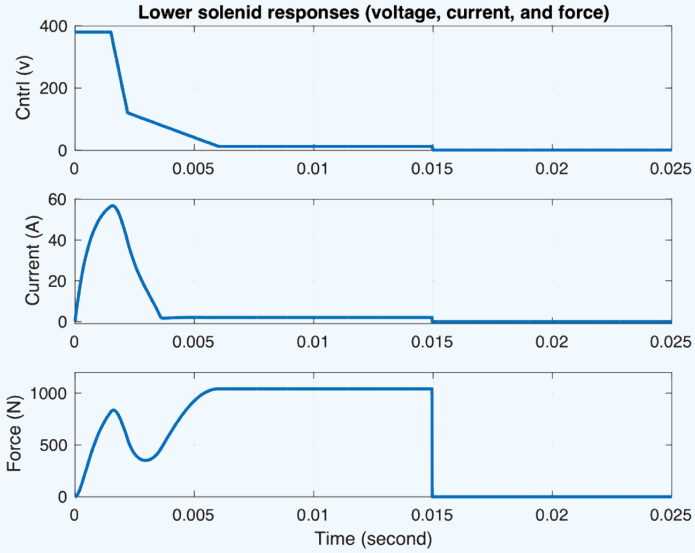


Figure 5.37: EMVA upper solenoid control voltage, current, and generated force.

Note that the response time of generated force is again about $3.5\ ms$ (both solenoids are modeled with the same characteristics) while the valve response time is about $3\ ms$. In addition, the force response is slower than the current one. The major response delay is due to the solenoid impedance (inductance) as high as 0.6 Henry, leading to slow conversion

from input voltage to current. A reminder once again that the EMVA has three states: closed, resting, and opened. The simulation results in this example show that energizing the upper solenoid leads to closing the valve and energizing the lower solenoid, which results in opening the EMVA within about $3.5\ ms$.

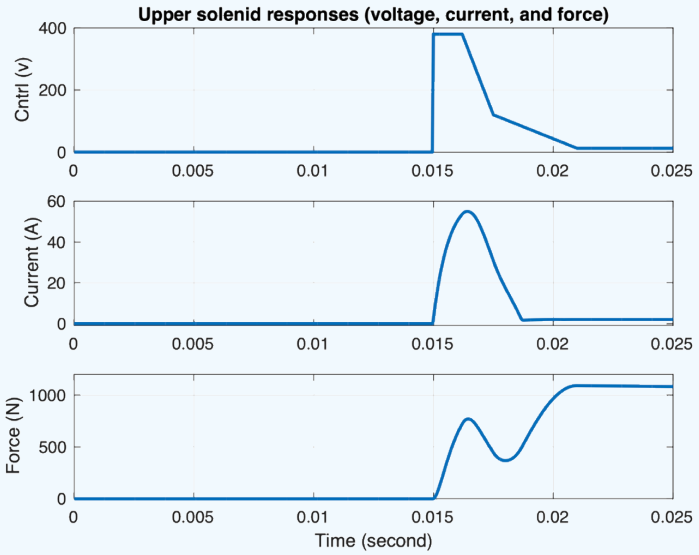


Figure 5.38: EMVA displacement, speed, and acceleration responses.

The valve responses including displacement (top), velocity (middle), and acceleration (bottom) are plotted in Figure 5.38. We can see from the top plot that the valve can be opened and closed within about $3\ ms$, which satisfies the internal combustion engine valve train requirements. From the middle plot, we can see the valve peak speed is about $3.5\ m/s$, which is a little high. Finally, from the bottom plot, we can see the maximum absolute acceleration is high at about $1.9 \times 10^4\ (m/s^2)$, which can be reduced by using hydraulic damper at both ends.

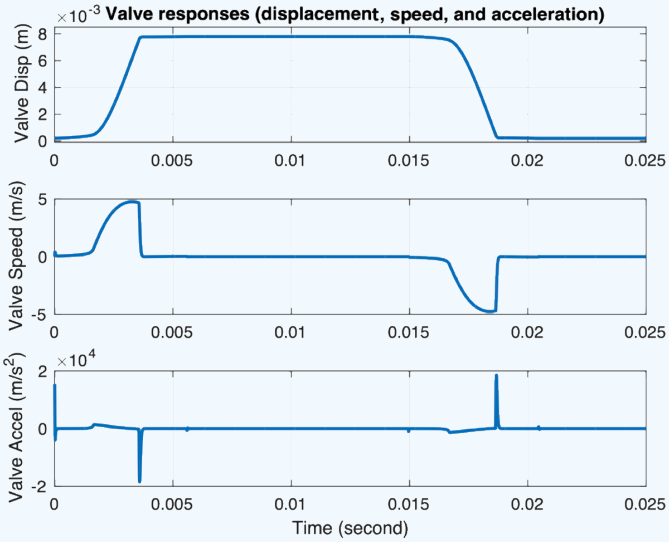


Figure 5.39: EMVA free travel response with initial valve position of close to 0 mm.

Now we will address the resting state with both solenoids disengaged. In theory, under this operational mode, either the EMVA will remain in the resting position with $y = l$ or it will converge to its resting from any initial displacement. Note that the bottom equation of 5.4.4 can be reorganized as follows by setting $F_{mag\_low} = F_{mag\_ip} = 0$.

$$m\frac{d(y-l)^2}{dt^2} + k_s\,(y-l) + bv = 0 \tag{5.4.8}$$

From the stability theory discussed in Section 3.2, because both coefficients b and k_s are positive, the system is open-loop asymptotically stable. That is, for any initial condition, the system displacement and speed will converge to its origin $\dot{y} = y - l = 0$ or $\dot{y} = 0,\ y = l$. Therefore, the resting state is $\dot{y} = 0$ and $y = l$. Figure 5.39 shows the simulation results of the EMVA with both solenoids disengaged and an initial valve displacement of zero. The Simulink model in this case can be obtained by modifying the model shown in Figure 5.35. Here, set the inputs to both solenoids to zero (replacing both lookup tables with time input by a zero constant block). The initial condition $y_0 = 0$ can be achieved by setting the initial condition of the displacement integral to zero. From Figure 5.39, we can see that the displacement and speed initial conditions in this case are both zero. Then, the oscillated displacement and speed converge to its settling position of $y = l$ within 0.3 second. In this case, the natural frequency can be calculated in the linear range, as below.

$$\omega_n = \sqrt{\frac{k_s}{m}} = \sqrt{\frac{158 \times 10^3}{0.27}} = 764.97\ \ (rad/s) = 121.75\ \ (Hz) \tag{5.4.9}$$

By counting the number of peaks between 0 and 0.1 second, we find 12 bottoms for the displacement response in Figure 5.39, which confirms a natural frequency close to 120 Hz.

Section Summary

This section introduces electromagnetic valve systems [2], [3], [4], [23], where the electromagnetic intake and exhaust valve systems for internal combustion engines are used as an example. The electromagnetic valve system converts the electric energy to achieve the desired valve displacement. It uses an electromagnetic valve actuator with lower and upper solenoids. Instead of generating the desired torque from a DC motor, the electromagnetic valve actuator generates linear forces by both lower and upper solenoids. These forces move valve armature to achieve the desired position. Often, a closed-loop control system is needed to make sure that the valve reaches the desired displacement with small impact force applied to valve seats.

Section Keywords

Practice Problems

1. Which of the following systems are included in camless engine valve actuation? Select the correct answer.
   a. Cam phaser, EMVA, and EPVA
   b. EMVA, EHA, and cam phaser
   c. EMVA, EPVA, and EHVA
2. For the equivalent circuits of DC motor and EMVA, provide the three equivalent EMVA components associated with DC-motor armature resistance R_a, back-EMF coefficient K_m, and rotational speed $\dot{\theta}$.
3. For the DC motor, the magnetic torque generated by the motor equals $\tau = K_\tau i_a$, where i_a is the armature current. Please provide the magnetic force generated by the lower solenoid, with solenoid current defined as i_{low}. (Hint: see equations 5.4.1 and 5.4.7 for information.) Point out the difference between DC-motor torque and EMVA force as a function of current.
4. Using the Simulink diagram shown in Figure 5.35, change the upper solenoid input voltage lookup table to move the valve closing start timing from $15\ ms$ to $20\ ms$.
5. Modify the Simulink model in Figure 5.35 by replacing both solenoid input lookup tables with zero constant block and setting the displacement integral initial condition to $8\ mm$ $(0.008\ m)$. Conduct simulation in Simulink and plot the corresponding displacement, speed, and acceleration responses shown in Figure 5.39.

This page titled 5.4: Discrete Electromagnetic Valve Actuation (EMVA) is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

5.5: Discrete Electrohydraulic Valve Actuation (EHVA)

Learning Objectives

After completing this section, you will be able to:

1. Explain the working principle of electrohydraulic valve actuation (EHVA).
2. Provide the modeling process for the EHVA actuation.
3. Describe the LPV modeling technique for the highly nonlinear EHVA systems.

This section discusses the discrete electrohydraulic valve actuation system. Intensive research has been focused on camless variable valve actuation systems for internal combustion engines with three main actuation systems: electromagnetic (Section 5.4), electrohydraulic (this section), and electropneumatic [24] actuation systems. The electromagnetic valve actuation (EMVA) system provides fully flexible valve timing and duration with relatively high efficiency. However, its repeatability is highly sensitive to valve opening load (in-cylinder pressure at opening for internal combustion engine application) with fairly high electric power demand [33]. In addition, it also has high valve seating velocity due to the nonlinear electromagnetic force; it is limited to a fixed valve lift [5], [33]. The modeling and control of the EMVAs have been discussed extensively [4], [23] in Section 5.4.

Electrohydraulic valve actuation (EHVA) systems are usually considered as the most flexible variable valve actuation design using the pressurized hydraulic fluid to drive the engine valves. Valve timing and duration can be continuously varied because the hydraulic flow rate can be tailored by controlling solenoid valves [34]. These valves have very fast responses due to high hydraulic power density and good soft-landing performance with low cost. The EHVA, designed by Zheng Lou, et al. [8], [17] has a built-in soft-seating feature and thus do not require a complicated control strategy for soft-landing. Pradeep Gillella, et al. [35] achieved precise valve motion control of an EHVA using a low-cost feedback control system. Zongxuan Sun, et al. [36] developed a transient control strategy for an electrohydraulic valve actuator to ensure precise valve profile tracking under both constant and varying engine speed. For EHVAs, as well as hydraulic ones, one key challenge is the nonlinear temperature-sensitive characteristics of the hydraulic fluid viscosity. It is critical during the engine cold start [36]. Extensive studies have indicated that this can be compensated by using low lifts at lower temperature, but this problem remains a challenge for electric-hydraulic valve actuation systems.

For the rest of this section, we will concentrate on the modeling and control of the EHVA system developed by Z. Lou, et al. [8], [17] with its system architecture shown below in Figure 5.40.

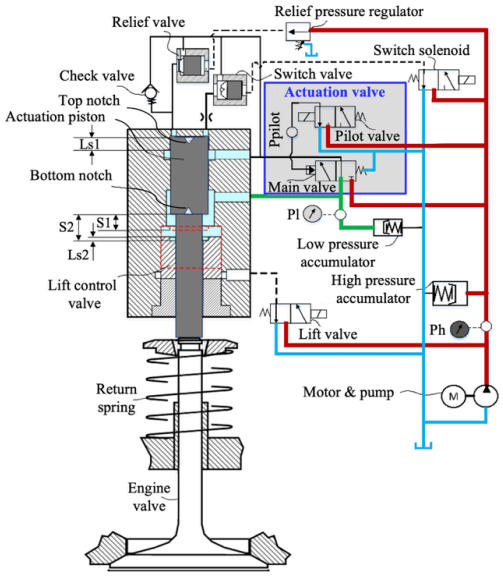


Figure 5.40: EHVA system architecture.

As Figure 5.40 shows, the EHVA system consists of an actuation piston, an actuation cylinder, an actuation valve consisting of

both pilot and main valves, a lift valve, a lift control sleeve, a check valve, a relief valve with a pressure regulator, an orifice switch valve with its control solenoid, and a returning spring (part of the engine valve system).

The actuation cylinder has top, bottom, and lift ports. The dash-dot lines denote the hydraulic fluid paths when hydraulic tank pressure is close to the atmosphere pressure. The bold solid-red lines are connected to the hydraulic fluid source, a high pressure (P_h) accumulator that is pressurized by a motor (or crankshaft) driven pump. The solid-blue lines are linked to the low-pressure fluid source with back pressure (P_l) that is pressurized by the returning fluid from the top port at valve closing and maintained by the low-pressure accumulator through a check valve.

The top port is switched between the low- and high-pressure fluid sources via the actuation valve that consists of a 3-way 2-position hydraulic main valve controlled by a solenoid pilot valve. The serial design is able to achieve response fast enough for engine applications with much lower cost than that of the single stage solenoid valve. The bottom port is connected to the low-pressure fluid source. The actuation piston and engine valve are pushed downward when the top port is pressurized by the high-pressure fluid. This occurs once the actuation valve is activated and returned upward by the returning spring when the top port is switched to the default low pressure fluid.

The lift port is switched between the tank and high-pressure fluid sources via the solenoid-controlled lift valve. The lift control sleeve is used to supply two discrete lifts (S_1 and S_2) corresponding to its two positions (high and low positions). The sleeve moves to its high position once the lift port is pressurized by high pressure (P_h) fluid when the lift control valve is activated. Otherwise, the sleeve remains at the default low position by the low pressure (P_l) fluid applied on the top of the sleeve surface.

Note that there is sufficient force to maintain the sleeve at its high or low position, respectively, because the low pressure (P_l) is designed to be far away from both the high pressure (P_h) and the tank pressure. The two discrete lifts will meet most of the engine operation demands with fast response and accurate lift due to high supply pressure and hard lift constrains, respectively. At the same time, continuous valve lifts can be achieved by accurately controlling the supply and back pressures.

The opening terminal and closing seating velocities are key sources of the valve-train noise and vibrations. High terminal and seating velocities could reduce engine valve reliability and durability. The valve bouncing due to high seating velocity can affect engine performance. The top and bottom snubbers are designed for EHVA to reduce both valve opening-terminal and closing seating velocities. This hydraulic snubber is a fluid-filled cylinder attached to the EHVA body to dampen vertical valve stem oscillations, vibrations, and shocks. The top snubber is used for soft-seating. It consists of a notch on the top edge of the actuation piston, an undercut (Ls_1) inside the top of actuation cylinder, a check valve, and a relief valve with two parallel arranged top orifices.

Note that one of the orifices can be opened and closed by a switch valve. When the actuation piston moves up and past the top undercut into the snubbing zone, the fluid above the piston is forced to escape only through the orifices and the notch. The notch area gets small as the piston goes up. It is designed to provide a smooth transition at the beginning of the snubbing process whereas the orifices are designed to provide more precise control during the seating process.

The soft seating is achieved basically by the top undercut, the notch, and orifices. The switch valve is designed to adjust the flow area of the top orifices to achieve different levels of soft seating under different operational conditions. The relief valve is pressure-sensitive. When the snubbing process is over-pressurized, it provides a fast pressure relief to prevent the valve from bouncing back. The check valve is designed to open at the valve opening for a fast opening. by allowing more fluid flow into the cylinder. It is also designed to close to reduce the flow area to ensure soft seating.

The bottom snubber is designed for a soft landing at the terminal stage of the valve opening. It consists of a notch on the bottom edge of the actuation piston and an undercut with a distance (Ls_2) to the top of the lift control sleeve. The bottom snubber is simpler than the top snubber. This is mainly due to the additional counterforce provided by the returning spring that reduces the opening terminal velocity.

Assume the returning spring force is large enough to keep the valve stem following the actuation piston movement. According to Figure 5.40, the actuation piston is balanced by the top and bottom hydraulic pressures, the system returning spring force and the associated friction force. The actuation piston dynamics can be obtained by applying Newton's second law as below.

\[\left\{

$$\begin{array}{c} m_e\ddot{l} + c_f\dot{l} + k_s\left(l + \delta_s\right) = A_t P_t - A_b P_b, \\ \ddot{l} = \dot{l} = 0, \end{array} \tag{5.5.1}$$

\right. \begin{array}{c} 0

where m_e is the effective mass of total moving parts; l is the engine valve displacement; c_f is the friction coefficient; and k_s and δ_s are the returning spring stiffness and preload, respectively. Note that A_t and A_b are the cross areas of the actuation piston top and bottom, respectively. In addition, P_t and P_b are the actuation piston top and bottom pressures to be derived later in the top and bottom snubber model. Also, S is the engine valve lift switched between the high lift S_1 (7.5 mm) and low lift S_2 (4 mm) through the lift control valve. When the valve lift l approaches the lower or upper limits (i.e. $l = 0$ or S), the bottom or top seat in the actuation cylinder provides additional counterforce. This counterforce balances the hydraulic and spring force difference and stops the valve immediately. The engine valve remains in the limit position ($l = 0$ or S) until the piston top pressure P_t is changed due to activating or deactivating the actuation valve.

The actuation valve has a serial design consisting of a solenoid pilot valve and a hydraulic operated main valve. Once the pilot valve is activated, the high-pressure fluid activates the main valve by pushing the spool open to connect the top port to the high-pressure fluid source. The pilot valve can be modeled as three parts: a spool model, a solenoid model, and an orifice model. The spool model shown in equation ??? describes the spool motion and calculates the associated displacement x_p based on Newton's second law.

\[\left\{

$$\begin{array}{c} m_{ep}\ddot{x}_p + c_{fp}\dot{x}_p + k_{s1}\left(x_p + \delta_{s1}\right) = f_{meg}, \\ \ddot{x}_p = \dot{x}_p = 0, \end{array} \tag{5.5.2}$$

\right. \begin{array}{c} 0

where m_{ep} is the effective mass of moving parts for pilot valve; c_{fp} is the friction coefficient; k_{s1} and δ_{s1} are the stiffness and the preload of the spring, respectively; and x_{p0} is the maximum allowed displacement of the pilot valve spool. When the spool approaches its limit (i.e. $x_p = 0$ or x_{p0}), it stops immediately due to the additional counterforce provided by the pilot valve cylinder. It remains at that position until the solenoid is activated or deactivated.

The solenoid model [24] is used to determine the electromagnetic force f_{meg} by applying Kirchhoff law, as shown in equation 5.5.3 .

$$u - iR - L_i\frac{di}{dt} = 0, F_{meg} = bi^2(1 + x_p/a)^{-1} \tag{5.5.3}$$

where u, i, R and L_i are the solenoid input voltage, current, resistance, and inductance, respectively. Also, a and b are constants used to curve-fit the empirical data provided by the manufacturer.

Consider the orifice flow model with a varying orifice area ($A_{orip} = w_p x_p$) determined by the pilot spool displacement x_p and the calibrated constant coefficient w_p. We can use this model to describe the fluid dynamics through the pilot valve orifice. We can then use it to calculate the delayed outlet pressure $\tilde{P}_{po}$, using the nonlinear orifice flow equation shown in equation 5.5.4.

$$Q_{Vp} = c_d A_{orip}\sqrt{\frac{2\mathrm{sgn}\left(\dot{l}\right)\left[u\left(P_h - P_{atm}\right) + P_{atm} - \tilde{P}_{po}\left(t + t_0\right)\right]}{\rho_{oil}}} = A_{spoolm}\dot{x}_m \tag{5.5.4}$$

where Q_{Vp} and c_d are the volumetric flow rate and discharge coefficient of the fluid across the pilot valve orifice; ρ_{oil} is the density of the fluid; and P_{atm} is the tank pressure, In addition, $\tilde{P}_{po}$ is the delayed P_{po} by time t_0; that is, $\tilde{P}_{po}(t) = P_{po}(t - t_0)$. Furthermore, u is the pilot solenoid on-and-off control input (on: 1 and off: 0). The term $u\left(P_h - P_{atm}\right) + P_{atm}$ represents the pilot valve inlet pressure; P_h will be calculated by the accumulator model in equation 5.5.10. Finally, A_{spoolm} and x_m are the stressed area and displacement of the main valve spool, respectively; and the sign function represents the flow direction.

The delayed pilot outlet pressure $\tilde{P}_{po}$ will then push the main valve open and determine the main valve displacement x_m based on the main valve model. Similarly, this model can be modeled as a spool valve and an orifice model, applying the same principles used in equations ??? and 5.5.4. As a result, the delayed main valve outlet pressure $\tilde{P}_{mo}(t) = P_{mo}(t - t_1)$ can be determined by the pilot valve model.

The top and bottom snubbers are designed for engine valve soft-landing and seating, respectively, by controlling the top and bottom orifice flow areas. We can describe the snubbers as equivalent orifice models. They use the same orifice flow equation as the pilot valve to obtain the actuation piston top pressure P_t and bottom pressure P_b as follows.

$$P_t = \tilde{P}_{mo} - \mathrm{sgn}\left(\dot{l}\right)\left(\frac{A_t\dot{l}}{C_d A_{orit}}\right)^2\frac{\rho_{oil}}{2} \tag{5.5.5}$$

$$P_b = P_l + \text{sgn}\left(\dot{l}\right)\left(\frac{A_b\dot{l}}{C_d A_{orib}}\right)^2 \frac{\rho_{oil}}{2} \tag{5.5.6}$$

where A_t and A_b are the actuation piston top and bottom surface areas, respectively. In addition, A_{orit} is the equivalent top orifice area for the top port orifice, top notch orifice, switch valve orifice, check valve orifice, and relief valve orifice. Therefore, A_{orit} is a function of the valve displacement and the associated control valve state.

From equation 5.5.5, we can see that A_{orit} is a key parameter for the soft seating. Its very small orifice leads to vary large top pressure change, which significantly affects the seating process. Similar to the equivalent top orifice, A_{orib} is the equivalent bottom orifice area for the bottom port and bottom notch orifices; it significantly affects the opening terminal (landing) velocity.

Two hydraulic accumulators are used for temporarily storing pressurized fluids, maintaining pressures, and absorbing pressure pulsations. Note that the accumulator pressures fluctuate cyclically due to the nature of the periodic motion of engine valves. By optimizing the spring-accumulator parameters, the pulsation can be limited to the acceptable level. We can use the linear fluid accumulator model, which describes the accumulator pressure variation due to the net flow rate change, to describe the flow dynamics for both supply and back pressure sources.

$$C_{acl}\dot{P}_l = \left[\frac{1-\text{sgn}\left(\dot{l}\right)}{2}A_t + A_b\right]\dot{l} - Q_{rel} \tag{5.5.7}$$

$$C_{ach}\dot{P}_h = -\frac{1+\text{sgn}\left(\dot{l}\right)}{2}A_t\dot{l} + Q_{pump} - Q_{reh} \tag{5.5.8}$$

where C_{acl} and C_{ach} are the capacitances of the high- and low-pressure accumulators, respectively; Q_{pump} is the pump volumetric flow rate; and Q_{rel} and Q_{reh} are leakage flow rates of the relief valves for high and back pressure accumulators, respectively.

Rearranging equations ??? to 5.5.8, along with the associated two counterpart equations ??? and 5.5.4 for the main valve model, leads to a 9th-order detailed nonlinear model for the EHVA system, shown in Figure 5.40, in equations 5.5.9 and 5.5.10.

$$\begin{cases} m_e\ddot{l} + c_f\dot{l} + k_s\left(l+\delta_s\right) = A_t\left[P_{mo}\left(t-t_d\right) - \dfrac{\rho_{oil}sgn\left(\dot{l}\right)}{2}\left(\dfrac{A_t\dot{l}}{c_d A_{orit}}\right)^2\right] - A_b\left[P_l + \dfrac{\rho_{oil}sgn\left(\dot{l}\right)}{2}\left(\dfrac{A_b\dot{l}}{c_d A_{orib}}\right)^2\right] \\ m_{ep}\ddot{x}_p + c_{fp}\dot{x}_p + k_{s1}\left(x_p+\delta_{s1}\right) = bI^2(1+x_p/a)^{-1} \\ -L_i\dot{i} - Ri + u = 0 \\ P_{po} = u\left(P_h - P_{atm}\right) + P_{atm} - \dfrac{\rho_{oil}sgn\left(\dot{l}\right)}{2}\left(\dfrac{A_{spoolm}\dot{x}_m}{c_d w_p x_p}\right)^2 \\ m_{em}\ddot{x}_m + c_{fm}\dot{x}_m + k_{s2}\left(x_m+\delta_{s2}\right) = P_{po}\left(t-t_0\right)A_{spoolm} \end{cases} \tag{5.5.9}$$

$$\begin{cases} P_{mo} = u\left(P_h - P_l\right) + P_l - \dfrac{\rho_{oil}sgn\left(\dot{l}\right)}{2}\left(\dfrac{A_t\dot{l}}{c_d w_m x_m}\right)^2 \\ C_{acl}\dot{P}_l = \left[\dfrac{1-sgn\left(\dot{l}\right)}{2}A_t + A_b\right]\dot{l} - Q_{rel} \\ C_{ach}\dot{P}_h = -\dfrac{1+sgn\left(\dot{l}\right)}{2}A_t\dot{l} + Q_{pump} - Q_{reh} \end{cases} \tag{5.5.10}$$

where m_{em}, c_{fm}, k_{s2}, δ_{s2}, and w_m are respectively the equivalent moving mass, friction coefficient, spring stiffness, spring preload, and constant orifice area coefficient for the main valve.

As referenced before [17], the EVHA system is highly nonlinear with respect to system operational parameters such as fluid pressure and temperature, as well as the designed valve profile such as the lift-time area, and opening and closing timings. The valve profile is very important to the engine intake and exhaust processes and also engine performance. The nonlinearities are mainly from the orifice equations, the temperature-sensitive viscosity of the hydraulic fluid, and the time-varying actuation valve delay. It is difficult to design a nonlinear controller with guaranteed performance because of the lack of associated nonlinear control theory. Therefore, we conduct model reduction to simplify the original nonlinear model with acceptable modeling error.

We also obtain an LPV (linear parameter-varying) model based on the reduced-order nonlinear model to be used in the future for the LPV controller design.

According to simulation results based upon the nonlinear model described by equations 5.5.9 and 5.5.10, both piston top pressure and valve displacement start rising almost at the same time, as well as the second rise of the pilot pressure. This implies that a very short duration is needed for the main valve to open and for its outlet pressure to reach the threshold that overcomes the preload force of the valve spring. Therefore, we can simplify the actuation valve model (both pilot and main valves) into first-order dynamics with time constant τ_p. We can also use an equivalent time delay t_d to account for the sum of pilot valve delay t_0 and main valve delay t_1, as below.

$$P_{mo}(s) = \frac{P_{mi}(s)}{\tau_p s + 1} = \frac{u(P_h - P_l) + P_l}{\tau_p s + 1} \tag{5.5.11}$$

$$\tilde{P}_{mo}(t) = P_{mo}(t - t_d) \tag{5.5.12}$$

where $P_{mi} = u(P_h - P_l) + P_l$ represents the main valve inlet pressure that is switched between P_h and P_l by the pilot solenoid control signal; and $\tilde{P}_{mo}$ is now the equivalent delayed main valve outlet pressure. Time delay t_d is explicitly modeled in the system instead of approximated by linear transfer functions such as the Fourier–Laguerre series, Padé approximation and shift-based approximations [37], [38] that are widely used in time delay systems.

These approximations could introduce significant modeling error during the start and end of the valve motion due to the pressure-sensitive offshore and landing processes of the actuation piston. Therefore, the time delay t_d will be used explicitly for the system state, and subsequently as a varying parameter directly, resulting in an LPV (linear parameter-varying) time-delayed system to be addressed later.

Note that LPV systems are a class of finite-dimensional linear parameter-varying plants whose system coefficients are affine functions of a set of measurable varying parameters. We can use the LPV modeling and control techniques to deal with nonlinear parameter-varying systems. These systems have received a great deal of attention in automotive applications; they were applied recently to both gasoline [40] and diesel [41] engines. In reference [40], a time-varying friction coefficient is used to account for the variation of the engine oil viscosity for a variable valve timing (VVT) system driven by an electric motor with a planetary gear-train. Engine speed and oil pressure are used as the varying parameters to form an LPV hydraulic VVT system. The LPV control is also used to compensate for the variable time delay presented in the air-fuel ratio control loop of a lean-burn spark-ignition engine [42].

Replacing the actuation valve model (pilot and main valve models) by equations 5.5.11 and 5.5.12, and rearranging equations ???, and 5.5.5 to 5.5.8, we can reduce the original full-order nonlinear model from equations 5.5.9 and 5.5.10 to the fifth-order nonlinear model shown below.

$$\begin{cases} m_e\ddot{l} + c_f\dot{l} + k_s(l + \delta_s) = A_t\left[P_{mo}(t - t_d) - \dfrac{\rho_{oil}\mathrm{sgn}(\dot{l})}{2}\left(\dfrac{A_t\dot{l}}{c_d A_{orit}}\right)^2\right] - A_b\left[P_l + \dfrac{\rho_{oil}\mathrm{sgn}(\dot{l})}{2}\left(\dfrac{A_b\dot{l}}{c_d A_{orib}}\right)^2\right] \\ \tau_p\dot{P}_{mo} = -P_{mo} + u(P_h - P_l) + P_l \\ C_{acl}\dot{P}_l = \left[\dfrac{1 - \mathrm{sgn}(\dot{l})}{2}A_t + A_b\right]\dot{l} - Q_{rel} \\ C_{ach}\dot{P}_h = -\dfrac{1 + \mathrm{sgn}(\dot{l})}{2}A_t\dot{l} + Q_{pump} - Q_{reh} \end{cases} \tag{5.5.13}$$

By analyzing both full- and reduced-order models and comparing the model simulation results with the experimental ones, it is clear that the system time delay $t_d(P_h, T)$ and time constant $\tau_p(P_h, T)$ for the piston top pressure vary with the supply fluid pressure and temperature. The discharge coefficient, as a function of fluid viscosity, is mainly influenced by the fluid temperature. Generally, the local discharge coefficient is also affected by the top and bottom equivalent orifice areas, $A_{orit}(l)$ and $A_{orib}(l)$. They are functions of the valve displacement. However, the accurate relationship between the orifice flow area and the discharge coefficient is usually difficult to obtain. As a result, it often relies on the empirical equation for a given orifice type [39]. Therefore, an area-irrelevant mean-effective discharge coefficient $c_d(T)$ is used to account for the viscosity as a function of temperature in the EHVA model. The high- and low-pressure sources P_h and P_l are assumed to be measurable varying parameters.

As a summary, for the LPV model, the last two accumulator equations in equation 5.5.13 can be eliminated, resulting in a third-

order time-delay system. Therefore, in the nonlinear model 5.5.13, $t_d(P_h,T)$, $\tau_p(P_h,T)$, $c_d(T)$, $A_{orit}(l)$, $A_{orib}(l)$, P_h and P_l are varying parameters. Defining the state vector $x=\left[\begin{array}{ccc} x_1 & x_2 & x_3 \end{array}\right]^T=\left[\begin{array}{ccc} P_{mo} & l & \dot{l} \end{array}\right]^T$ and letting the system output equation be $y=x_2=l$, the resulting third-order nonlinear model is shown below.

$$\begin{cases} \dot{x}_1(t)=-\dfrac{1}{\tau_p(P_h,T)}x_1(t)+\dfrac{1}{\tau_p(P_h,T)}\left[u(t)(P_h-P_l)+P_l\right] \\ \dot{x}_2(t)=x_3(t) \\ \dot{x}_3(t)=\dfrac{A_t}{m_e}x_1(t-t_d)-\dfrac{k_s}{m_e}x_2(t)-\dfrac{\rho_{oil}\mathrm{sgn}(x_3)}{2m_e c_d(T)^2}\left(\dfrac{A_t^3}{A_{orit}^2(x_2)}+\dfrac{A_b^3}{A_{orib}^2(x_2)}\right)x_3(t)^2 \\ \quad -\dfrac{c_f}{m_e}x_3(t)-\dfrac{A_bP_l+k_s\delta_s}{m_e} \\ y(t)=x_2(t) \end{cases} \tag{5.5.14}$$

Although the time delay $t_d(P_h,T)$ is treated as a varying parameter, it is feasible to design LPV controllers for the EHVA system with time delay in control input and parameters [43] for the convenience of online implementation. Therefore, we apply an equivalent transformation to the nonlinear model 5.5.14. Because x_1 is only a function of the input and varying parameters, the first equation in model 5.5.14 can be replaced by a dynamic equation with the delayed state $\tilde{x}_1(t)=x_1(t-t_d(P_h,T))$. We do this by substituting the input and varying-parameters with their delayed counterparts.

$$\dot{\tilde{x}}_1(t)=-\frac{1}{\tau_{p\left(\tilde{P}_h,T\right)}}\tilde{x}_1(t)+\frac{1}{\tau_{p\left(\tilde{P}_h,T\right)}}\left[\tilde{u}(t)\left(\tilde{P}_h-\tilde{P}_l\right)+\tilde{P}_l\right] \tag{5.5.15}$$

where $\tilde{u}(t)=u\left(t-t_d\left(\tilde{P}_h,T\right)\right)$ is the delayed input; and $\tilde{P}_h(t)=P_h\left(t-t_d\left(\tilde{P}_h,T\right)\right)$ and $\tilde{P}_l(t)=P_l\left(t-t_d\left(\tilde{P}_h,T\right)\right)$ are the delayed varying parameters. The fluid temperature $T(t)$ changes much slower than any other parameters; therefore, that it does not need to be delayed.

Substituting equation 5.5.15 into the dynamic equation for x_1 in equation 5.5.14, and replacing $x_1(t-t_d(P_h,T))$ with $\tilde{x}_1(t)$, we can convert the time-delayed system into an input- and parameter-delayed system. We can easily handle this system by directly using the delayed input and parameters in the control system. Meanwhile, to convert the nonlinear model into the LPV model format, we require additional transformation and parameter merging to deal with the nonlinearities in the model.

First, define the new input, states, and output as $\hat{u}=\left[\begin{array}{cc} u_1 & u_2 \end{array}\right]^T=\left[\begin{array}{cc} \tilde{u} & 1-\tilde{u} \end{array}\right]^T$; $\hat{x}=\left[\begin{array}{ccc} \hat{x}_1 & \hat{x}_2 & \hat{x}_3 \end{array}\right]^T=\left[\begin{array}{ccc} x_1 & x_2+\delta_s & x_3 \end{array}\right]^T$; and $\hat{y}=\hat{x}_2$, respectively. Second, in the dynamic equation of x_3 in equation 5.5.14, merge one x_3 in the quadratic term and the sign function into the coefficient; we can then obtain a lumped varying-parameter Λ. We then have a function of the top and bottom equivalent orifice areas, the discharge coefficient, and the engine valve velocity. Now we convert the nonlinear model into an LPV time-delay system model below.

$$\dot{\hat{x}}(t)=A(t)\hat{x}(t)+B(t)\hat{u}(t),\hat{y}(t)=C(t)\hat{x}(t) \tag{5.5.16}$$

where

\[A=\left[

$$\begin{array}{ccc} -\dfrac{1}{\tau_{p\left(\tilde{P}_h,T\right)}} & 0 & 0 \\ 0 & 0 & 1 \\ \dfrac{A_t}{m_e} & -\dfrac{k_s}{m_e} & \Lambda \end{array} \tag{5.5.17}$$

\right], B=\left[\begin{array}{cc} \dfrac

ParseError: invalid DekiScript (click for details)

```
Callstack:
    at (Bookshelves/Introductory_Engineering/Mechatronics:_Fundamentals_Design_Integration_and
```

& \dfrac
ParseError: invalid DekiScript (click for details)

```
Callstack:
    at (Bookshelves/Introductory_Engineering/Mechatronics:_Fundamentals_Design_Integration_and
```

\ 0 & 0 \ \dfrac{A_bP_l}{m_e} & \dfrac{A_bP_l}{m_e} \end{array} \right], C=\left[

$$0 \quad 1 \quad 0 \tag{5.5.18}$$

\right] \label{5.60}\]

and

$$\Lambda = -\frac{c_f}{m_e} - \frac{\rho_{oil}\text{sgn}\left(\hat{x}_3\right)\hat{x}_3}{2m_e c_d\left(T\right)^2}\left(\frac{A_t^3}{A_{orit}^2\left(\hat{x}_2\right)} + \frac{A_b^3}{A_{orib}^2\left(\hat{x}_2\right)}\right) \tag{5.5.19}$$

Define the scheduling parameters as follows.

$$\theta = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{bmatrix}^T = \begin{bmatrix} \frac{1}{\tau_{p\left(\tilde{P}_h,T\right)}} & \Lambda & \frac{\tilde{P}_h}{\tau_{p\left(\tilde{P}_h,T\right)}} & \frac{\tilde{P}_l}{\tau_{p\left(\tilde{P}_h,T\right)}} & P_l \end{bmatrix}^T \tag{5.5.20}$$

We can write the system and input matrices of the LPV system model into the following affine parameter-dependent form suitable for future LPV control design [44].

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ \frac{A_t}{m_e} & -\frac{k_s}{m_e} & 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\theta_1 + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\theta_2 \tag{5.5.21}$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\theta_3 + \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\theta_4 + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{A_b}{m_e} & \frac{A_b}{m_e} \end{bmatrix}\theta_5 \tag{5.5.22}$$

Among the scheduling parameters, we can obtain θ_1, θ_3, θ_4, and θ_5 directly or indirectly based on the measurable parameters under different operational conditions. By contrast, θ_2 can be constructed using the measurable state $\hat{x}_2$ and parameter T.

This section presents a process of modeling an EHVA system in the linear parameter-varying (LPV) framework with a process of developing a full nonlinear model, simplifying it, and constructing the LPV model. Detailed control design for the EHVA system can be found in reference [6] for LQT (linear quadratic tracking) control and reference [45] for adaptive control based on the developed LPV model shown in equation 5.5.16.

Section Summary

In this section, electrohydraulic valve systems (EVHA) [5], [6], [7], [8] are introduced, where the internal combustion engine electrohydraulic intake and exhaust valve systems are used as example. The electrohydraulic valve system requires an expensive high-pressure hydraulic fluid source. Electromagnetic-controlled hydraulic valves are used to control the displacement (or speed). Instead of generating the desired torque or force, the electrohydraulic actuator generates a proper linear (or rotational) displacement to tune on or off (or move the valve linearly) a hydraulic valve in order to achieve the desired valve displacement. Similarly, a closed-loop control system is often required to regulate the valve displacement response accurately.

Section Keywords

Practice Problems

1. Consider the state-space LPV model ??? with system matrix A described by equation 5.5.21 and control input matrix B defined by equation 5.5.22. Justify whether system matrix A described by equation 5.5.21 and control input matrix B defined by equation 5.5.22 are affine or linear.

2. Develop a MATLAB/Simulink model for the LPV EHVA system shown in equations 5.5.16 to 5.5.22.

This page titled 5.5: Discrete Electrohydraulic Valve Actuation (EHVA) is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control

Learning Objectives

After completing this section, you will be able to:

1. Explain the working principle of a permanent magnetic synchronous motor (PMSM).
2. Describe the vector control of PMSM along with the rotational d-q coordinate, the fixed stator (α, β) coordinate, and the stator $(a,\ b,\ c)$ coordinate, and associated coordinate transformations.
3. Provide an explanation of the PMSM PWM (pulse-width modulated) drive circuit.

A synchronous electric motor is an AC (alternative current) electric motor; its rotational shaft is synchronized with the frequency of the supply current at the steady state. In addition, its rotation period equals an integer number of AC cycles. Synchronous motors use so-called electromagnets. These are energized by applying voltage to the stator windings to create a rotating magnetic field synchronized in time with the oscillations of the current. The electric machine rotor with permanent magnets or electromagnets turns in step with the stator field at the same rate. As a result, it provides the second synchronized rotating magnet field.

Synchronous and induction motors are widely used AC motors. Because they do not rely on induction to produce the rotor's magnetic field, synchronous motors rotate at a rate locked to the line frequency. By contrast, induction motors require slip between the rotor rotating speed and line frequency. This is because the rotor must rotate at a frequency slightly slower than the AC alternations in order to induce current in the rotor. Small synchronous motors are often used in timing devices such as in synchronous clocks, timers, tape recorders, and precision servomechanisms that must operate at a precise speed. In these cases, the speed accuracy is dependent on the supply power line frequency controlled in interconnected grid systems or by DC-to-AC inverters.

In fact, synchronous motors also include synchronous generators when their rotors are driven by the external torque. The two major types of synchronous motors are distinguished by how the rotor is magnetized: non-excited and direct-current excited. In non-excited motors, the rotor is made of steel. It rotates in step with the stator's rotating magnetic field, so it has an almost-constant magnetic field through it. The external stator field magnetizes the rotor, inducing the magnetic poles needed to turn it. The rotor is often made of a highly retentive steel such as cobalt steel.

Permanent-magnet synchronous motors (PMSMs) use permanent magnets embedded in the rotor to create a constant magnetic field. The stator carries windings connected to an AC electricity supply in order to produce a rotating magnetic field (as in asynchronous motors). At synchronous speed, the rotor poles lock to the rotating magnetic field. Note that PMSMs are similar to brushless DC motors, where Neodymium magnets are widely used with possible ferrite magnets. Machines that use ferrite magnets have lower power and torque densities when compared with neodymium machines [46].

Almost all PMSMs use three-phase alternating current to set up a rotating magnetic field (RMF). In these cases, the frequency and power are controlled by the power electronics (or DC-AC converters) that respond to the command speed/acceleration. In a PMSM, stators contain numerous parallel slots stuffed with interconnected loops of copper windings. These can be bulky loops of round copper wire or tidy hairpin-shaped copper insertions with square cross-sections that increase both fill density and direct wire-to-wire contact within the grooves. Denser windings improve torque capability, and tidier interlacing at the ends amounts to less bulk and a smaller overall package.

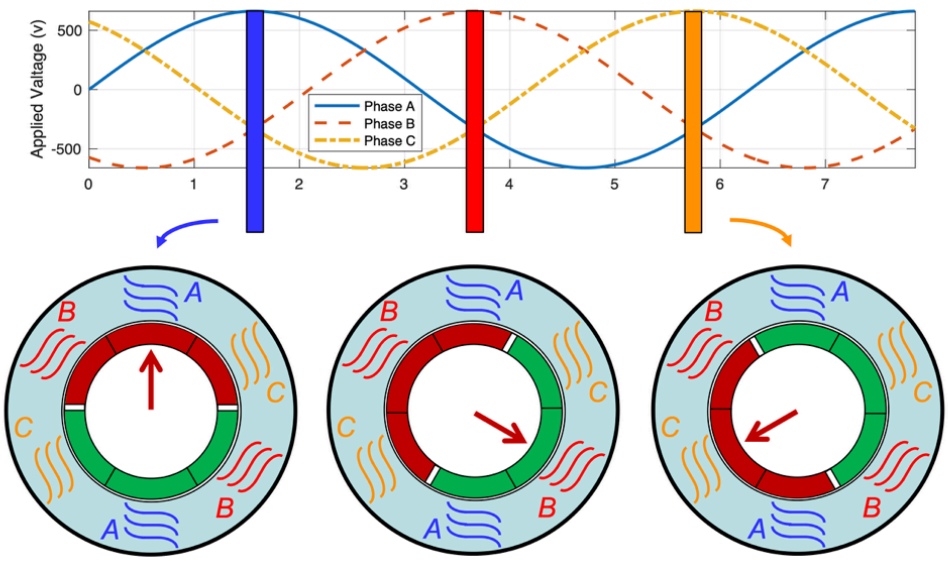


Figure 5.41: Rotational magnetic field and corresponding rotor position.

Figure 5.41 shows the working principle for a three-phase controlled two-pole PMSM. In the top plot, the three-phase voltages (Phases A, B, and C) are applied to the three windings of the PMSM. As a result, a rotation magnetic field (RMF) is formed because the three-phase power has a 120-degree phase apart. The applied three-phase voltages and the corresponding windings are color coded in Figure 5.41. The bottom plot shows the magnetic field direction (red arrow at the center) and the corresponding rotor position with two half-circle strips representing the permanent magnets. As the magnetic field rotates in a clockwise direction due to applied three-phase voltages, the rotor rotates in the same direction (clockwise) and is synchronized with the rotating magnetic field. This confirms the synchronous operation of the motor.

PMSMs are typically controlled using so-called direct torque control (DTC) [12] or field-oriented control (FOC) [47]. In this section we will mainly address the field-oriented control (FOC). The field-oriented control, also called vector control, is a variable-frequency drive (VFD) control method in which the stator currents of a three-phase AC (or brushless DC) electric motor are identified as two orthogonal components. These components can be visualized with a vector; one component is defined as the motor magnetic flux and the other as the torque. The control system of the motor drive calculates the corresponding current component references from the flux as well as the torque references provided by the speed control command. Typically, proportional-integral (PI) controllers (Section 3.4) are used to keep the measured current components at their reference values. The PWM (pulse-width modulation) of the variable-frequency drive defines the switch-commands for these transistors (switches), according to the desired stator voltage references.

The electrical PMSM model in the d-q reference frame can be expressed by equations 5.6.1 and 5.6.2.

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = \begin{bmatrix} R_s + L_d\mathcal{D} & -\omega L_q \\ \omega L_d & R_s + L_q\mathcal{D} \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} 0 \\ \omega\psi_f \end{bmatrix} \tag{5.6.1}$$

$$\tau_{em} = n_p\psi_f i_q \tag{5.6.2}$$

where u_d and u_q are the d- and q-axis terminal voltages, respectively; i_d and i_q are the d- and q-axis armature currents, respectively; and L_d and L_q are the d- and q-axis inductance, respectively. In addition, R_s is the armature resistance; ω is the electrical angular velocity (or motor angular speed); ψ_f is the permanent magnet flux-linkage; n_p is the number of pole pairs of the PMSM; and $\mathcal{D}$ is a differential operator such that the following is true.

$$\mathcal{D}\left[\cdot\right] = \frac{d\left[\cdot\right]}{dt}, \ e.g., \mathcal{D}i_q = \frac{di_q}{dt} \tag{5.6.3}$$

Inductors L_d and L_q are equal for non-salient motors. As indicated by equation 5.6.2, the motor torque is determined only by i_q and has nothing to do with i_d. In order to get better motor efficiency, i_d is controlled to be zero during operation, which reduces the copper (energy) loss [46].

Figure 5.42 shows a simplified PMSM control circuit (or an inverter circuit converting DC power to AC). The circuit uses six

MOSFETs (metal-oxide-semiconductor field-effect transistors). Note that these MOSFETs are controlled by PWM signals, where the lower PWM signal (e.g., PWM_{Q_2}) has an opposite level of voltage (see PWM_{Q_1} in Figure 5.42). Three stator windings are connected together at one end. The other three are controlled by the inverter. A power MOSFET is a voltage-controlled transistor designed for high-frequency operations. It has a very small load voltage drop when it is turned on. However, the saturation temperature sensitivity limits its application in high power applications. Thus, the IGBT (insulated-gate bipolar transistor) should be used for high power applications. However, IGBT has a high voltage drop due to utilizing of bipolar transistor, leading to high conduction loss.

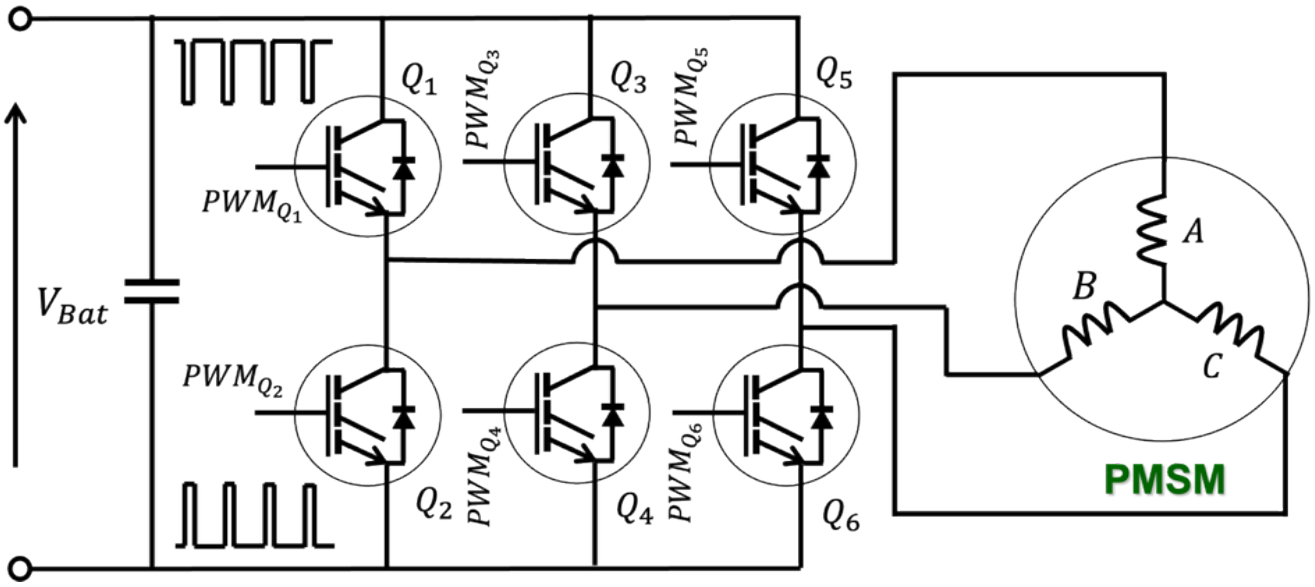


Figure 5.42: A simplified three-phase inverter circuit.

Vector control (VC) is an elegant control method for PMSMs, where field-oriented control (FOC) theory is used to control space vectors of magnetic flux, current, and voltage. It is possible to set up the coordinate system to decompose the vectors into a magnetic field-generating part (q) and a torque-generating part (d). The structure of the motor controller (vector control controller) is then almost the same as for a separately excited DC motor, which simplifies the control of PMSM. The VC technique was developed specifically to achieve a similarly dynamic performance in PMSM. Figure 5.43 presents a control system block diagram that is widely used for speed and torque control with current ($i_d,\ i_q$) feedback.

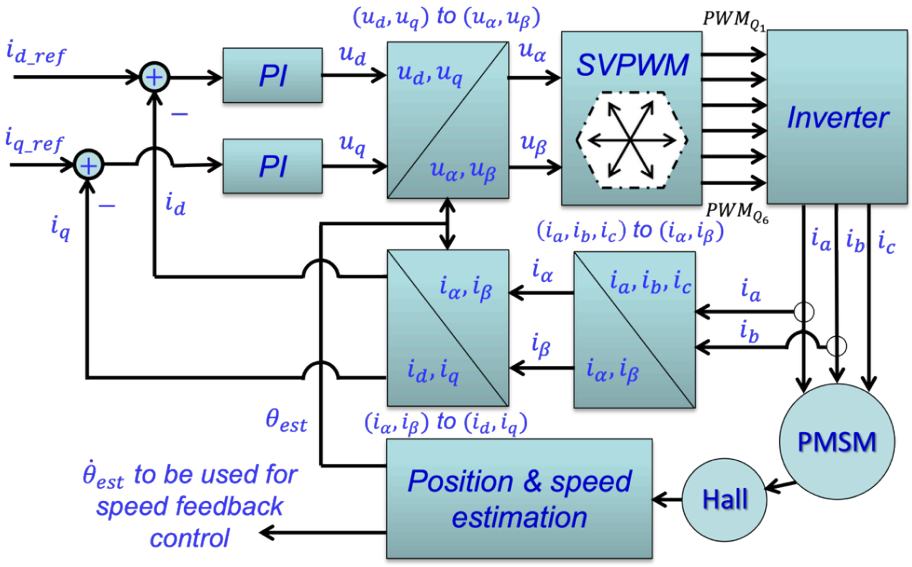


Figure 5.43: A FOC (field-oriented control) diagram for PMSM.

Figure 5.43 shows the basic structure of vector control of the PMSM. First, there are two feedback blocks. One of these, labeled *Hall*, is the hall-effect sensor (see Section 6.2 for details of its working principle) used for motor-axle position and speed

measurements. The other, is for stator current measurements. Denoted by the circles drawn on the stator current wires, it is used for feedback control of $(i_d,\ i_q)$ for torque and flux regulation. The pulse train signals from these hall-effect encoders are to be processed in the block labeled *Position & speed estimation*. Here, the signals are converted to the motor rotor position and speed signals. Section 6.2 will note that the hall-effect sensor outputs are two square-pulse trains with their period inversely proportional to the rotational speed $\dot{\theta}_{est}$. The rotor position θ_{est} can be obtained by counting the signal pulse train with known initial position (or using multichannel hall-effect pulse trains).

Next, we have the block named (i_a, i_b, i_c) *to* $(i_\alpha,\ i_\beta)$. Based upon the Kirchhoff current law (summation of currents entering a junction equals to that of leaving it), we have the following.

$$i_a + i_b + i_c = 0,\ e.g., i_c = -i_a - i_b \tag{5.6.4}$$

As a result, even though only stator currents i_a and i_b are measured, state current i_c can be calculated based on equation 5.6.4. Now, we need to convert the measured currents i_a, i_b, and i_c into the two-phase current space denoted by the $(\alpha,\ \beta)$ coordinate. This transformation is also called the Clark transformation [48]. The Clark transformation utilizes the balanced three-phase components (i_a, i_b, i_c) in the $(a,\ b,\ c)$ reference frame and outputs the balanced two-phase orthogonal components (i_α, i_β) in the stationary $(\alpha,\ \beta)$ reference frame, demonstrated in Figure 5.44.

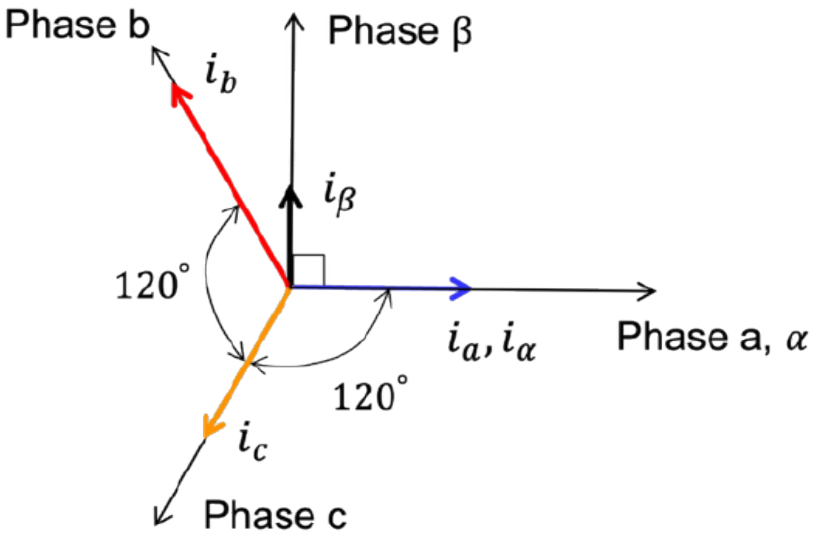


Figure 5.44: Clark transformation coordinate relationship.

From Figure 5.44, the coordinate transformation from (i_a, i_b, i_c) to (i_α, i_β) reference frame can be completed utilizing the following equation.

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = k \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{5.6.5}$$

Using equation 5.6.4 and condition $i_\alpha = i_a$, equation 5.6.5 can be simplified as below.

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = T_{clark} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{5.6.6}$$

Returning to Figure 5.43, after the current signals i_a, i_b, i_c are transferred to (i_α, i_β) coordinate, we next use the block labeled (i_α, i_β) *to* (i_d, i_q) to transfer the current signals i_α, i_β in the stationary coordinate to the d-q coordinate in order to obtain i_d, i_q. This transformation is also called the Park transformation [48]. Note that vector control is performed entirely in the d-q coordinate system to simplify the control of PMSMs.

Figure 5.45 shows the coordinate relationship between (i_α, i_β) and (i_d, i_q) based upon the Park transformation shown below.

Extra close brace or missing open brace

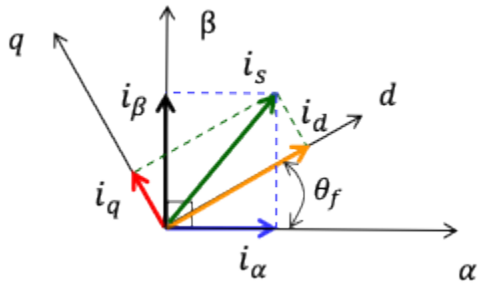


Figure 5.45: Park transformation coordinate relationship.

where θ_f is the rotational field angle and i_s is the rotational current vector. Because we are dealing with the synchronous motor, θ_f is equal to the rotor position that can be measured and estimated using the hall-effect sensor signals. In practical applications, θ_{est} will be used in place of θ_f in Figure 5.43.

Next, from Figure 5.43, the given reference current signals i_{d_ref} and i_{q_ref} are subtracted by the estimated signals i_d and i_q, respectively. The resulting error signals are used as the input to the two PI controllers (see Section 3.4) to generate the two control voltage signals u_d and u_q, respectively.

The control magnitudes u_d and u_q are in the rotational d-q coordinate. Therefore, they need to be transferred back to the static $(\alpha,\ \beta)$ coordinate; see the (u_d, u_q) *to* (u_α, u_β) block in Figure 5.43. This transfer can be achieved by reversing the process described in equation ??? as follows.

Extra close brace or missing open brace

The block in Figure 5.43 labeled SVPWM (space vector pulse width modulation) generates control signals for these switches on the DC-AC inverter (see an example in Figure 5.42). Thus, precise output voltage and current waveforms can be generated. The purpose of the SVPWM is to convert the static voltages u_α, u_β to the $(a,\ b,\ c)$ coordinate using the inverse Clark transformation T^+_{clark} and then follow the process described in references [49] and [50], where $[\cdot]^+$ is the pseudo inverse of matrix $[\cdot]$. As a result, the pseudo inverse T^+_{clark} can be calculated using the singular value decomposition (SVD) as below.

$$T_{clark} = U \cdot S \cdot V^* = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix}, \Sigma > 0 \tag{5.6.7}$$

where "*" represents the complex conjugate transpose. Based on SVD results, the pseudo inverse of T_{clark} can be calculated as below.

$$T^+_{clark} = V_1 \cdot \Sigma^{-1} \cdot U_1^*, \Sigma > 0 \tag{5.6.8}$$

With T^+_{clark} defined in equation 5.6.8, the desired voltage signals u_a, u_b, and u_c in $(a,\ b,\ c)$ coordinate can be calculated as follows.

$$\begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} = T^+_{clark} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} \tag{5.6.9}$$

The PWM control signals PWM_{Q1} to PWM_{Q6} (see Figure 5.43 for detail) can be generated by a certain kind of DSP (digital signal processor). Its mathematical principle can be found in reference [51].

Last, the *Inverter* in Figure 5.43 is a device converting the PWM command from the SVPWM block to generate motor control voltage signals. This conversion results in the motor current i_a, i_b, and i_c, where a sample inverter circuit can be found in Figure 5.42.

As a summary, a permanent magnet synchronous motor (PMSM) can be considered similar to a brushless direct current (BLDC) motor with a permanent magnet rotor and wound stators. For example, a three-phase stator layout is designed to create sinusoidal flux distribution in the air gap; it makes the back electromotive force (BEMF) sinusoidal with motor rotation period equal to an integer number of the current oscillations frequency.

Section Summary

This section discusses permanent magnet synchronous motor (PMSM) actuators. These types of motors are widely used in the powertrain of electric vehicles and have attracted intensive research and development in the automotive and other industries. A permanent magnet synchronous electric motor [9], [10], is an AC electric motor. Its steady-state shaft rotation speed is synchronized with the control current frequency. The motor rotor with permanent magnets or electromagnets turns in step with the stator field at the same rate. As a result, it provides the second synchronized rotating magnet field. This section also reviews drive hardware, along with vector control of PMSMs [11], [12].

Section Keywords

DC-AC converter: This electronic device converts direct current (DC) power into alternating current (AC) power. It uses electronic switches to rapidly switch the DC power on and off, reversing the polarity to create an AC waveform.

Direct torque control: This advanced AC motor control technique directly regulates the motor's electromagnetic torque and magnetic flux, rather than indirectly controlling motor currents.

Field-oriented control: This method for controlling AC motors mathematically separates the motor stator current into two orthogonal components: one for magnetic flux (d-axis) and one for torque (q-axis).

Induction motor: This electric motor uses electromagnetic induction to create rotation, making it the most common type of electric motor used in industrial and household applications.

Permanent-magnet synchronous motor (PMSM): This motor uses permanent magnets embedded in the rotor to create a constant magnetic field. The stator carries windings connected to an AC electricity supply to produce a rotating magnetic field (as in an asynchronous motor). Note that the AC supply can also be created using a power inverter with DC power supply.

Practice Problems

1. Label the following statements as "true" or "false". If the statement is false, provide the correct statement.
    1. The speed of a PMSM is equal to its stator current frequency.
    2. The speed of an induction motor is greater than its stator current frequency.
2. Calculate the pseudo inverse of T_{clark} defined in equation 5.6.6.

This page titled 5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

5.7: Chapter References

1. R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 12th ed., Englewood Cliffs, NJ, USA: Prentice Hall, 2011, ISBN-13: 978-0-1360-2458-0.
2. W. S. Chang, T. A. Parlikar, M. D. Seeman, D. J. Perreault, J. G. Kassakian, and T. A. Keim, "A new electromagnetic valve actuator," *Power Electron. in Transp.*, Auburn Hills, MI, pp. 109–118, Oct. 24–25, 2002, doi: 10.1109/PET.2002.1185558.
3. K. S. Peterson, A. G. Stefanopoulou, and J. Freudenberg, "Current versus flux in the control of electromechanical valve actuators," in *Proc. 2005 Amer. Control Conf.*, Portland, OR, USA, Jun. 8–10, 2005, pp. 5021–5026.
4. I. Haskara, V. V. Kokotovic, and L. A. Mianzo, "Control of an electromechanical valve actuator for a camless engine," *Int. J. of Robust and Nonlinear Control*, vol. 14, no. 6, pp. 561–579, Feb. 2004.
5. D. Lou and G. Zhu, "Review of advancement in variable valve actuation of internal combustion engines," *Appl. Sci.: Mech. Eng.*, vol. 10, p. 1216, Feb. 2020, doi: 10.3390/app10041216.
6. H. Li, Y. Huang, G. Zhu, and Z. Lou, "Adaptive LQT valve timing control for an electrohydraulic variable valve actuator," *IEEE Trans. on Control Syst. Tech.,* vol. 27, no. 5, pp. 2182–2194, Sep. 2019, doi: 10.1109/TCST.2018.2861865.
7. H. Li, Y. Huang, G. Zhu, and Z. Lou, "LPV model of an electrohydraulic variable valve actuator for internal combustion engines," *ASME J. of Dyn. Syst., Meas., and Control,* vol. 140, no. 1, p. 011005, Jan. 2018, doi: 10.1115/1.4037286.
8. Z. Lou, Q. Deng, S. Wen, Y. Zhang, M. Yu, M. Sun, and G. Zhu, "Progress in camless variable valve actuation with two-spring pendulum and electrohydraulic latching," *SAE Int. J. of Engines, SAE 2013-01-0590,* vol. 6, no. 1, 2013, doi:10.4271/2013-01-0590.
9. A. E. Fitzgerald, C. Kingsley Jr., and A. Kusko, "Synchronous machines, steady state," in *Electric Machinery*, 3rd ed., Columbus, OH, USA: McGraw-Hill, 1972, ch. 6, pp. 283–330.
10. P. C-K. Luk, H. A. Abdulrahem, and B. Xia, "Low-cost high performance ferrite permanent magnet machines in EV applications: a comprehensive review," *eTransportation,* vol. 6, *p. 100080,* Nov. 2020, doi: 10.1016/j.etran.2020.100080.
11. Z. Wang, J. Chen, M. Ming, and K. T. Chau, "Field-oriented control and direct torque control for paralleled VSIs Fed PMSM drives with variable switching frequencies," *IEEE Trans. on Power Electron.*, vol. 31, no. 3, pp. 2417–2428, Sep. 2020, doi: 10.1109/TPEL.2015.2437893.
12. K. Suman, K, Suneeta, and M. Sasikala, "Direct torque-controlled induction motor drive with space vector modulation fed with three-level inverter," in *Proc. 2012 IEEE Int. Conf. of Power Electron., Drives Energy Syst. (PEDES),* Dec. 16–19, 2012, pp. 1–6, doi: 10.1109/PEDES.2012.6484405.
13. C. W. de Silva, *Mechatronics: An Integrated Approach*, Boca Raton, FL, USA: CRC Press, 2004, p. 675, ISBN-13: 978-0-8493-1274-8.
14. T. Arefin and H. Kabir, "Simple discussion on stepper motors for the development of electric device," *Int. J. of Sci. & Eng. Res.*, vol. 5, no. 1, pp. 1089–1096, Jan. 2014.
15. J. F. Gieras, Z. J. Piech, and B. Tomczuk, *Linear Synchronous Motors: Transportation and Automation Systems*, Boca Raton, FL, USA: CRC Press, 2011, ISBN-13: 978-1-4398-4221-8.
16. D. J. Patterson, C. W. Brice, R. A. Dougal, and D Kovuri, "The 'goodness' of small contemporary permanent magnet electric machines," in *Proc. Int. of Electric Machines and Drives Conf. (IEMDC)*, Madison, WI, USA, Jun. 2003, vol. 2, pp. 1195–1200, doi: 10.1109/IEMDC.2003.1210392.
17. Z. D. Lou, S. Wen, J. Qian, H. Xu, et al., "Camless variable valve actuator with two discrete lifts," *SAE Technical Paper 2015-01-0324*, 2015.
18. F. Balduzzi, G. Vichi, L. Romani, G. Ferrara, et al., "Development of a low-pressure direct injection system for a small 2S engine. Part I — CFD analysis of the injection process," *SAE Technical Paper 2015-01-1727*, 2015.
19. C. W. Misner, K. S. Thorne, and J. A. Wheeler, *Gravitation,* Princeton, NJ, USA: Princeton Univ. Press, 1973, pp.72–73, ISBN: 978-0-6911-7779-3.
20. I. S. Grant and W. R. Phillips, *Electromagnetism*, 2nd ed., Hoboken, NJ, USA: John Wiley & Sons, Oct. 1990, pp. 122, ISBN-13: 978-0-4719-2712-9.
21. A. Waygood, *An Introduction to Electrical Science,* Oxfordshire, UK: Taylor & Francis, 2013, ISBN-13: 978-0-4158-1002-9.
22. E. Hossain, *MATLAB and Simulink Crash Course for Engineers*, Switzerland: Springer, 2022, ISBN-13: 978-3-0308-9761-1.
23. K. S. Peterson, A. G. Stefanopoulou, and J. Freudenberg, "Current versus flux in the control of electromechanical valve actuators," in *Proc. 2005 Amer. Control Conf.*, Portland, OR, USA, Jun. 8-10, 2005.
24. J. Ma, G. Zhu, and H. Schock, "A dynamic model of an electropneumatic valve actuator for internal combustion engines," *ASME J. of Dyn. Syst., Meas., and Control*, vol. 132, no. 2, p. 021007, Mar. 2010, doi: 10.1115/1.4000816.
25. Y. Wang, A. Stefanopoulou, M. Haghgooie, I. Kolmanovsky, and M. Hammoud, "Modeling of an electromechanical valve actuator for a camless engine," in *Proc. 2000 (5th) Int. Symp. of Adv. Veh. Control*, no. 93, Ann Arbor, MI, Aug. 22–24, 2000.
26. C. Tai and T. Tsao, "Control of an electromechanical camless valve actuator," in *Proc. 2002 Amer. Control Conf.*, Anchorage,

AK, USA, May 8–10, 2002.
27. Y. Wang, T. Megli, M. Haghgooie, K. Peterson, and A. Stefanopoulou, “Modeling and control of electromechanical valve actuator,” *SAE 2002 Trans. J. of Engines, SAE 2002-01-1106*, 2002.
28. S. Butzmann, J. Melbert, and A. Kock, “Sensor-less control of electromagnetic actuators for variable valve train,” *SAE Technical Paper 2000-01-1225,* 2000.
29. M. Montanari, F. Ronchi, C. Rossi, and A. Tonielli, “Control of a camless engine electromechanical actuator: Position reconstruction and dynamic performance analysis,” *IEEE Trans. on Ind. Electron.*, vol. 51, no. 2, pp. 299–311, Apr. 2004.
30. K. Peterson, A. Stefanopoulou, Y. Wang, and T. Megli, “Output observer-based feedback for soft landing of electromechanical camless valvetrain actuator,” in *Proc. 2002 Amer. Control Conf.*, Anchorage, AK, USA, May 8–10, 2002, pp. 1413–1418.
31. K. Peterson and A. Stefanopoulou, “Rendering the electromechanical valve actuator globally asymptotically stable,” in *Proc. 2003 IEEE Conf. on Decis. Control*, Maui, HI, USA, Dec. 2003, pp. 1753–1758.
32. L. Mianzo, “A unified Hamiltonian framework for $\mathcal{H}_2$ and $\mathcal{H}_\infty$ preview control algorithms with application to a variable valve timing engine,” *Ph.D. dissertation*, University of Michigan, MI, USA, 2002.
33. Z. Lou, “Camless variable valve actuation designs with two-spring pendulum and electrohydraulic latching,” *SAE Tech. Paper 2007-01-1295,* 2007.
34. J. P. Watson and R. J. Wakeman, “Simulation of a pneumatic valve actuation system for internal combustion engine,” *SAE Technical Paper 2005-01- 0771,* 2005.
35. P. Gillella and Z. Sun, "Design, modeling, and control of a camless valve actuation system with internal feedback," *IEEE/ASME Trans. on Mechatron.*, vol. 16, no. 3, pp. 527–539, July 2011.
36. Z. Sun and T. W. Kuo, “Transient control of electrohydraulic fully flexible engine valve actuation system,” *IEEE Trans. on Control Syst. Tech.*, vol. 18, no. 3, pp. 613–621, May 2010.
37. R. Zope, J. Mohammadpour, K. Grigoriadis, and M. Franchek, “Robust fueling strategy for an SI engine modeled as a linear parameter-varying time-delayed system,” in *Proc. 2010 Amer. Control Conf.*, Baltimore, MD, USA, Jun. 30–Jul. 2, 2010.
38. J. Partington, “Some frequency-domain approaches to the model reduction of delay systems,” *Annu. Rev. of Control*, vol. 28, no. 1, pp. 65–73, Apr. 2004.
39. F. White, *Fluid Mechanics*, 7th ed., New York, NY, USA: McGraw-Hill Education, 2010, ch. 6, ISBN-13: 978-0-0735-2934-9.
40. A. White, Z. Ren, G. Zhu, and J. Choi, “Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ observer-based LPV control of a hydraulic engine cam phasing actuator," *IEEE Trans. on Control Syst. Tech.*, vol. 21, no. 1, pp. 229–238, Jan. 2013.
41. C. Gamache, G. Zhu, and T. He, “LPV boost pressure and EGR rate control of a diesel engine air charge system with eBoost assistance,” in *Proceedings 2024 (63rd) Conf. on Decis. Control (CDC),* Milan, Italy, December 16–19, 2024.
42. X. Wei and L. Re, "Gain scheduled $\mathcal{H}_\infty$ control for air path systems of diesel engines using LPV techniques," *IEEE Trans. on Control Syst. Tech.*, vol. 15, no. 3, pp. 406–415, May 2007.
43. F. Wu and M. G. Karolos, “LPV systems with parameter-varying time delays: analysis and control,” *Automatica*, vol. 37, no. 2, pp. 221–229, Feb. 2001.
44. A. White, G. Zhu, and J. Choi, *Linear Parameter Varying Control for Engineering Applications*, London, UK: Springer-Verlag, 2013, ch. 2, ISBN-13: 978-1-4471-5039-8.
45. H. Li, Y. Huang, G. Zhu, and L. Zheng, “Profile tracking for an electrohydraulic variable valve actuator using receding horizon LQT,” *IEEE/ASME Trans. on Mechatron.,* vol. 24, no. 1, pp. 338–349, Feb. 2019, doi: 10.1109/TMECH.2019.2892038.
46. P. C-K. Luk, H. A. Abdulrahem, and B. Xia, “Low-cost high-performance ferrite permanent magnet machines in EV applications: a comprehensive review,” *RTransportation*, vol. 6, p. 100080, Nov. 2020, doi: 10.1016/jetran.2020.100080.
47. R. Gabriel, W. Leonhard, and C. J. Nordby, "Field oriented control of standard AC motors using microprocessors," *IEEE Trans. on Ind. Appl.,* vol. 16, no. 2, pp. 186–192, Mar. 1980, doi: 10.1109/tia.1980.4503770.
48. C. J. O’Rourke, “A geometric interpretation of reference frames and transformations: $dq0$, Clark, and Park,” *IEEE Trans. on Energy Convers.,* vol. 34, no. 4, pp. 2070–2083, Dec. 2019.
49. M. A. Perales, M. M. Prats, R. Portillo, J. L. Mora, J. I. León, and L.G. Franquelo, “Three-dimensional space vector modulation in abc coordinates for four-leg voltage source converters,” *IEEE Power Electron. Lett.*, vol. 1, no. 4, pp. 104–109, Dec. 2003.
50. Ó. Lopez, J. Alvarez, J. Doval-Gandoy, and F. D. Freijedo, “Multilevel multiphase space vector PWM algorithm,” *IEEE Trans. on Ind. Electron.,* vol. 55, no. 5, pp. 1933–1942, May 2008.
51. V. R. Reddy and V. C. V. Reddy, “Mathematical analysis of SCPWM for Inverter fed DTC of induction motor drive,” *Int. J. of Appl. Eng. Res.*, vol. 13, no. 1, pp. 47–52, 2018.

Solutions to Practice Problems

1. $$SA = \frac{360^\circ}{2 \times 2 \times 6} = 15^\circ \tag{5.7.1}$$

    1.

    The solution for part b) is as below.

$$d_l = G_{r2l} \times \theta_r = 0.007056 \times 900 = 0.635\ mm \tag{5.7.2}$$

1. The control sequence for the desired movement is shown in the following table.

Moving Steps	Energized Solenoids	Resulting Polarities
Step #0	Solenoid #5 & #7	N (#5) & S (#7)
Step #1	Solenoid #4 & #6	N (#4) & S (#6)
Step #2	Solenoid #3 & #5	N (#3) & S (#5)
Step #3	Solenoid #2 & #4	N (#2) & S (#4)

1.
2. The three-torque curves plus the motor output torque curve can be plotted as below.

From the red curve, the first peak occurs at $\theta_{max} = \pi/6$ and the first valley at $\theta_{max} = \pi/3$. The maximum and minimal torque can be calculated as follows.

$$T_{max} = 2\sin\frac{\pi}{6} + 2\sin\left(\frac{\pi}{6} + \frac{\pi}{3}\right) + 2\sin\left(\frac{\pi}{6} + \frac{2\pi}{3}\right) = 1 + 2 + 1 = 4 \tag{5.7.3}$$

$$T_{min} = 2\sin\frac{\pi}{3} + 2\sin\left(\frac{\pi}{3} + \frac{\pi}{3}\right) + 2\sin\left(\frac{\pi}{3} + \frac{2\pi}{3}\right) = \sqrt{3} + \sqrt{3} + 0 = 2\sqrt{3} \tag{5.7.4}$$

and the percentage motor torque ripple is shown below.

$$R_{\%} = \frac{T_{max} - T_{min}}{T_{max}} \times 100 = \frac{4 - 2\sqrt{3}}{4} \times 100 = 13.4\% \tag{5.7.5}$$

This indicates that as the number of poles increases, the torque ripple reduces. For most DC motors in production, the number of poles is often greater than or equal to six.

1.

The no-load speed can be calculated using the equation below.

$$\omega_{no-load} = \frac{e_a}{K_m} = \frac{10}{0.5} = 20\ rad/s \tag{5.7.6}$$

Therefore, the calculated stall torque is $50\ Nm$ and the calculated no-load speed $20\ rad/s$.

1.

The no-load speed of the gearbox output shaft can be calculated using the equation below.

$$\frac{\omega}{\omega_o} = 0.5 \quad \rightarrow \quad \omega_o = \frac{\omega}{0.5} = \frac{8}{0.5} = 16\ rad/s \tag{5.7.7}$$

Therefore, the calculated gear ratio $\eta = 0.5$ and the calculated associated no-load speed $16\ rad/s$.

1.

The time constant can be calculated by the following equation.

$$T = \frac{JR_a}{R_a b_m + K_\tau K_m} = \frac{0.4 \times 0.1}{0.1 \times 2 + 0.5^2} = 0.08889\ s \tag{5.7.8}$$

Therefore, the calculated $K = 1.1111\ (rad/s)/A$ and the calculated $T = 0.08889$ second. The steady state speed in this case can be calculated by the following equation.

$$\omega_{ss} = K \times e_a = 1.1111 \times 5 = 5.555\ rad/s \tag{5.7.9}$$

1.

Therefore, the reference voltage v_{ref} is $1.5\ V$.

1.

With the 20% duty-cycle, the averaged motor current is shown below.

$$i_a = i_{a-on} \times 20\% = 60 \times 0.2 = 12\ A \tag{5.7.10}$$

1. The other braking regeneration operational mode, when the motor rotates in the anticlockwise direction, can be implemented by turning on only the upper-right switch, as illustrated below.
2. Because the PWM conversion block has a saturation effect to the analog control signal generated by the PID controller, the controller shown in Figure 5.29 is the fastest one.
3. The answer is c).
4. For the equivalent circuits of DC motor and EMVA (lower solenoid), a) the lower solenoid inductance, $L_{low}\,(i_{low}, y)$, is related to DC-motor armature inductance L_a; b) the lower solenoid back-EMF coefficient $K_{low}\,(i_{low}, y)$ is related to the back-EMF coefficient of DC-motor K_m, and c) EMVA linear motion speed $\dot{y}$ is related to the DC-motor rotational speed $\dot{\theta}$.

1.

Note that the torque generated by the DC motor is a linear function of armature current i_a (see DC-motor modeling in Chapter 2). The magnetic force generated by the EMVA lower solenoid is a quadratic function of its current i_{low}; see equation ???.

1. To delay the valve closing starting time from $15\ ms$ to $20\ ms$, we need to adjust the upper solenoid control input timing shown in the following table. Note that the upper solenoid time input value changes in the bottom line of the table.

Output voltage	380	380	120	12	12	0	0	0
Input time (ms)	0	1.5	2.2	6	19.95	25	25	40
Output voltage	0	0	380	380	120	12	12	12
Input time (ms)	0	14.95	20	21.2	22.5	26	30	40

1. The corresponding displacement, speed, and acceleration responses (similar to Figure 5.39) are plotted below.
2. The system matrix A described by equation ??? is affine because there is a constant term in equation ???. Control input matrix B defined by equation ??? is linear because there is no constant part in matrix B.
3. The MATLAB/Simulink diagram of the LPV EHVA system is shown below.
4. Part a) is true. Part b) is false; the correct statement is: The speed of an induction motor is slightly below its stator current frequency.

1. The pseudo inverse of T_{clark} can be calculated by conducting SVD first as below for T_{clark}.

$$T_{clark} = U \cdot S \cdot V^* = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.8165 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.707 & 0.707 \\ 0 & 0.707 & 0.707 \end{bmatrix} \tag{5.7.11}$$

and

$$T^+_{clark} = \begin{bmatrix} 1 & 0 \\ 0 & -0.707 \\ 0 & 0.707 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0.8165^{-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0.866 \\ 0 & -0.866 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{\sqrt{3}}{2} \\ 0 & -\frac{\sqrt{3}}{2} \end{bmatrix} \tag{5.7.12}$$

This page titled 5.7: Chapter References is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6: MECHATRONIC SYSTEM COMPONENT- SENSORS

Learning Objectives

1. Introduce the two sensor groups: physics-based and image/video-based sensing technologies and explain the concept of sensor fusion technologies.
2. Describe working principles of both analog and digital rotational speed sensors along with the signal processing of the hall-effect incremental encoder signals for rotational direction and speed.
3. Provide working principles of inertia-based acceleration sensor using piezoelectric material and capacitance as the sensing elements along with the concept of realizing the sensor in MEMS. Also describe the coordinate transform of two- and three-dimensional sensor coordination.
4. Introduce the working principle of an angular velocity sensor (rate gyro) along with its MEMS and piezoelectric realizations, based on the Coriolis force effect.
5. Explain the concept of sensor fusion and its application to tilt-angle sensing by fusion of rate gyro and accelerometer signals.
6. Describe MEMS-based sensor technology, including its architecture, material used, and the manufacturing process for MPU 6050 3-axis accelerometers and rate gyros.
7. Summarize working principles of temperature, pressure, mass air flow, current, and voltage sensing and the associated electronics.
8. Outline image-based sensing, with examples of its application for detecting faulty emitter nozzles and road lanes.

This chapter provides an introduction to sensors used in mechatronic systems. These include rotational and linear speed sensors as well as accelerometers for measuring linear acceleration. They also include, rate gyro sensors used to measure angular velocity. We will briefly review temperature, pressure, air flow, current, and voltage sensors, and introduce the concept of sensor fusion technology.

6.0: Prelude

6.1: Mechatronic Sensor Background

6.2: Rotational Speed Sensors and Their Speed Calculation

6.3: Acceleration Sensors

6.4: Angular Velocity Sensors (Rate Gyro)

6.5: Sensor Fusion

6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing

6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing

6.8: Vision-Based Sensing

6.9: Chapter References

This page titled 6: Mechatronic System Component- Sensors is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6.0: Prelude

Earlier in Chapter 1, we discussed that a mechatronic system's physical variables are often converted to analog electronic signals and then sampled into the information domain as system feedback signals. They use A/D converters or input directly into the information domain in a digital manner. In the information domain, the microprocessor generates the control signals for these mechatronic actuators (covered in Chapter 5) based upon the sampled sensor signals. In this chapter, we will address the sensor subsystems in the physical domain. Note that the communication from the physical to information domain is through either A/D (analog-to-digital) conversion or direct digital input (digital sampling and digital pulse counting). There are two counting methods: counting the number of input pulses for a given time period and counting the number of clock pulses when the state of input pulse train is high (or low). On the other hand, the communication from the information to physical domain is through either D/A (digital-to-analog) conversion or direct digital output.

Sensors are critical for closed-loop controlled mechatronic systems, which directly or indirectly measure system physical variables such as speed and position. They provide feedback signals for the closed-loop controller. In this chapter, we introduce various sensors, including these used in mini-Segways (Chapter 8) and three-wheel carts (Chapter 9), along with their physical principles. We also discuss how their signals are input to the information domain.

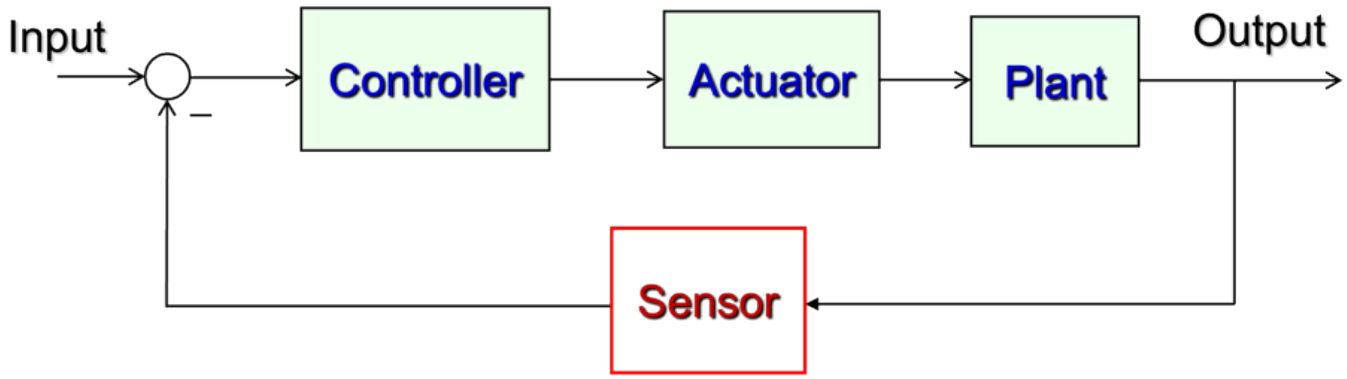


Figure 6.1: Sensors in a closed-loop controlled mechatronic system.

In a mechatronic system, sensors use physical laws (e.g., hall-effect, optics, electromagnetism, image processing) to convert physical parameters to electrical outputs (voltage, current, and pulse train). They also provide feedback to system microcontrollers in the information domain. Sensors can be divided into analog sensors with continuous output signals and digital (or discrete) sensors with high and low logic outputs, respectively. Table 6.1 lists their advantages and disadvantages.

Table 6.1: Analog and digital sensor features.

Analog sensors	Digital sensors
Limited dynamic range of measured parameters/ variables Direct measurement, however, subject to noise pollution No need of further signal processing or conversing	Possible to achieve high dynamic range Output digital signal with high or low voltage/current, thus, with high noise rejection capabilityNeed further signal processing to convert measurements to true valuesAccuracy often depends on digital (clock) resolution

Section 6.1 reviews various mechatronic sensors. These include detecting variables in the physical domain which measure physical quantities (e.g., voltage, pulse train, current) and can be input to the information domain. They also include sensors that capture images and videos in the physical domain that allow microprocessors in the information domain to convert them into useful information. Examples are distance and direction of an object, and lane detection for a vehicle.

There are also sensors that use multiple-sensor fusion technology to generate useful measurement based on multiple sensor signals. For example, with both physical variable measurements and image/movie sensors, we can obtain a more accurate location estimate of an indoor object by combining multiple data sources including video cameras and Wi-Fi localization signals. In this case, a Wi-Fi positioning system (WPS, WIPS or WFPS) is a geolocation system that uses the characteristics of nearby Wi-Fi access points to discover the device location [1].

Section 6.2 discusses rotational speed sensing including analog and digital sensors for both rotational and linear speed. The analog rotational speed sensor can be easily realized by using a DC motor (see [2] and Chapter 2). When the DC motor spins, its armature voltage output is proportional to the rotational speed. A digital speed sensor can be realized by an encoder [3] with the help of a so-called hall-effect sensor. In this case, a digital counting device in the interface from the physical to the information domain is required to convert the pulse train generated by the hall-effect sensor to actual digital speed signal.

Linear speed sensing can often be realized by converting the rotational speed signal to a linear one. For example, vehicle travel speed can be calculated from its wheel rotational speed assuming small slip between tire and road. One of the direct measurement methods of linear speed is the radar- or laser-based linear velocity sensor that emits a radar wave. This wave is reflected by the target surface and returned to the sensor; the speed is determined by measuring the frequency difference (Doppler effect) [4].

Section 6.3 addresses these sensors used to measure the acceleration of an object. In Section 2.4, we modeled the piezoelectric accelerometer with detailed working principle. In this section, we will provide detailed information regarding the MEMS (micro-electromechanical systems) technology used in these advanced sensors. This technology integrates microscale mechanical sensors into a single silicon chip along with associated electronic circuits. As a result, both the mechanical sensor and the required electronics are integrated. We will also discuss multidirectional accelerometer technology. An alternative method for obtaining the acceleration signal is to take the time derivative of a measured speed signal if the required bandwidth is relatively low. In this case, a low-pass filter is often used to reduce the noise generated by taking the derivative.

Section 6.4 addresses rate gyro (angular velocity) sensors. In general, rate gyro angular speed sensors can be divided into two groups: mechanical (including vibration) [5] and optical [6]. We review them in detail in this section. The vibration rate gyro sensor technology is discussed in detail, from its principle to the associated sensor hardware, and then its implementation into MEMS. In addition, we briefly review the piezoelectric vibration rate gyro. We then briefly introduce the principle of optical gyro sensors.

Section 6.5 addresses the concept of sensor fusion [7]. It combines multiple-sensor data or data derived from disparate sources so that the resulting information is more reliable than using these sources individually. The section presents two examples of fusing two sensor signals with different measurement noise levels, indicating that, in these cases, sensor fusion significantly reduces the measurement noise. In addition, the section shows that the Kalman filter can be deployed for sensor fusion. We use an example of how both accelerometer and rate gyro signals help estimate the rotational angle and how the estimation result improves, compared to using the integrated rate gyro signal (rotational speed signal).

Section 6.6 addresses the dimension, system architecture, and making of a micro-electromechanical system (MEMS). We also introduce a 3-axis accelerometer and gyroscope integrated sensor system MPU 6050 [8], utilizing MEMS technology. The MPU 6050 is the world's first integrated 6-axis motion tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and digital motion processor (DMP) all in a small $4 \times 4 \times 0.9$ mm package. Here, the mechanical acceleration and gyroscope sensors are integrated with the digital motion processor (DMP). Both complex motion fusion algorithms and run-time calibration firmware are processed in DMP to eliminate the costly and complex calibration process.

Section 6.7 briefly discusses a variety of sensors not already covered, including temperature, pressure, flow, current, and voltage sensing. We start with the RTD (resistance temperature device) temperature [9] and pressure sensing [10], followed by the mass air flow sensor [11] that uses hot wire air flow sensing technology. Current and voltage sensing technology widely used in electronic systems, such as inverter and battery management systems, are then addressed along with a few examples. Note that these sensing technologies are important for many mechatronic applications.

Finally, Section 6.8 briefly reviews another class of sensing technology that is image-based. For example, camera images and video can be used to diagnose water and chemical spray systems in an orchard [12] and to detect road lanes for autonomous driving [13]. These applications may have one or more sensing components, e.g., a conventional camera or one with an infrared (IR) video camera. The resulting images and videos can be used to sense multiple objects such as chemical spray situations (normal or abnormal), lane locations, and types of front moving objects. Sensor fusion technology is often used to combine multiple sensing sources to improve estimation accuracy.

Orienting Questions

1. What are the two sensing technology groups?
2. What is the rotational speed sensor? How can the hall-effect sensor signals be converted into direction and speed information?
3. How do the MEMS accelerometer and rate gyro sensors work?

4. What is the functionality of an MPU-6050 sensor?
5. What are technologies used in temperature, pressure, and flow sensors?
6. How is the faulty emitter nozzle detected in an agriculture spray system?

This page titled 6.0: Prelude is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6.1: Mechatronic Sensor Background

Learning Objectives

After completing this section, you will be able to:

1. Describe two sensing technology categories: direct physics-based detection and graphic- (image/video) based detection.
2. Distinguish the characteristics of direct physical variable detection and graphic-based detection technologies.

In general, a mechatronic sensor converts the system variables from the physical domain into electronic signals (such as voltages, currents, pulse trains, and environmental information) that can be sampled into the information domain through analog-to-digital conversion or the direct digital input method. The mechatronic sensor is defined below.

Definition 6.1

A mechatronic sensor is a device that converts system variables from the physical domain into electronic signals such as voltage, current, pulse-train, and environmental information. These signals can then be sampled into the information domain, utilizing mechanical, electromagnetic, and image processing approaches.

Remark 6.1

In a mechatronic system, sensors are essential for the information domain to obtain information in the physical domain so that the feedback control and optimization can be realized.

Sensors use two main approaches to convert physical domain information to be input to the information domain. One is to directly convert the system variables in the physical domain so that they can be input into the information domain. The other is to convert captured images and videos into variables that can be input into the information domain.

Let's consider two examples of direct conversion sensing technologies. The first example is the current sensing widely used in feedback control of the electric motors (DC and permanent magnetic synchronous motors). We will demonstrate how to use the DC-motor current feedback to control a DC-motor torque output with detailed current sensing description. Figure 6.2 shows a diagram of PID (proportional-integral-derivative) negative feedback control; this diagram is for DC-motor output torque control based on current feedback.

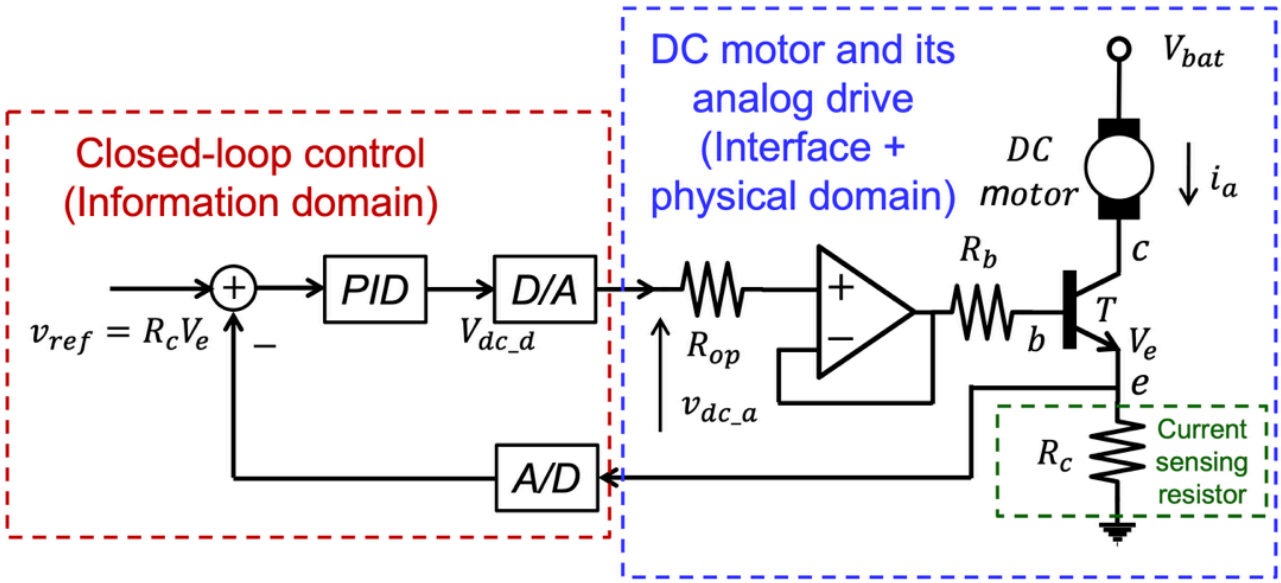


Figure 6.2: DC-motor PID torque control with current feedback.

The simple mechatronic system in Figure 6.2 is used for closed-loop control of a DC-motor current, where v_{ref} is proportional to the desired DC-motor armature current and i_a is the actual DC-motor armature current. Because DC-motor output torque is proportional to its armature current, controlling the DC-motor armature current is equivalent to controlling its output torque.

The PID control block in the information domain (red-dashed line box) is discussed in Section 3.4 with both A/D (analog-to-digital) and D/A (digital-to-analog) conversion blocks connecting to the interface between the information and physical domains. Within the physical domain (blue-dashed box) plus the interface block, a low-side drive circuit is used to drive the DC motor with an operational amplifier used to isolate the control signal from the D/A converter. Resistors R_{op}, R_b, and R_c represent the operational amplifier input resistor, transistor input resistor, and current sensing resistor, respectively. Assuming large amplification coefficient for the transistor T, the current flow through the current sensing resistor R_c can be used to approximate the motor armature current i_a.

The goal of this mechatronic system is to make the DC motor produce the torque at the reference level. Section 2.4 and Section 5.2 indicate that the torque generated by the DC motor is proportional to the motor armature current i_a as shown in equation 6.1.1.

$$\tau = K_\tau i_a \tag{6.1.1}$$

where K_τ is the DC-motor torque coefficient and τ is the DC-motor output torque. From Figure 6.2, we find that the voltage on the current sensing resistor R_c can be calculated using the following equation based on current and voltage relationship of a resistor.

$$V_e \approx R_c i_a \tag{6.1.2}$$

As a result, to regulate the armature current i_a to the desired level denoted by i_{ref}, the reference voltage level v_{ref} can be calculated using the following equation.

$$v_{ref} = R_c i_{ref} = \frac{R_c}{K_\tau}\tau_{desired} \tag{6.1.3}$$

where $\tau_{desired}$ is the reference DC-motor torque. Therefore, with the reference voltage v_{ref} defined by equation 6.1.3, the DC-motor output torque τ can be regulated to the target level of τ_{ref} with a properly tuned PID controller; see Section 3.4 for PID gain tuning. In this case, the achieved DC-motor armature current is i_{ref}.

When a very large armature current is required to achieve the desired torque, resistance-based current sensing is not preferred; a lot of electrical energy can be consumed by the resistor, leading to a high level of heat generation. For high current sensing, electromagnetic current sensing is preferred instead; see Section 6.7 for detail.

Example 6.1

Consider the DC-motor torque management system in Figure 6.2. Find the reference voltage v_{ref} so that the DC-motor output torque is regulated to 50 Nm, assuming that the current sensing resistor value is $R_c = 0.1\ \Omega$ and the DC-motor torque coefficient $K_\tau = 2.0\ Nm/A$.

Note that equation 6.1.3 can be used for this calculation as follows.

$$v_{ref} = \frac{R_c}{K_\tau}\tau_{desired} = \frac{0.1 \times 50}{2.0} = 2.5\ (V) \tag{6.1.4}$$

In this case, the associated DC-motor armature current can be calculated as follows.

$$i_a = \frac{\tau}{K_\tau} = \frac{50}{2.0} = 25\ (A) \tag{6.1.5}$$

The resulting energy consumed by the current sensing resistor R_c can be calculated as follows.

$$e_c = R_c i_a^2 = 0.1 \times 25^2 = 62.5\ (W) \tag{6.1.6}$$

Therefore, the energy wasted on the current sensing resistor is very high with a lot of heat generated. Either the current sensing resistor value should be reduced or an electromagnetic current sensor should be used (see Section 6.7).

Next, we will provide an example for the second type of sensing technique, which is based on image processing. This study is related to the internal combustion engine research for autoignition control of a homogenous charge compression ignition (HCCI) process [14], [15]. It is critical to achieve a good charge mixing of fresh air charge and residual gas for initiating HCCI combustion process. Therefore, it is useful to develop a mixing model for the operational range of HCCI combustion; the charge mix temperature determines ignition timing for HCCI combustion [15]. One of these models was developed in reference [15].

For model derivation, the cylinder volume is artificially divided into two zones with a fictitious boundary dividing the two. One is the fresh charge (unmixed) zone induced by opening intake valves. The other is the residual gas zone transferred from the unmixed to the mixed zone. Each zone is assumed to have been mixed homogeneously. Mass transfer between them, which is forced by a fluid jet, is directed from the unmixed zone to the mixed one.

To validate the model developed in reference [15], an optical engine test bed was developed for model validation purpose (see Figure 6.3). Here, an IR (infrared) camera is used to take images inside the engine combustion chamber through a reflection mirror and optical piston lens. Inside the combustion chamber, the red cloud represents the hot burned gas. The white background inside the combustion chamber indicates the fresh air charge flew into the piston after the engine intake valve is opened with the exhaust valve closed. In this case, the engine piston moves downward.

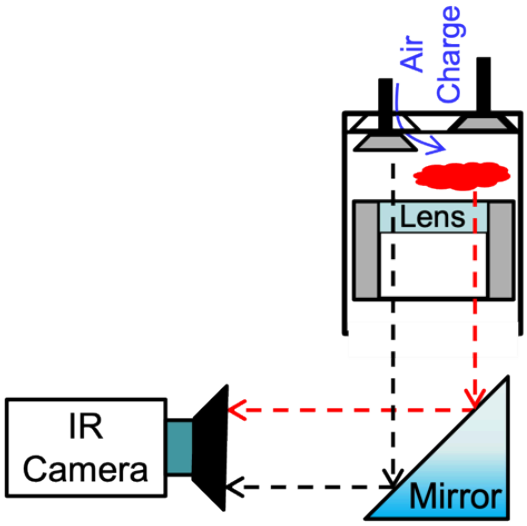


Figure 6.3: Illustration of an optical engine test bed.

Optical engine tests were performed with an infrared (IR) camera. Infrared images were collected by the IR camera every 10 degrees at crank angles of 30°, 40°, 50°, 60°, and 70° after the exhaust top-dead-center (AE-TDC). These resulted in the five gray pictures at the bottom row of Figure 6.4, where the bright dot in the original images is due to the high temperature of the spark-plug electrode. Under the same conditions, GT-power simulations were conducted (GT-power is an advanced engine combustion simulation software [16]). The purpose was to calculate the fresh air charge mass temperature inside the unmixed zone (represented by the white background inside the combustion chamber) and the mixed zone (represented by the red cloud in Figure 6.3).

To estimate mean temperature of the two zones (unmixed and mixed), an IR image should be divided into the two zones according to its intensity distribution. Because raw IR images inherently have anisotropic noise [17], a median filter is applied. It will filter out undesired noise sufficiently without unnecessary distortion of structural features, for example, edges [18]. In addition to suppressing noise, the filter gets rid of isolated pixels in the mixed zone. If a tiny portion of the residual gas is mixed with the fresh charge, it appears as a small-isolated region having high intensity. By applying a median filter, such an isolated region can be included in the mixed zone. The filtered images are shown in the middle row of Figure 6.4. These four semicircles shown in the middle and top rows represents locations of two intake and two exhaust valves.

After filtering (see the middle row of Figure 6.4), pixels outside the optical window are excluded (see the dark-blue area outside of optical window in the top row of Figure 6.4). Note that only these pixels showing intensity of in-cylinder gas are used for zone identification from IR images. The pixels are also ruled out where the spark plug locates, as seen by these dark-blue circles on the top row of Figure 6.4. Because soot is presently used to estimate temperature, it is likely to accumulate on the spark plug and cylinder head. Note that the temperature of the spark-plug electrode remains high, which could lead to excessively high image output around the spark plug. Therefore, pixels—except for the area covered by the dark-blue circle—are used for image segmentation. To partition the images into unmixed (red) and mixed (yellow) zones in the top row of Figure 6.4, the k -means cluster method [19] is employed. This technique partitions pixels into k clusters so that a predetermined cost function is minimized. The partitioned images are shown in the top row of Figure 6.4, where the light-blue area represents the burned zone.

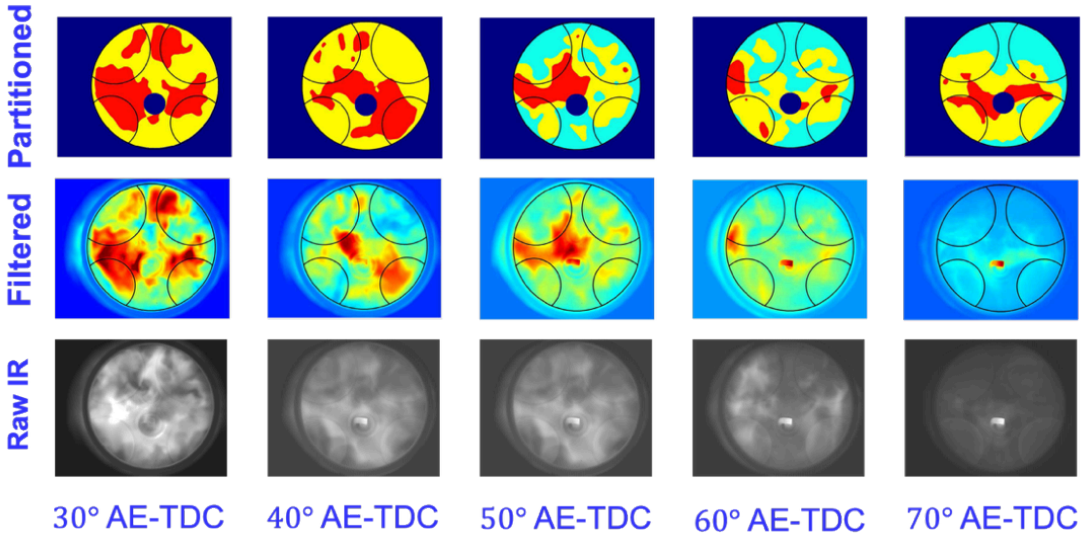


Figure 6.4: IR camera images and the associated processed images.

The temperature of the mixed zone (red cloud area in Figure 6.3) can be estimated by applying the method used in [14]. Figure 6.5 presents mixed-zone temperature levels of the GT-Power simulated and image-based estimations. The red squares represent the GT-Power simulation results. The magenta circles are calculated based on IR images through a process of filtering the IR images, partitioning the filtered images into two zones, and calculating the temperature of the mixed zone, using the method specified in [14]. We can see that the two estimations are pretty close.

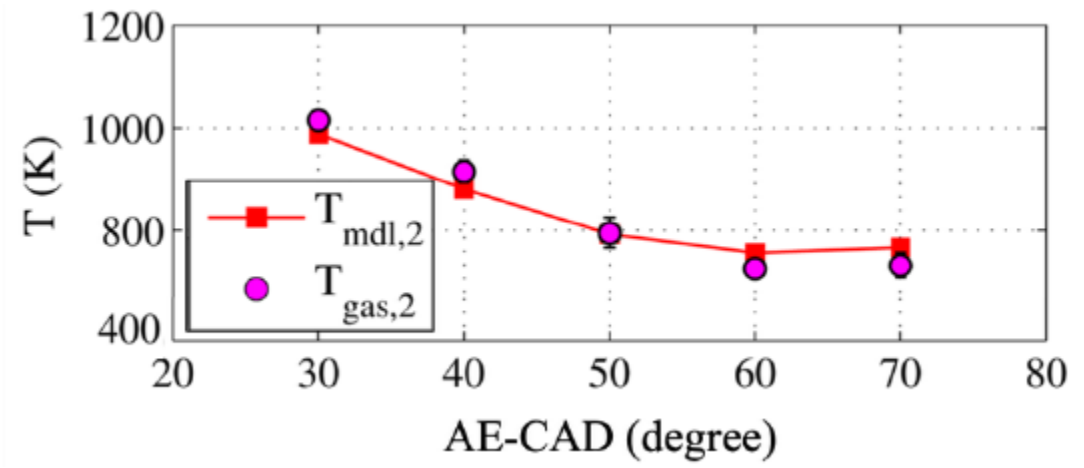


Figure 6.5: Image-processed and GT-Power modeled mixed-zone temperatures.

To summarize, the image-based sensing technology is often used for detecting the variables when a physical sensor cannot be placed. For the example of detecting the mixed-zone temperature inside the combustion chamber, it is impossible to put a temperature sensor inside; it would disturb the combustion and charge mixing process. In addition, the combustion process is completed within tens of milliseconds. However, most physical temperature sensors, such as thermal couples, are not fast enough for temperature sensing. In such cases, image-based temperature sensing is a good option.

We will briefly discuss the concept of sensor fusion technique used in mechatronic systems; a detailed example will be provided in Section 6.5 to detect the pendulum angle of a mini-Segway using the fusion of accelerometer and rate gyro signals. But for now, consider a temperature-sensing problem using two temperature sensors. Sensor A has a range of zero to 200 $^\circ F$ with an error bound of 2% of maximum temperature (200 $^\circ F$). Sensor B has a range of zero to 400 $^\circ F$ with an error bound of 2% of maximum temperature (400 $^\circ F$). If sensor B is used alone to measure the temperature, then within the range of zero to 200 $^\circ F$, the error bound can be calculated by the following equation.

$$e_{0-200} = \frac{(400\ ^\circ F) \times 2\%}{200\ ^\circ F} = 4\% \tag{6.1.7}$$

However, if sensor A is used over that range, the error e_{0-200} can be reduced down to 2%. That is, over the range of zero to 200 $^{\circ}F$, sensor A signal should be used, whereas over the range between $200\ ^{\circ}F$ and $400\ ^{\circ}F$, sensor B signal should be used.

Section Summary

This section reviews various mechatronic sensors including those that convert mechatronic system variables in the physical domain into physical quantities such as voltage, pulse train, and current, which can be input to the information domain. The section also reviews those sensors that capture images and movies in the physical domain. Microprocessors and microcontrollers in the information domain can then convert them into useful information, such as the distance and direction of an object, or lane detection for a vehicle. There are also sensor technologies that use multiple-sensor fusion to generate helpful measurements based on multiple sensor signals. For example, with both physical variable measurement and image/movie sensors, we can obtain a more accurate location estimate of an indoor object by combining multiple data sources such as video cameras and Wi-Fi localization signals.

Section Keywords

Practice Problems

1. What are the two-categories of sensing approaches?
2. Consider the DC-motor torque management system in Figure 6.2. Find the reference voltage v_{ref} so that the DC-motor output torque is regulated to 50 Nm, assuming that the current sensing resistor value is $R_c = 0.02\ \Omega$ and the DC-motor torque coefficient $K_\tau = 2.0\ NM/A$. Then find the energy consumed by the current sensing resistor.
3. Provide two situations when temperature is not suitable for direct measurement by a physical sensor.
4. Provide an example of sensor fusion using at least two sensor signals.

This page titled 6.1: Mechatronic Sensor Background is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6.2: Rotational Speed Sensors and Their Speed Calculation

Learning Objectives

After completing this section, you will be able to:

1. Provide the working principle of an encode-based rotational speed sensor.
2. Describe the hall-effect sensor along with its working principle.
3. Calculate the rotational speed based upon the pulse train generated by the hall-effect sensor.

As mentioned in the beginning of this chapter, rotational speed sensors can be divided into two groups: analog and digital. An analog rotational speed sensor generates an analog voltage or current signal that is proportional to the rotational speed. A digital rotational sensor generates a pulse train with its frequency (or a digital number directly) proportional to the rotational speed. An analog speed sensor signal in the physical domain can be input to the information domain via the A/D (analog-to-digital) converter. The digital rotational sensor signal will be read directly into the information domain by counting the number of pulses—if the digital sensor signal output is in the form of a pulse train or through a digital communication network directly.

First, we will discuss the analog rotational speed sensor. Recall that the DC-motor characteristics discussed in Sections 2.4 and 5.2 indicate that the DC-motor back-EMF (electromagnetic field) voltage is proportional to the motor speed, as below.

$$e_m(t) = K_m \dot{\theta}(t) \tag{6.2.1}$$

where $e_m(t)$ is the back-EMF voltage, K_m is the fixed-motor back-EMF coefficient, and $\dot{\theta}(t)$ is the DC-motor rotational speed. If the motor shaft is connected to the rotational shaft measuring its speed, with no armature voltage applied, the DC-motor armature circuit can be shown as in Figure 6.6.

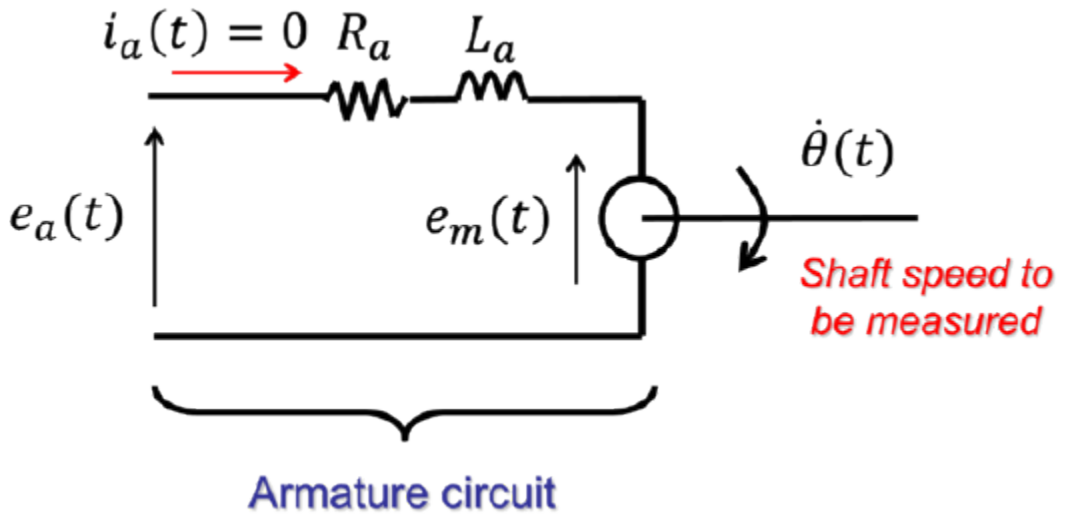


Figure 6.6: Analog rotational speed sensing by a DC motor.

Because there is no armature voltage applied with an open armature circuit, the armature voltage e_a equals the back-EMF voltage, assuming the armature current $i_a = 0$. The DC-motor shaft speed can be solved by the following equation. In practical applications, a low-pass filter is often used to get rid of output noise.

$$\dot{\theta}(t) = \frac{e_m(t)}{K_m} = \frac{e_a(t)}{K_m} \tag{6.2.2}$$

Example 6.2

Let the back-EMF coefficient $K_m = 0.5\ V/(rad/s)$. If the armature voltage output is 10 volts, find the motor shaft speed. Note that the motor shaft speed can be found using equation 6.2.3.

$$\dot{\theta}(t) = \frac{e_a(t)}{K_m} = \frac{10\ (V)}{0.5\ (V/(rad/s))} = 20\ rad/s \tag{6.2.3}$$

Before getting into digital rotational speed sensing, we first need to introduce the hall-effect sensor [20]. Hall-effect sensors are devices that are activated by an external magnetic field, where the output signal from a hall-effect sensor is the function of magnetic field density around the device. When the magnetic flux density around the sensor exceeds a certain preset threshold, the sensor detects it and generates an output voltage, v_{hall}, called the hall-voltage.

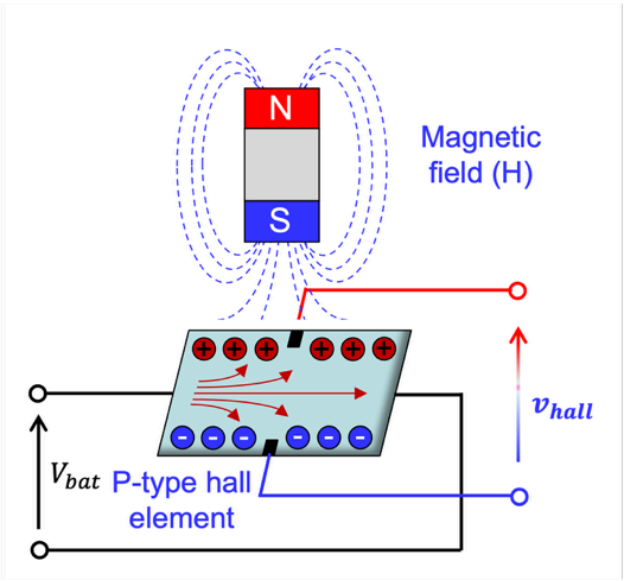


Figure 6.7: Hall-effect sensing principle.

The main component of a hall-effect sensor consists basically of a thin piece of rectangular p-type semiconductor material (see Figure 6.7) such as gallium arsenide (GaAs), indium antimonide (InSb), or indium arsenide (InAs). The material has a continuous current through itself, where a power supply voltage V_{bat} is applied. When the external magnetic field (H) is activated or the magnetic flux around the semiconductor exceeds a threshold, the current is redirected to generate a hall-voltage v_{hall}.

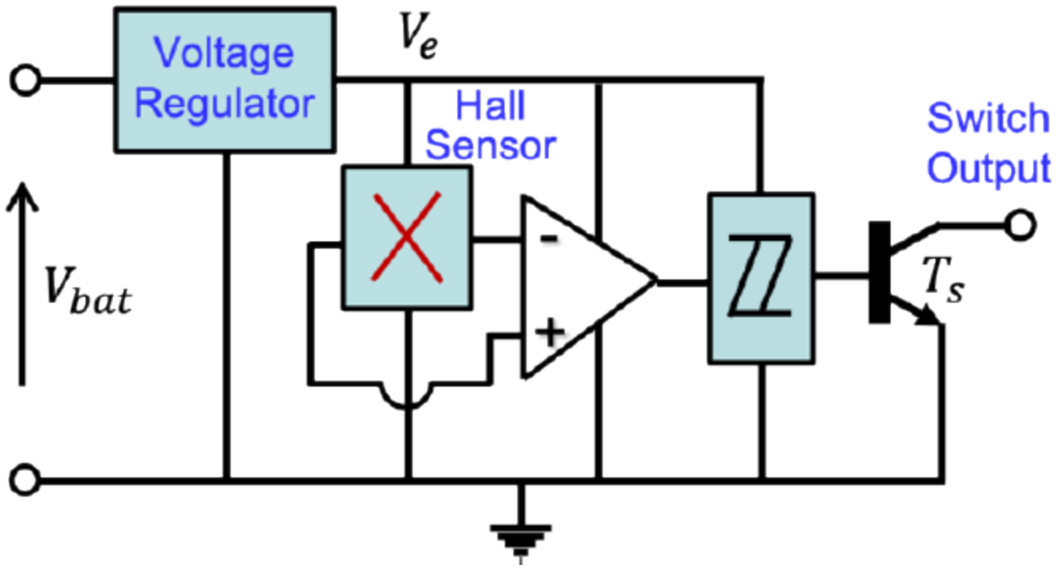


Figure 6.8: A sample hall-effect sensor.

Figure 6.8 shows an architecture diagram of a hall-effect sensor, where a voltage regulator is used to supply a stable voltage to the hall sensor, operational amplifier, and hysteresis logic block. That is, the hall-effect sensor output depends not just on the current magnetic field input, but also on its past output, creating a "lag." Note that the hall-effect circuit output is a low side switch signal that is turned on and off by the hall-effect sensor, which is affected by the outside magnetic field (Figure 6.7).

To utilize the switched output of a hall-effect sensor, a burn resistor of around a few hundred ohms will be used. The burn resistor connects to the switched output of the hall-effect sensor and the power supply. With the connected resistor, the switched output of the hall-effect sensor outputs a voltage high signal with no outside magnetic field and a voltage low signal with an active outside magnetic field.

Now the question is how to form a rotational speed sensor based on hall-effect sensors. Note that rotational speed sensing needs two signals: rotational direction and magnitude. Therefore, let's discuss how to convert the hall-effect sensor signals to these two signals using the hall-effect based incremental encoder. Figure 6.9 illustrates the hall-effect encoder using two hall-effect sensors.

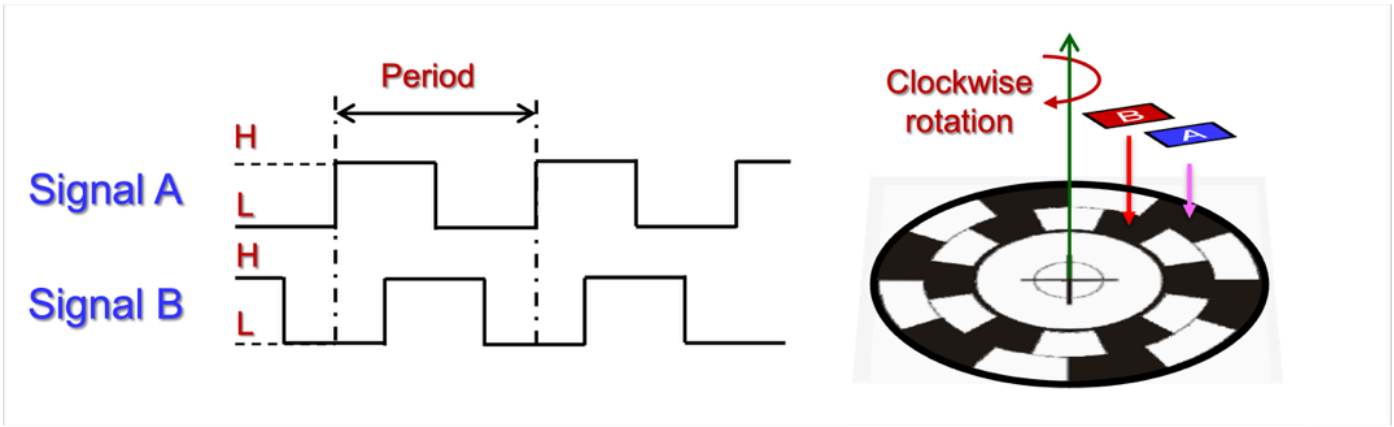


Figure 6.9: An incremental encoder and its output signals.

The right plot of Figure 6.9 shows that two hall-effect sensors (A and B) are used in the encoder system. Hall-effect sensor A points to the outside magnetic ring and sensor B points to the inside ring. On the magnetic disk, the magnetic ring is evenly spaced with six magnetized areas denoted in black and six non-magnetized areas denoted in white. When the magnetic disk makes a clockwise rotation as indicated by the red arrow, two pulse signals (A and B) are generated by the two hall-effect sensors (A and B), respectively. See the left plot of Figure 6.9, where the period of the pulse signal A is inversely proportional to the rotational speed. (Signals A and B for the anticlockwise rotation case can be found in the solution for practice problem P6.6.) We can calculate the rotational speed by counting the number of pulses over a given time period or sample period; see Section 4.6

for details. In Example 6.3, we calculate the rotational speed.

Example 6.3

For the incremental encoder shown in Figure 6.9, if the sample period $T = 0.1$ (second) and the number of pulses counted over the sample period is $n_p = 15$, find the rotation speed in RPM (rotations per minute).

For the encoder in Figure 6.9, each rotation generates six pulses—there are six magnetized areas per rotation. In this case, the number of pulses (n_p) divided by the number of pulses per rotation (n_m) is the number of rotations ($n_{p/r}$) within the sample period T. We can calculate the rotational speed in RPM by dividing $n_{p/r}$ by the sample period *T* and multiplying by 60 . Equation 6.2.4 provides a generalized formula for calculating the rotational speed using the incremental encoder, along with the speed calculation for the data provided in this example.

$$N = \frac{n_p \times 60}{T \times n_m} = \frac{15 \times 60}{0.1 \times 6} = 1500 \ (RPM) \tag{6.2.4}$$

Note that multiplication by 60 is needed to convert from “per second” to “per minute,” and n_m is the number of magnetized areas per rotation. Substituting these parameters into equation 6.2.4, we find the rotational speed is 1500 RPM. Note that RPM is often used instead of (rad/s) as the engineering unit of rotational speed.

To determine the rotational direction, we need to determine whether the rising edge of the hall-effect sensor signal A is leading the signal B. In this case, we can use two trigged blocks (connected to trigged signals A and B and located on the left of Figure 6.10) to record both rising edge time values, respectively. A trigged block contains code to be executed only when the trigged input on top of the block is active. For example, the code inside of the trigged block (with a rising edge) on the top-left of Figure 6.10 will be executed only when a rising edge of signal A is detected, and the code inside of the trigged block (with a pulse) on the top-right of Figure 6.10 will be executed only when the trigged input is high. We then calculate the difference to find out the rotation direction. Figure 6.10 shows the detailed logic.

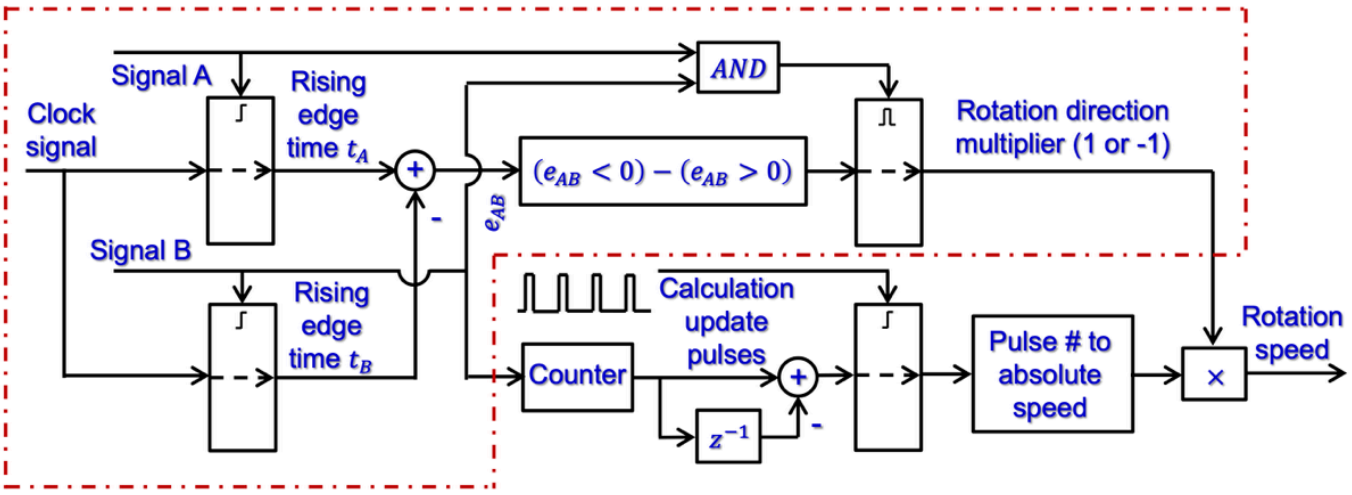


Figure 6.10: A flow diagram for calculating the shaft speed from incremental encoder signals.

Figure 6.10 shows a detailed logical diagram of calculating the shaft rotational speed, where positive is clockwise and negative is anticlockwise. First, consider the red dash-dotted box used to identify the rotational direction. The two trigged blocks at the most left position are trigged by the incremental encoder signals A and B. At the rising edges of signals A and B, the rising edge time values are recorded for signals A and B, respectively. The values t_A and t_B are locked until the next rising edges of signals A and B arrive, respectively. Then, we calculate the difference ($e_{AB} = t_A - t_B$) and input it to the logic block described by logic $(e_{AB} < 0) - (e_{AB} > 0)$. That is, when $(e_{AB} < 0)$, the output of rotational direction multiplier is one (1), and when $(e_{AB} > 0)$, the output is negative one (-1). Because this output is true only when levels of signals A and B are “high”, signals A and B are sent into an “AND” logic block with its output equal to one when both signal levels are “high.”

These blocks outside the dash-dotted box in Figure 6.10 are used for calculating the absolute rotational speed, where a digital counter is used to obtain the number of pulses of signal B. It doesn’t matter if signal A or B is used in this case. The counted

pulse number of signal B is subtracted by the delayed counted number using delay operator z^{-1} (one sample delay). The resulting number is input to the trigged block enabled by the rising edge of calculation update pulse signal with its period equal to sample period time T.

The output of the trigged block is the number of pulses n_p counted over the time period of T. Along with the given sample period of T and the number of evenly distributed number of magnetized areas n_m on the sensor disk logic, the block named by "Pulse # to absolute speed" calculates the absolute sensor shaft rotational speed using equation 6.2.4. The calculated absolute rotational speed is multiplied by the rotational direction multiplier (which is generated by the red dash-dotted box). Their product is the rational speed, where a positive number indicates clockwise rotation direction and a negative number leads to an anticlockwise rotational direction.

At first, the logic of these calculations in Figure 6.10 looks very complex. However, in the practical implementation, these blocks inside the red dash-dotted box and the counter block outside the box are realized by electronic hardware. Only the remaining logics outside the red dash-dotted box (excluding the counter) are implemented in software. Also, Section 4.6 indicated that the time a pulse train's level is "high" can be found by counting a clock signal (see Figure 4.61). As a result, we can calculate the frequency f in Hz of pulse signal A or B by using equation ???. We can then calculate the absolute rotational speed in RPM using the following formula.

$$N = \frac{f \times 60}{n_m} \ (RPM) \tag{6.2.5}$$

In practice, a gearbox is often connected to the motor shaft to reduce motor speed and increase its torque output. Figure 6.11 show a DC motor, used in the mini-Segway (see Chapter 8). It has an incremental encoder on the backside connected to the DC motor through a gear and a speed reduction gearbox in the front. Example 6.4 demonstrates how to calculate the geared output shaft speed using the counted number over a sample period based on the geared incremental encoder output signal A or B.

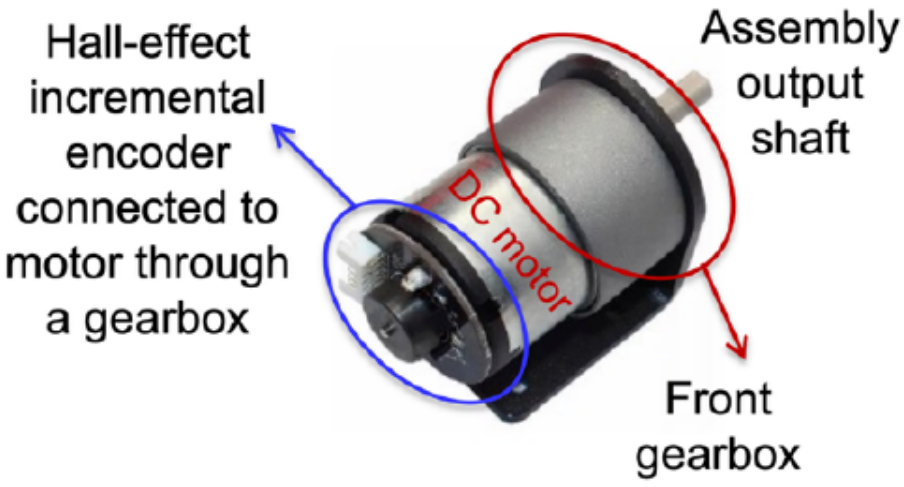


Figure 6.11: A DC motor (JGB37-520) with a geared incremental encoder and gearbox.

Example 6.4

For the DC motor shown in Figure 6.11 and used in the mini-Segway (see Chapter 8), the incremental encoder has eleven magnetized areas ($n_m = 11$). It is connected to the back side of the DC-motor shaft through a speed reduction gearbox with a gear ratio of $\eta_{en} = 4$. The DC motor is connected to the assembly output shaft through a speed-reduction gearbox with a gear ratio of $\eta_o = 30$.

Now, let's find the number of pulses per rotation of the assembly output shaft. One rotation of the assembly output shaft results in 30 rotations of the DC motor. In turn, one rotation of the DC motor leads to 4 rotations of the incremental encoder. As a result, one rotation of the assembly output shaft results in $\eta_{en} \times \eta_o = 4 \times 30 = 120$ rotations of the incremental encoder. In this case, the equivalent number of magnetized areas of the incremental encoder can be calculated as below.

$$n_{me} = n_m \times \eta_{en} \times \eta_o = 11 \times 4 \times 30 = 1320 \tag{6.2.6}$$

That is, one rotation of the assembly output shaft leads to 1320 pulses. We can calculate the resolution of the incremental encoder with front and back gearboxes as below.

$$e_{res} = \frac{360^\circ}{n_{me}} = \frac{360^\circ}{1320} = 0.2727^\circ \tag{6.2.7}$$

That is, the angle between two incremental pulses represents 0.2727° at the assembly output shaft. We can obtain rotation speed based upon the incremental encoder by converting the number of pulses per sampling period to the rotation per minute (RPM) below, using equation 6.2.4 by replacing n_m with n_{me}.

$$N = \frac{n_p \times 60}{T \times n_{me}} \ (RPM) \tag{6.2.8}$$

where n_p and T are defined early. Now, consider the case with a sample period of $T = 0.01$ second and $n_p = 50$ pulse counts over the period T. The assembly output shaft speed is calculated below.

$$N = \frac{n_p \times 60}{T \times n_{me}} = \frac{50 \times 60}{0.01 \times 1320} = 227.27 \ (RPM) \tag{6.2.9}$$

The other kind of speed sensor is for detecting linear speed, for example, the vehicle speed of a passage car. We can often calculate linear speed by converting the wheel rotational speed to its linear speed. We multiply the wheel speed by its effective tire radius, assuming that there is no slip (or very small slip) between the tire and the road.

There is also a class of linear speed sensors that measure linear speed directly, for example Doppler and laser speed sensors. A laser surface velocimeter (LSV) [21] is a noncontact optical speed sensor measuring speed and distance on moving surfaces. LSVs apply the laser Doppler principle to evaluate the laser light scattered back from a moving object. They are widely used for manufacturing processes and quality control in industrial production settings. However, according to author John Cimbala, the term Doppler used in laser surface is a misnomer because no Doppler shift measurement is involved in the technique [22]. The detailed principle of an LSV can be found in reference [22] and will not be addressed in detail in this book.

Section Summary

This section discusses rotational speed sensing methods including linear speed, and analog and digital rotational speed sensors. The rotational analog speed sensor can be easily realized by using a DC motor [2]. When the DC motor spins, the armature voltage, which is equal to back-EMF one, is proportional to the rotation speed. A digital speed sensor can be realized by an encoder [3] with the help of so-called hall-effect sensors. In this case, a digital counting device in the information domain is required; the device converts the pulse trains generated by the encoder sensors to actual digital speed signal in terms of rotational direction and speed. The linear speed sensing can often be realized by converting the rotational speed signal to a linear one. For example, vehicle travel speed can be calculated from its wheel rotational speed. One of the direct measurement methods of linear speed is the radar- (or laser-) based linear velocity sensor that emits a radar wave. This wave is reflected by the target surface and returned to the sensor. The speed is then determined by measuring the frequency difference between sent and received signals.

Section Keywords

Analog rotational speed sensing: This sensor produces a continuous, varying analog signal, such as a sine wave or a voltage proportional to speed, to measure the speed of a rotating object.

Hall-effect sensor: This magnetic sensor can be used for detecting the strength and direction of a magnetic field. The field might be produced from either a permanent magnet or an electromagnet with its output varying in proportion to the strength of the magnetic field being detected. Hall-effect sensors are solid state devices. They are becoming increasingly popular because they can be used in many different types of applications such as sensing position, velocity, and directional movement.

Incremental encoder: This linear or rotary electromechanical device has two output signals, A and B, that issue pulses when the device is moved.

Speed calculation using encoder signals: Speed is calculated from encoder signals first by measuring pulses over a set time or by measuring the time between pulses, and then using the encoder's pulses per revolution (PPR) to convert the data to speed.

Practice Problems

1. Let the back-EMF coefficient $K_m = 0.2\ V/(rad/s)$. If the armature voltage is 20 volts, find the motor shaft speed.
2. For the incremental encoder shown in Figure 6.9, draw encoder signals A and B if the magnetic disk rotates in the anticlockwise direction.
3. Derive the speed calculation formula from equation 6.2.4 so that the speed unit is in rad/s.
   a. Find the correct the statement below. The DC-motor gearbox is normally used to
   b. Increase motor output torque and speed.
   c. Decrease motor output torque and speed.
   d. Increase motor output torque and reduce motor output speed.
4. Consider the DC-motor assembly in Example 6.4 with the same front and back gearboxes. Find the assembly output shaft speed in RPM assuming the sample period $T = 0.02$ second and the number of pulses counted over a sample period is 20.

This page titled 6.2: Rotational Speed Sensors and Their Speed Calculation is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6.3: Acceleration Sensors

Learning Objectives

After completing this section, you will be able to:

1. Describe the working principle of a piezoelectric accelerometer.
2. Explain the working principle of a capacitance-based accelerometer.
3. Introduce the MEMS-based (micro-electromechanical systems) accelerometer.
4. Conduct coordinate transformation of the two- and three-dimensional accelerometer signals.

This section discusses how to measure the acceleration of an objector using an acceleration sensor called an accelerometer. Measuring an objector's acceleration is very important. It is a critical physical variable that can be used, for example, for feedback control, calculating other variables (such as a pendulum angle), and measuring the movement of a cell phone or a remote controller. Note that the accelerometer is based upon the measured inertia force $m\ddot{x}$, where m is the sensing mass and $\ddot{x}$ is the acceleration of the mass. The main difference in accelerometer working principles is the way the inertia force was measured. The accelerometer (acceleration sensor) introduced in Section 2.4, an acceleration sensor, is features in Figure 6.12. It uses a piezoelectric force measurement device to obtain the system acceleration.

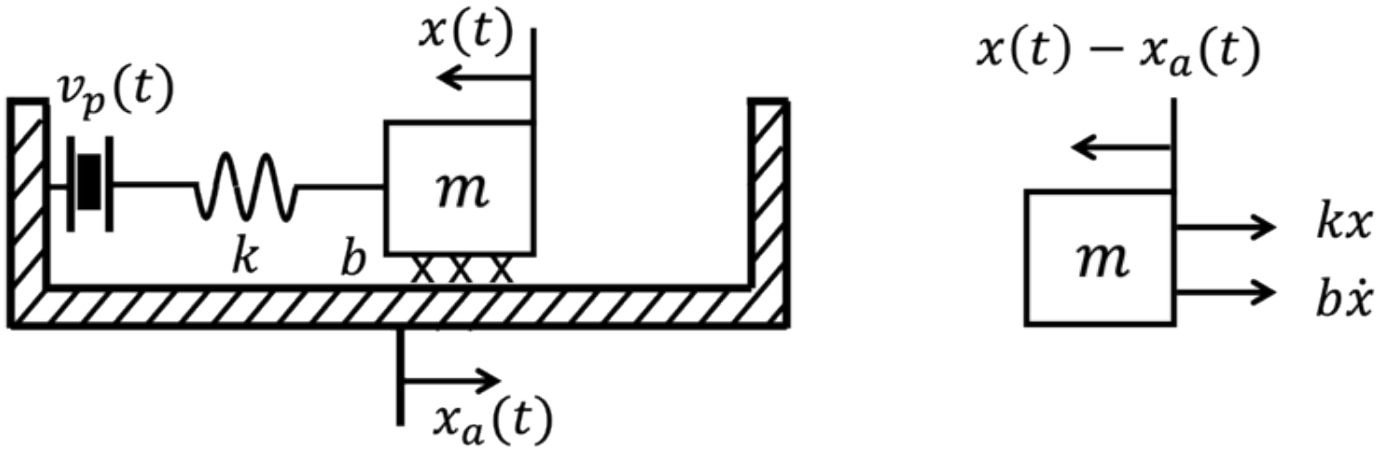


Figure 6.12: Piezoelectric acceleration sensor architecture.

Section 2.4 discussed the transfer function from the sensing element acceleration $\ddot{x}$ to output voltage $V_p(s)$. The Laplace transform for the acceleration $\ddot{x}$ is $s^2X(s)$, where $X(s)$ is the Laplace transform of x. That transfer function was shown in equation ??? and is also shown below.

$$G_{ap}(s) = \frac{V_p(s)}{s^2X(s)} = \frac{V_{p-norm}\omega_{min}s}{s+\omega_{min}} \cdot \frac{\omega_{max}^2\left(ms^2+bs+k\right)/m}{s^2+2\zeta\omega_{max}^2s+\omega_{max}^2} \cdot k \tag{6.3.1}$$

where the parameter definition of the above transfer function can be found in Section 2.4 and reference [23]. The frequency response of the transfer function in equation 6.3.1 was plotted earlier in Figure 2.38. Note that the DC gain of this transfer function is zero because there is a differentiation component s in the numerator of the transfer function 6.3.1. The DC gain can be obtained by setting $s = 0$, which leads to the following.

$$G_{ap}(0) = 0 \tag{6.3.2}$$

That is, for constant acceleration, the output of the piezoelectric acceleration sensor is zero, which is due to the characteristics of the piezoelectric material. Under constant pressure, there is no new energy supplied to the piezoelectric material. As a result, there is no electric piezoelectric energy (signal) output. Therefore, the piezoelectric accelerometer is good for measuring transient acceleration, but not good for measuring steady-state (constant) acceleration. This distinction is very important for selecting a

proper accelerometer for a specific application.

Next, we will discuss sensing technology for capacitance-based acceleration. This sensing technology is also based upon the principle of inertia force. By this principle, a displacement is generated against a spring due to the acceleration of the measuring object. This displacement results in movement of a plate between the two electrodes. Figure 6.13 shows the architecture of such a sensor.

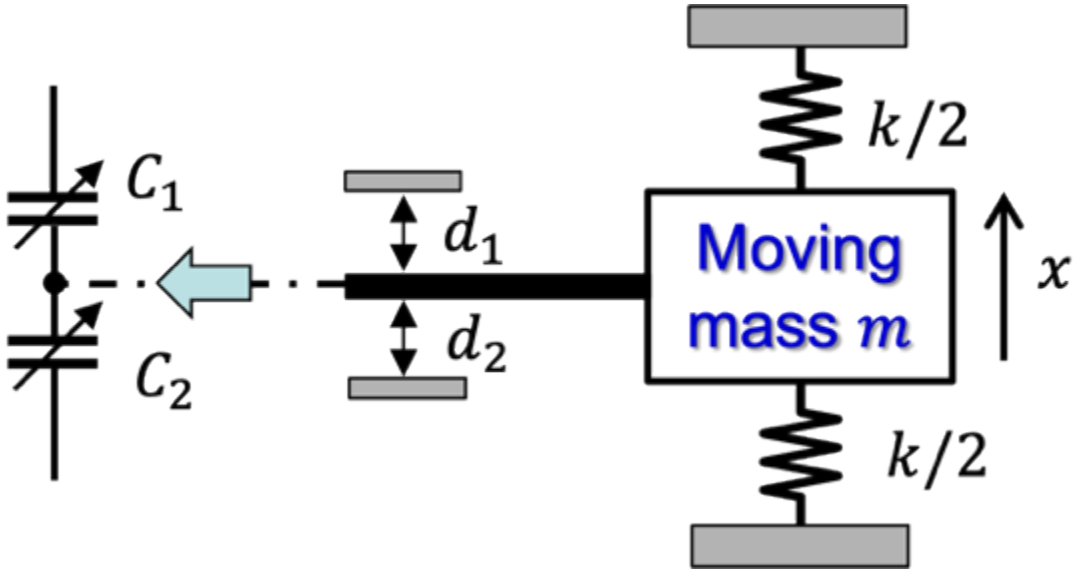


Figure 6.13: Principle of a capacitance-based accelerometer.

Assume that without the acceleration (at rest), the displacements $d_1 = d_2 = d$ and the moving mass displacement $x = 0$. When $x \neq 0$, we have the following.

$$d_1 = d - x, d_2 = d + x \tag{6.3.3}$$

Based upon equation 6.3.3, we can summarize the relationships of displacements (d_1 and d_2) and the capacitance values (C_1 and C_2) in Table 6.2. Note that capacitance values C_1 and C_2 are inversely proportional to the associated distances d_1 and d_2, respectively. That is, when the distance increases, the capacitance reduces.

Note that capacitance is the capacity of a device to store an electric charge (energy). It can be measured by the charge in response to a difference in electric potential, expressed as the ratio of those quantities. There are essentially two closely related capacitances: self-capacitance and mutual capacitance [24]. An object capable of being electrically charged exhibits self-capacitance, for which the electric potential is measured between the object and ground. Mutual capacitance is measured between two components (or plates), which is particularly important for these capacitors in the electronic circuits.

Table 6.2: Displacements (d_1 and d_2) and the associated capacitance values (C_1 and C_2).

$x > 0$	\(d_1	$C_1 > C_2$
0\)" style="text-align:left;" class="lt-eng-137571" data-id="lt-eng-137571">$x = 0$	$d_1 = d_2$	C_2\)" style="text-align:left;" class="lt-eng-137571" data-id="lt-eng-137571">$C_1 = C_2$
0\)" style="text-align:left;" class="lt-eng-137571" data-id="lt-eng-137571">$x < 0$	$d_1 > d_2$	C_2\)" style="text-align:left;" class="lt-eng-137571" data-id="lt-eng-137571">\(C_1

Consider a capacitor with two parallel plates. The capacitance C (dependent on its geometry) is given by the formula below.

$$C = \frac{\rho A}{d} \tag{6.3.4}$$

where A is the area of each plate; d is the distance between the plates; and ρ is the permittivity of the material between the plates of the parallel capacitor, where the permittivity of free space, $\rho = \rho_0$, is a physical constant approximately equal to $\rho_0 = 8.854 \times 10^{-12}$ Farads per meter (F/m). Equation 6.3.4 indicates that the capacitance is inversely proportional to the distance between

the two plates.

Example 6.5

Calculate the capacitance of a two-plate capacitor with distance $d = 0.2\ mm$ and plate area of $2\ mm^2$, with vacuum between the two plates. The capacitance can be calculated using equation 6.3.4 as below with the approximated permittivity of free space for air permittivity.

$$C = \frac{\rho A}{d} = \frac{8.854 \times 10^{-12}\ (F/m)\ \times 2\ (mm^2)}{0.2\ mm} = 8.854 \times 10^{-14}\ F = 0.08854\ pF \tag{6.3.5}$$

For the case indicated in Figure 6.13, the capacitance difference $C_2 - C_1$ can be calculated below based on equation 6.3.3.

$$C_2 - C_1 = \frac{\rho A(d_1 - d_2)}{d_2 d_1} = \frac{-2\rho A x}{(d+x)(d-x)} = \frac{-2\rho A x}{d^2 - x^2} \tag{6.3.6}$$

The relative permittivity of air in a two-plate capacitor is 1.0006 at standard temperature and pressure. However, it can be rounded off to 1.0 for practical purposes, or $\rho = \rho_0$. This is because air is a gas with very few molecules between the capacitor plates. Therefore, it acts like the capacitor plates in a vacuum.

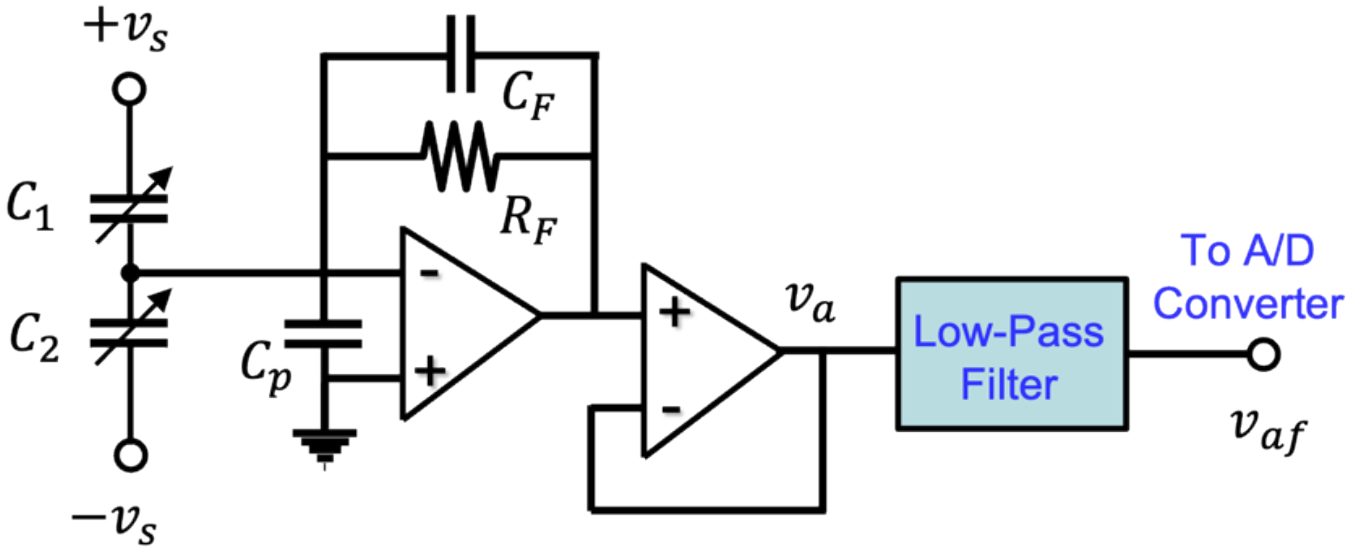


Figure 6.14: Equivalent circuit of a capacitance-based accelerometer.

We can now derive the transfer function for a capacitance-based accelerometer. Figure 6.14 shows an equivalent electronic circuit for acceleration measurement, where C_1 and C_2 are the two capacitors defined in Figure 6.13. Two operational amplifiers are used. One has negative feedback to track the combined inputs v_s and $-v_s$. The other is a following circuit to isolate the sensing circuit from the low-pass filter circuit as an anti-aliasing filter.

Let's next derive the transfer function from $V_s(s)$ to $V_a(s)$, where $V_s(s)$ and $V_a(s)$ are the Laplace transforms of $v_s(t)$ and $v_a(t)$ defined in Figure 6.14, respectively. Using equation ???, we can calculate the transfer function from $+V_s(s)$ to $V_a(s)$ as below.

$$\frac{V_a(s)}{V_s(s)} = -\frac{Z_o(s)}{Z_i(s)} = -\frac{\frac{R_F/C_F s}{R_F + 1/C_F s}}{1/C_1 s} = -\frac{R_F C_1 s}{R_F C_F s + 1} \tag{6.3.7}$$

In equation 6.3.7, $Z_o(s)$ is a parallel connection of resistor R_F and impedance $1/C_F s$, and $Z_i(s)$ is impedance $1/C_1 s$. The transfer function from $-V_s(s)$ to $V_a(s)$ can be found in a similar way as below.

$$\frac{V_a(s)}{-V_s(s)} = -\frac{Z_o(s)}{Z_i(s)} = -\frac{R_F/C_F s}{R_F + 1/C_F s} = -\frac{R_F C_2 s}{R_F C_F s + 1} \tag{6.3.8}$$

In this case, $Z_o(s)$ is the same as before, but $Z_i(s)$ is the impedance $1/C_2 s$ (different in this case). To obtain the complete transfer function, equations 6.3.7 and 6.3.8 can be consolidated. The resulting equation can be found below; it is a linear circuit with two different inputs $V_s(s)$ and $-V_s(s)$ to the operational amplifier.

$$\frac{V_a(s)}{V_s(s)} = \frac{R_F s}{R_F C_F s + 1}(C_2 - C_1) \tag{6.3.9}$$

Power supplied to the capacitance-based accelerometer is constant (say v_s) with its transfer function below.

$$V_s(s) = \frac{v_s}{s} \tag{6.3.10}$$

Then, with measured $V_a(s)$, we can calculate gain $C_2 - C_1$. Equation 6.3.6 can be used to solve for displacement x. Using the following relationship,

$$m\ddot{x} = -xk \tag{6.3.11}$$

acceleration $\ddot{x}$ can be calculated with k to be the spring stiffness defined in Figure 6.13, and m is the value of moving mass.

Now we will address how an accelerometer is realized in the form of a MEMS. Figure 6.15 shows an example of MEMS implementation of a one-directional accelerometer.

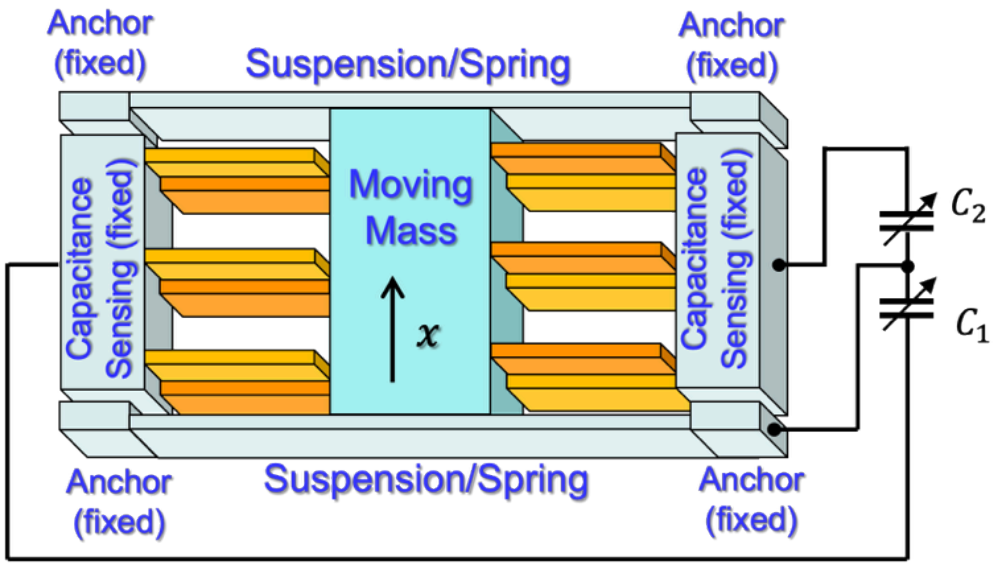


Figure 6.15: Accelerometer MEMS implementation.

From Figure 6.15, we can see that a one-dimensional accelerometer can be implemented in a microchip with a moving part attached with electrode fins as well as two fixed parts attached with electrodes. The top and bottom bars are attached to the moving mass. They act as two springs similar to these in the architecture plot of Figure 6.13. Both ends of these two bars are anchored to the fixed base. The same is true for two sensing elements attached with electrode fins on the left and right sides of plot in Figure 6.13. Figure 6.15 also provides an equivalent circuit to mark both capacitors C_1 and C_2 defined in Figure 6.13. Recall that in Figure 6.13, when the mass moves up, capacitance C_1 increases with reduced d_1, while at the same time, capacitance C_2 decreases with increased d_2.

For the sensing system shown in Figure 6.15, when the mass moves up, capacitance C_1 increases with reduced distance between the two electrodes on the left. At the same time, capacitance C_2 decreases with increased distance between the two electrodes on the right. In most of the MEMS accelerometers, three-dimensional accelerometers are implemented in a single chip (see Figure 6.16). In this case, the one-dimensional accelerometer shown in Figure 6.15 is replicated three times in three orthogonal directions. In addition, the electronic circuit shown in Figure 6.14 is also integrated into the same MEMS ship. Therefore, a MEMS

ship itself is a mechatronic subsystem.

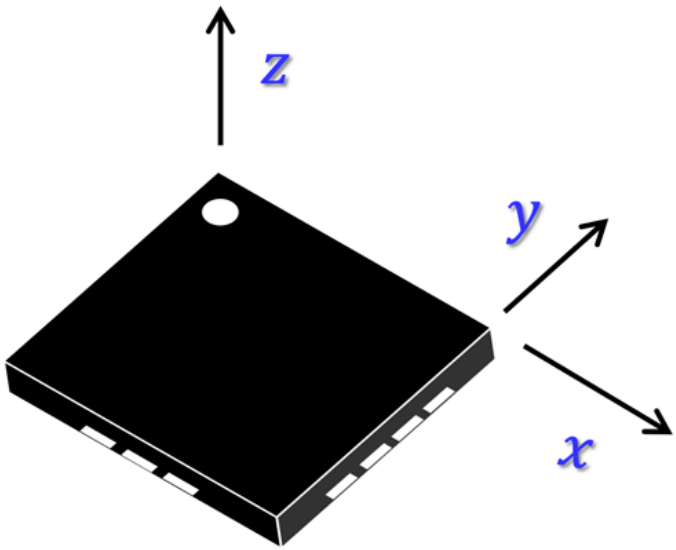


Figure 6.16: A three-dimensional MEMS accelerometer.

The installed sensor directions (body-coordinate: $x,\ y,\ z$ shown in Figure 6.16) are often not in line with the physical coordinates. In this case, coordinate transformation is needed. First, consider the two-dimensional case (x-y plane) with an acceleration vector $\tilde{a}$ shown in Figure 6.17.

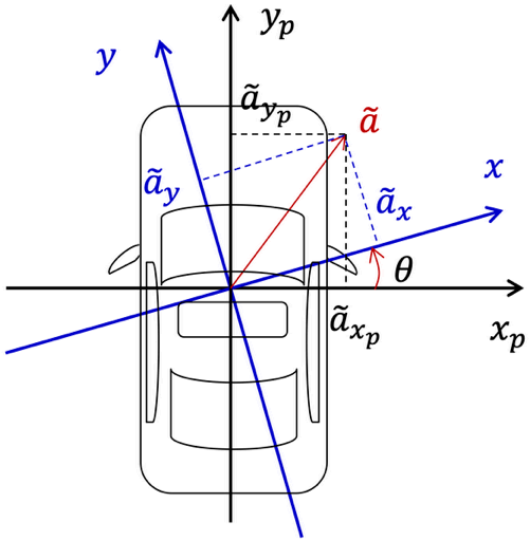


Figure 6.17: Two-dimensional coordinate transformation.

The sensor measures $(\tilde{a}_x, \tilde{a}_y)$ in the sensor coordinate, and $(\tilde{a}_{x_p}, \tilde{a}_{y_p})$ in the physical coordinate (the vehicle coordinate in Figure 6.17) can be calculated by the following equation.

$$\begin{bmatrix} \tilde{a}_{x_p} \\ \tilde{a}_{y_p} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \tilde{a}_x \\ \tilde{a}_y \end{bmatrix}, \tag{6.3.12}$$

where

$$\tilde{a} = \sqrt{\tilde{a}_x^2 + \tilde{a}_y^2} = \sqrt{\tilde{a}_{x_p}^2 + \tilde{a}_{y_p}^2} \tag{6.3.13}$$

Note that θ is positive if the sensor coordinate rotates anticlockwise relative to the physical coordinate and can be either calibrated or obtained through a machine learning process.

For the three-dimensional case, the coordinate transformation can be divided into three sequential rotations: a) around z_p axis (θ), b) y_z axis (ϕ), and c) x_y axis (ψ). Figure 6.18 shows these three rotations.

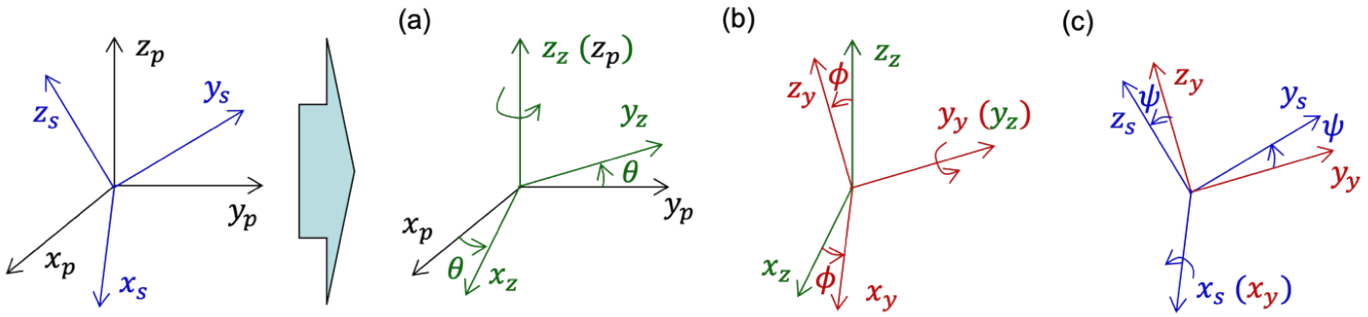


Figure 6.18: Three-dimensional coordinate transformation.

As a result, the coordinate transformation can be factored into three stages below.

$$\begin{bmatrix} \tilde{a}_{x_p} \\ \tilde{a}_{y_p} \\ \tilde{a}_{z_p} \end{bmatrix} = R_{yz} R_{zx} R_{xy} \begin{bmatrix} \tilde{a}_{x_s} \\ \tilde{a}_{y_s} \\ \tilde{a}_{z_s} \end{bmatrix} \tag{6.3.14}$$

with rotation in z_p defined below

$$\begin{bmatrix} \tilde{a}_{x_z} \\ \tilde{a}_{y_z} \\ \tilde{a}_{z_z} \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{a}_{x_p} \\ \tilde{a}_{y_p} \\ \tilde{a}_{z_p} \end{bmatrix} = R_{xy} \begin{bmatrix} \tilde{a}_{x_p} \\ \tilde{a}_{y_p} \\ \tilde{a}_{z_p} \end{bmatrix}, \tag{6.3.15}$$

rotation in y_z defined below

$$\begin{bmatrix} \tilde{a}_{x_y} \\ \tilde{a}_{y_y} \\ \tilde{a}_{z_y} \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} \tilde{a}_{x_z} \\ \tilde{a}_{y_z} \\ \tilde{a}_{z_z} \end{bmatrix} = R_{zx} \begin{bmatrix} \tilde{a}_{x_z} \\ \tilde{a}_{y_z} \\ \tilde{a}_{z_z} \end{bmatrix}, \tag{6.3.16}$$

and rotation in y_z defined below

$$\begin{bmatrix} \tilde{a}_{x_s} \\ \tilde{a}_{y_s} \\ \tilde{a}_{z_s} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \tilde{a}_{x_y} \\ \tilde{a}_{y_y} \\ \tilde{a}_{z_y} \end{bmatrix} = R_{yz} \begin{bmatrix} \tilde{a}_{x_y} \\ \tilde{a}_{y_y} \\ \tilde{a}_{z_y} \end{bmatrix} \tag{6.3.17}$$

Example 6.6

Find the two-dimensional coordinate transformation matrix defined in equation 6.3.11 if the rotational angle $\theta = 30°$. The matrix can be calculated as follows.

$$T_p = \begin{bmatrix} \cos 30° & -\sin 30° \\ \sin 30° & \cos 30° \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 \\ 0.5 & 0.866 \end{bmatrix} \tag{6.3.18}$$

Section Summary

This section addresses sensors that measure an object's acceleration. Note that the piezoelectric-mass accelerometer was modeled in Chapter 2 with its detailed working principle. In this section, we use a capacitive-based accelerometer as an example

and provide detail information regarding the MEMS (micro-electromechanical systems) technology used in these advanced sensors. This technology integrates microscale mechanical sensors into a single silicon chip. As a result, both micromechanical sensors and the required electronics are integrated. Multidirectional accelerometer technology is also discussed. The other method to obtain the acceleration signal is to take the derivative of a speed signal if the required bandwidth is relatively low.

Section Keywords

Practice Questions

1. Which of the following statements is true? Select the correct answer.
   a. Piezoelectric accelerometer can be used to measure steady-state acceleration.
   b. The DC gain of a piezoelectric accelerometer is nonzero.
   c. Piezoelectric accelerometers can only be used for measuring object transient acceleration.
2. Calculate the capacitance of a two-plate capacitor with distance $d = 0.1\ mm$, plate area of $4\ mm^2$, and air-filled gap between the two plates. Note that the capacitance can be calculated using equation 6.3.4 with the approximated permittivity of free space for air permittivity.
3. What does MEMS mean? Select the correct answer.
   a. Memory device
   b. Micro-electromechanical system
   c. Mechanical-electrical machining system
4. Find the two-dimensional coordinate transformation matrix defined in equation 6.3.11 if the rotational angle $\theta = 60^\circ$. Note that it can be calculated using the following equation.

$$T_p = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

This page titled 6.3: Acceleration Sensors is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6.4: Angular Velocity Sensors (Rate Gyro)

Learning Objectives

After completing this section, you will be able to:

1. Describe the basic working principle of a rate gyro sensor.
2. Explain the process of implementing a mechanical rate gyro sensor into a MEMS device.
3. Provide the basic working principle of a piezoelectric rate gyro sensor.

This section discusses a family of angular velocity sensors (also called rate gyro) used to measure the angular velocity of a given axis. In general, the gyroscope is an inertial sensor that is used to measurement or control the orientation and rotational velocity of a body. A rate gyro is a type of gyroscope. Instead of indicating direction, it measures the angular rate of change with time. If a gyroscope has only one gimbal ring, with consequently only one degree of freedom, it can be adapted for use as a rate gyro to measure a rate of angular movement. That is, a gyroscope measures multidimensional angular velocity, whereas a rate gyro is for one direction. There are mainly two types of rate gyros: mechanical sensors (e.g., piezoelectric, MEMS) [26] and optical laser gyro sensors [6], [25]. Mechanical rate gyros are based on the principle of Coriolis force [27], defined below. In addition, both piezoelectric and MEMS-based rate gyro sensors are vibration-based, to be explained below.

Definition 6.2

In physics, the Coriolis force is defined as an inertia (or fictitious) force that applies to objects in motion within a reference coordinate (frame) that rotates with respect to the inertia coordinate. For example, Figure 6.19 shows a reference coordinate with anticlockwise rotation, where the force acts to the left of the motion of the object.

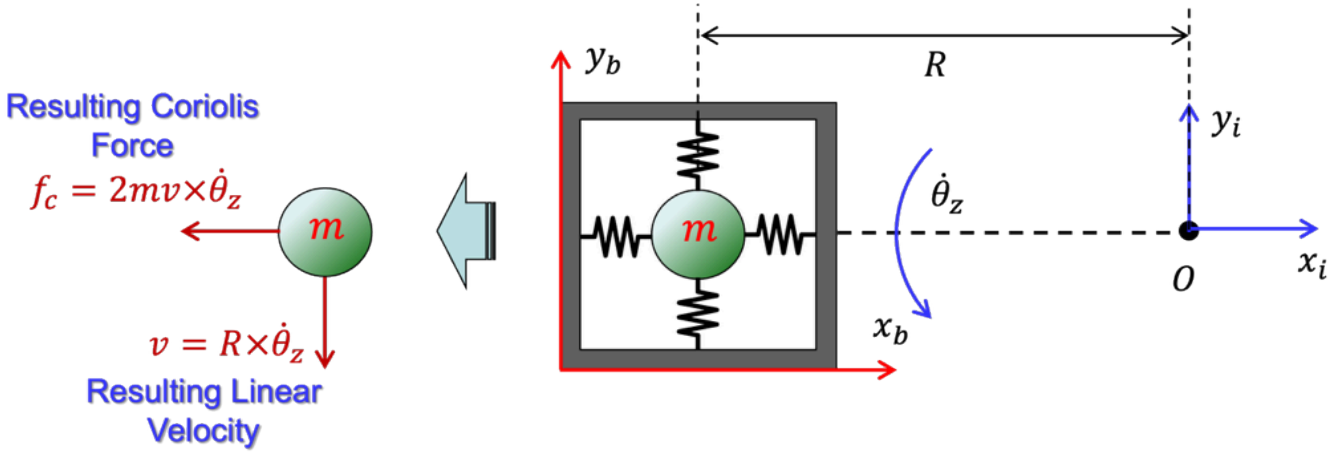


Figure 6.19: A proof mass fixed in the reference coordinate with respect to the inertia coordinate.

Consider the reference coordinate, also called the body coordinate, denoted by x_b and y_b and the inertia coordinate denoted by x_i and y_i on a two-dimensional plane.

In Figure 6.19, the reference frame (coordinate) is represented by the gray box with a proof mass m suspended by four springs. Note that a proof mass is a known, suspended, or inertial mass used in sensors to measure external acceleration or vibration by deflecting relative to a housing. The reference coordinate rotates around origin O in the inertia frame (coordinate x_i and y_i) with an angular speed of $\dot{\theta}_z$. This results in a downward linear speed of $v = R\dot{\theta}_z$. Based upon the Coriolis force principle, the resulting Coriolis force is shown below.

$$f_c = 2mv\dot{\theta}_z \tag{6.4.1}$$

From equation 6.4.1, we can see that the reference frame rotational speed $\dot{\theta}_z$ can be found if both the Coriolis force f_c and linear velocity $v = R\dot{\theta}_z$ can be measured. The Coriolis force can be measured by the displacement of mass m due to two horizontal springs and applied Coriolis force f_c. However, the linear speed is difficult to measure under constant speed, especially when it is combined with the linear motion velocity of the reference frame (that is, $v \neq R\dot{\theta}_z$), leading to the following concept of vibration-based rate gyros.

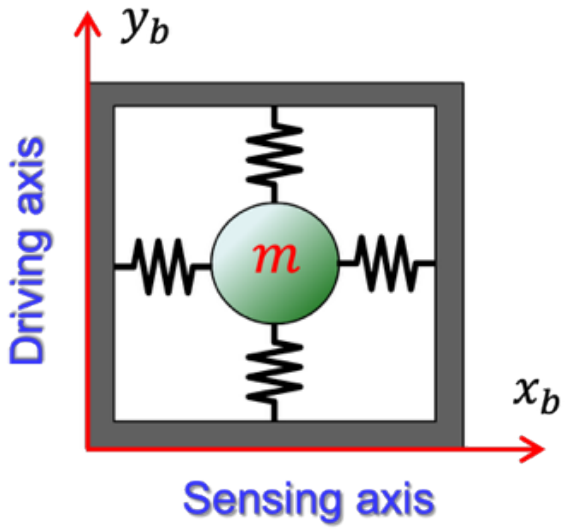


Figure 6.20: Proof mass and axis definitions.

Most single-axis vibrating rate gyros have a proof mass m (same as that in Figure 6.19) with two orthogonal degrees of freedoms (vertical and horizontal); see Figure 6.20. Note that the vertical axis (y_b) is the driving axis to create a vibration motion in that direction. The horizontal axis (x_b) is the sensing axis to detect the Coriolis force. When the rate gyro continuously vibrates at a resonant frequency in one direction (y_b), the oscillation remains absent in the orthogonal (horizontal) direction (x_b) until the external rotation has occurred. When the external rotation occurs, the rotation rate $\dot{\theta}_z$ triggers energy shifting from the vertical vibration towards the horizontal direction due to the Coriolis effect [27].

The two degrees of freedom equations of motion for vibrating rate gyros based upon the Newton and Coriolis' principles are shown in equations 6.4.2 to 6.4.4.

$$m\frac{d^2y_b}{dt^2} + c\frac{dy_b}{dt} + ky_b = f_d + 2m\dot{\theta}_z\frac{dx_b}{dt} \tag{6.4.2}$$

For equation 6.4.2, m is the proof mass, k is the spring stiffness in y_b direction shown in Figure 6.20, c is the friction coefficient of the proof mass, and f_d is the driving force used to create the oscillation motion of proof mass. The response due to rotation $\dot{\theta}_z$ $\left(2m\dot{\theta}_z\left(dx_b/dt\right)\right)$ is very small in magnitude compared to the driving response due to a relatively large driving force f_d. As a result, the Coriolis force in the vertical direction (y_b) can be ignored. Equation 6.4.2 can then be rewritten as follows.

$$m\frac{d^2y_b}{dt^2} + c\frac{dy_b}{dt} + ky_b = f_d \tag{6.4.3}$$

In turn, the equation of motion in the horizontal direction can be expressed as follows.

$$m\frac{d^2x_b}{dt^2} + c\frac{dx_b}{dt} + kx_b = -2m\dot{\theta}_z\frac{dy_b}{dt} \tag{6.4.4}$$

where it is assumed that the spring stiffness k and friction coefficient in both x_b and y_b directions are the same. Furthermore, $2m\dot{\theta}_z\left(dy_b/dt\right)$ is the Coriolis force applied in the x_b direction due to rotation $\dot{\theta}_z$ in the z direction (see Figure 6.19). The free body diagram of the proof mass can be found in Figure 6.21, where $f_c = 2m\dot{\theta}_z\left(dy_b/dt\right)$.

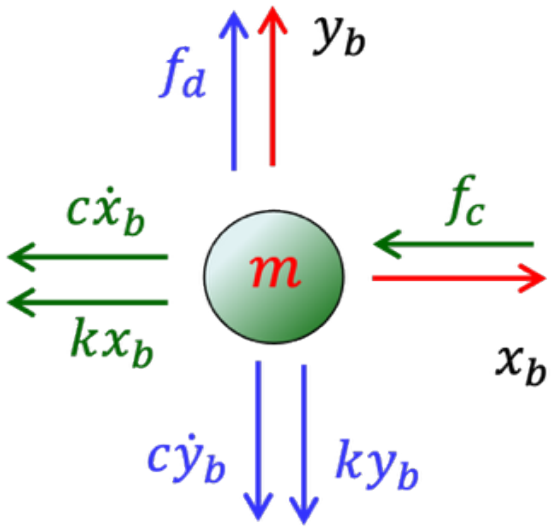


Figure 6.21: Free body diagram of the rate gyro proof mass.

Next, assume that a closed-loop control system is employed so that the resulting oscillation in the vertical direction is a cosine signal, as below with magnitude m_y and frequency ω_b.

$$y_b = m_y \cos\omega_b t \quad \rightarrow \quad \dot{y}_b = -m_y \omega_b \sin\omega_b t \tag{6.4.5}$$

If ω_b is way greater than the maximum frequency component of $\dot{\theta}_z$, we can consider $\dot{\theta}_z$ as a constant because $\dot{\theta}_z$ is much slower than ω_b. Recall that Theorem 3.2 (Section 3.2) states that for a sinusoidal input $A\sin\omega t$, the steady-state output of a stable system described by transfer function $G(s)$ is in the form of $A\,|G(j\omega)|\sin\left(\omega t + \angle G(j\omega)\right)$. We can derive the transfer function of the sensing system in Figure 6.20 described by equation 6.4.4, resulting in the following equation, assuming that $\dot{\theta}_z$ is a constant (much slower than ω_b).

$$G_{rg}(s) = \frac{X_b(s)}{U_b(s)} = \frac{2mm_y\omega_b\dot{\theta}_z}{ms^2 + cs + k} \tag{6.4.6}$$

where $X_b(s)$ is the Laplace transform of $x_b(t)$, and $U_b(s)$ is the Laplace transform of the oscillation signal $\sin\omega_b t$. Again, using Theorem 3.2 yields the following steady-state response of x_b.

$$x_{b-ss}(t) = \frac{2mm_y\omega_b\dot{\theta}_z}{k}\sin\left(\omega_b t + \angle G_{rg}(j\omega)\right) \tag{6.4.7}$$

If the magnitude of the steady-state response, $2mm_y\omega_b\dot{\theta}_z/k$, can be detected, the angular velocity $\dot{\theta}_z$ can be obtained from the detected magnitude.

Example 6.7

Let a rate gyro proof mass $m = 0.05\ kg$, sinusoid vibration displacement signal $y_b(t) = -10^{-5}\cos 2\pi 10^5 t$, and rotational speed $\dot{\theta}_z = 10\ rad/s$. Find the Coriolis force. In this case, we can find the Coriolis force using the following equation with $m_y = 10^{-5}$ and $\omega_b = 2\pi 10^5$.

$$f_c = 2m\dot{\theta}_z\frac{dy_b}{dt} = 2m\dot{\theta}_z m_y\omega_b \sin\omega_b t \tag{6.4.8}$$

Substituting known parameters, we have the following.

$$f_c = 2 \times 0.005 \times 10 \times 10^{-5} \times 2\pi \times 10^5 \sin 2\pi 10^5 t = 0.2\pi \sin 2\pi 10^5 t (Nm) \tag{6.4.9}$$

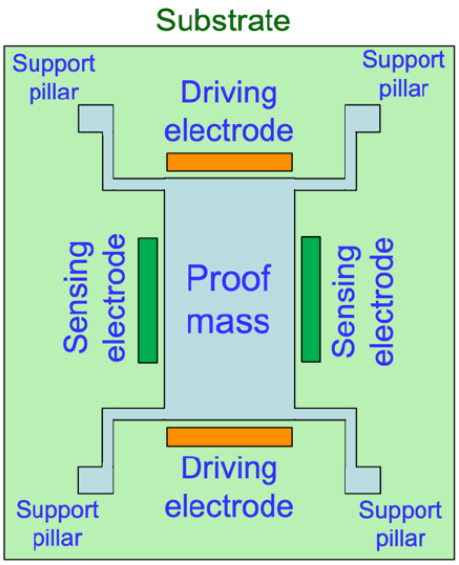


Figure 6.22: A MEMS based rate gyro sensor.

There are two widely used implementations of the rate gyro sensors based on vibration principle. The first one is MEMS based. Figure 6.22 shows the simplest structure of a MEMS vibration rate gyro sensor. Here, a proof mass is placed on top of the substrate representing the reference frame. Four support pillars are fixed on to the substrate through four suspension beams to generate the spring forces in two directions. Both top-bottom driving electrodes (orange) and side sensing electrodes (green) are fixed to the substrate. The driving electrodes apply force to the proof mass, causing the proof mass to vibrate vertically without angular movement. In addition, the angular movement of the substrate leads to horizontal vibrations due to the generated Coriolis force. Therefore, the capacitance-based sensing electrodes can measure the vibration magnitude and convert it to the angular speed information. Note that the working principle of the vibration rate gyro was addressed earlier.

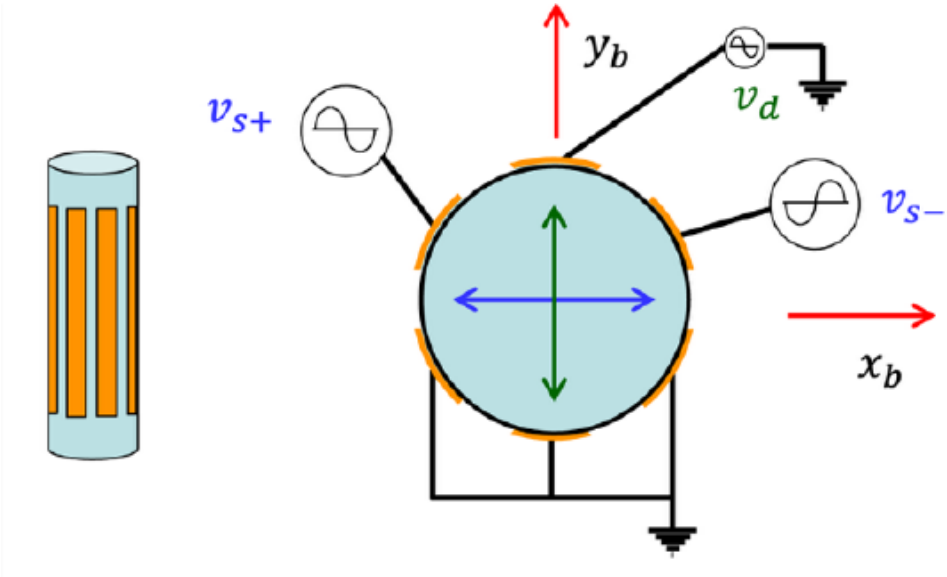


Figure 6.23: A piezoelectric rate gyro sensor.

Piezoelectric rate gyros are sensors using piezoelectric oscillators to capture the rotational movements of objects; see Figure 6.23. They are widely used for compensating hand shaking in video and digital still cameras. The principle of piezoelectric rate gyros (gyroscopes) is the same as that of the Foucault pendulum. The right plot of Figure 6.23 shows a piezoelectric element vibrating up and down (green arrow in y_b direction) in the middle section of the piezoelectric rod (see left plot). This is due to applied AC drive voltage v_d to the upper and lower electrodes (see the right drawing).

When there is no angular velocity $\dot{\theta}_z$ in the z direction, the applied voltage only causes the up and down vibration. However, when the angular velocity $\dot{\theta}_z$ is present, the Coriolis force is generated and applied to the middle section of the piezoelectric rod,

causing the oscillation horizontally (blue arrow in the x_b direction in the right plot). When the rod oscillates in the x_b direction, AC voltages (v_{s+} and v_{s-}) are generated, where the magnitude of AC voltages (v_{s+} and v_{s-}) are proportional to the angular velocity. In summary, the principle of a piezoelectric rate gyro is the same as that of a MEMS rate gyro. The only difference is how the electric signal is collected. See reference [28] for a detailed review of piezoelectric rate gyros.

A ring laser gyroscope (RLG) [29] consists of a ring laser having two independent counter-propagating resonant modes over the same path [30]. The difference in phase is used to detect rotational speed. It operates based on the principle of the Sagnac effect [31] which shifts the nulls of the internal standing wave pattern in response to angular rotation. Interference between the counter-propagating beams (frequency difference between the two beams) can be observed externally, resulting in motion of the standing wave pattern used to calculate rotational speed.

Section Summary

This section discusses rate gyro (angular velocity) sensors. In general, gyro angular speed sensors can be divided into mechanical (including vibration) [5] and optical [6]. Mechanical sensors were reviewed in detail. The vibration rate gyro sensor technology is discussed in detail from principle to associated sensor hardware. Its sample implementation into MEMS will be introduced in Section 6.6. The piezoelectric vibration rate gyro is also reviewed briefly, and the principle of optical (laser) gyro sensors is briefly introduced with provided references. To form a three-dimensional rate gyro, three single-axis rate gyro sensors are used in three orthogonal directions.

Section Keywords

Practice Problems

1. The rate gyros can be divided into which of the following two categories? Select the correct answer.
   a. Mechanical and optical
   b. Piezoelectric and MEMS
   c. MEMS and optical
2. Let a rate gyro proof mass $m = 0.025\ kg$, cosine vibration displacement signal $y_b(t) = -10^{-5}\cos 2\pi^5 t$, and rotational speed $\dot{\theta}_z = 10\ rad/s$. Find Coriolis force f_c.
3. Select True or False for the following statements.
   a. The piezoelectric rate gyro is based upon the Coriolis force principle.
   b. The piezoelectric rate gyro generates the sensing signal (voltage) by vibrations caused by Coriolis force due to angular movement.
   c. If there is no AC voltage v_d applied (see Figure 6.23), the piezoelectric rate gyros can still sense the angular motion.
4. Laser rate gyro sensors are based on which of the following principles? Select the correct answer.
   a. Sagnac effect
   b. Ring laser

This page titled 6.4: Angular Velocity Sensors (Rate Gyro) is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6.5: Sensor Fusion

Learning Objectives

After completing this section, you will be able to:

1. Describe the basic working concept of sensor fusion.
2. Calculate the tilt-angular displacement signal by using sensor fusion of accelerometer and rate gyro signals.
3. Explain the effect of sensor fusion based on the Kalman filter technique.

This section discusses how to use sensor fusion technology to improve measured signal accuracy. Sensor fusion is a technology combining multiple sensor signals. It helps us obtain one or more estimates that are more accurate than those from the original sensor signals or estimates that cannot be measured directly. Figure 6.24 provides an illustration.

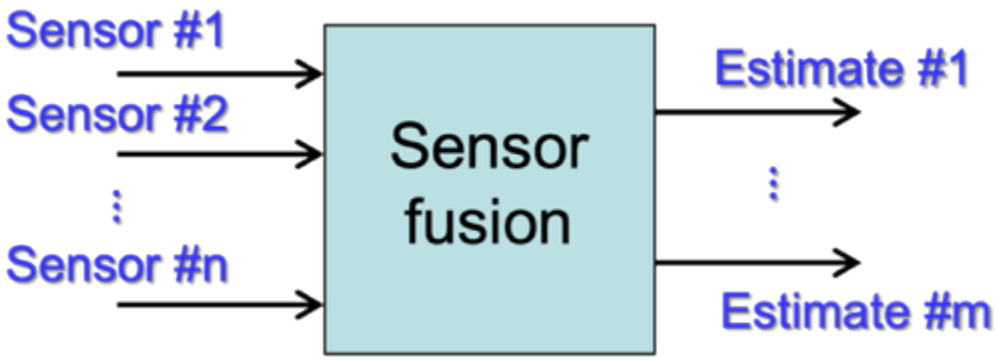


Figure 6.24: Sensor fusion general system architecture.

As Figure 6.24 shows, the sensor fusion technology utilizes multiple sensor signals (in this case, n signals) to construct m multiple estimates. Example 6.8 demonstrates how to use sensor fusion to reduce the signal noise. Let z_1 x1and z_2 x2 denote two measured signals from two independent sensor measurements with noise variances $\sigma_1^2\sigma$12 and $\sigma_2^2\sigma$22, respectively. One possible way to obtain a combined (fused) estimate x3z_3 is to apply inverse-variance weighting, employed within the Fraser-Potter fixed-interval smoother [32], as follows. Assume x3=σ32(σ1−2x1+σ2−2x2).

$$z_3 = \sigma_3^2\left(\sigma_1^{-2}z_1 + \sigma_2^{-2}z_2\right) \tag{6.5.1}$$

where $\sigma_3^2 = \left(\sigma_1^{-2} + \sigma_2^{-2}\right)^{-1}$ is the variance of the combined estimate z_3. We can see that the fused estimate is simply a linear combination of the two measurements weighted by their respective variance information. The following example demonstrates how much noise variance can be reduced for the fused signal.

Example 6.8

Let z_1x1 and z_2x2 be two measured signals from two independent sensor with noise variances $\sigma_1^2 = 1\sigma$12 and $\sigma_2^2 = 2\sigma$22, respectively. Calculate the variance of fused signal $z_3 = \sigma_3^2\left(\sigma_1^{-2}z_1 + \sigma_2^{-2}z_2\right)$. The fused signal variance can be calculated as below based on [32].

$$\sigma_3^2 = \left(\sigma_1^{-2} + \sigma_2^{-2}\right)^{-1} = \left(1^{-1} + 2^{-1}\right)^{-1} = 1.5^{-1} = 0.667 \tag{6.5.2}$$

After fusion, the variance of estimate z_3 is significantly reduced and is smaller than both variances of z_1 and z_2.

Remark 6.2

From the fused signal variance $\sigma_3^2 = \left(\sigma_1^{-2} + \sigma_2^{-2}\right)^{-1}$, we can see that the fused signal variance is always smaller than the individual one. Therefore, one simple way to reduce sensor measurement noise variance is to use two independent sensors with the same noise characteristics ($\sigma_1^2 = \sigma 12 \sigma_2^2$) and fuse the two signals z_1x1 and z_2x2 based on equation 6.5.1. As a result, the fused estimate z_3 has its variance $\sigma_3^2 = 0.5\sigma_1^2 = \sigma 1200.5\sigma_2^2$. That is, the noise variance can be cut by half.

The Kalman filter [33], [34] can also be used for fusion of multiple sensor measurements. Let's demonstrate this with a simple example. Consider the following first-order state space dynamic system (see Section 3.5).

$$\begin{aligned} \dot{x} &= Ax + Dw \\ y &= Cx \\ z &= Cx + v \end{aligned} \tag{6.5.3}$$

where $A = 0.5$, $D = 1$, and $C^T = [1,\ 1]$. Variables x, $y = [y_1,\ y_2]^T$, and $z = [z_1,\ z_2]^T$ are state, output, and measurement vectors, respectively. Because the system has an order of one, we have a redundant output $y_1 = y_2$ due to matrix C. However, measurements z_1 and z_2 are not redundant because it is assumed that their measurement noises are independent. For this example, input w is system noise with its noise intensity $W = 0$; and $v = [v_1,\ v_2]^T$ is a noise vector with its covariance matrix V defined below.

$$V = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} > 0 \tag{6.5.4}$$

where $\sigma_1 \neq \sigma_2$. Note that based on the Kalman filter design principle, a Kalman filter (state estimator) can be obtained in the following form

$$\dot{\hat{x}} = A\hat{x} + L\left(z - C\hat{x}\right) = A\hat{x} + LC\left(x - \hat{x}\right) + Lv \tag{6.5.5}$$

where $L = \tilde{X}C^T V^{-1}$ is the estimation gain (see Section 3.5) and $\tilde{X} \geq 0$ is the covariance matrix of the state estimation error vector $\tilde{x} = x - \hat{x}$ that can be solved by the following Riccati equation for a positive semidefinite solution.

$$0 = A\tilde{X} + \tilde{X}A^T - \tilde{X}C^T V^{-1} C\tilde{X} + DWD^T \tag{6.5.6}$$

Substituting system matrices and associated covariance matrices, we have the following with $W = 0$.

$$0 = A\tilde{X} + \tilde{X}A^T - \tilde{X}C^T V^{-1} C\tilde{X} = \tilde{X}\left(1 - \left(\sigma_1^{-2} + \sigma_2^{-2}\right)\tilde{X}\right) \tag{6.5.7}$$

Combining state equations 6.5.3 and 6.5.5 leads to the following error state equation for the estimation below.

$$\dot{\tilde{x}} = (A - LC)\,\tilde{x} - Lv \tag{6.5.8}$$

Because A is open-loop unstable, a nontrivial solution of equation 6.5.7 is required to stabilize the estimator 6.5.8. As a result,

$$\tilde{X} = \left(\sigma_1^{-2} + \sigma_2^{-2}\right)^{-1} \tag{6.5.9}$$

And the estimation gain L can be found as below.

$$L = \left(\sigma_1^{-2} + \sigma_2^{-2}\right)^{-1} \begin{bmatrix} \sigma_1^{-2} & \sigma_2^{-2} \end{bmatrix} \tag{6.5.10}$$

With the above equations derived for the error covariance and estimation gain in equations 6.5.9 and 6.5.10, we can make the following conclusions in Remark 6.3. Lk = [σ22Pkσ22Pk+σ12Pk+σ12σ22σ12Pkσ22Pk+σ12Pk+σ12σ22].

Remark 6.3

From the above derivation, we can conclude the following.

1. The Kalman filter can be used as a method for sensor fusion.
2. With two sensors measuring the same physical variable, but with difference measurement noise variances, the fusion based on the Kalman filter could reduce the error variance of the fused estimate, due to equation 6.5.9.
3. Equation 6.5.10 shows that the Kalman filter estimation gain L is dependent on the sensor variances σ_1 and σ_2. The sensor with smaller noise variance than the other will be weighted more in the estimation. In other words, the Kalman estimation in this case relies more on the sensor signal with smaller variance.

Example 6.9 shows that the system matrix A could significantly impact the variance of the Kalman estimate.

Example 6.9

Consider system 6.5.3 with $A = 0.7$, $D = 1$, and $C^T = [1,\ 1]$. The input system covariance $W = 0$ and the measurement noise covariance matrix $V = diag\left[\sigma_1^2, \sigma_2^2\right]$. Let $\sigma_1^2 = 1\sigma 12$ and $\sigma_2^2 = 1.5\sigma 22$, respectively. Calculate the variance of fused signal by the Kalman filter defined in equations 6.5.5, 6.5.6, and 6.5.10. Note that the fused signal variance $\tilde{X}$ can be obtained by solving the Riccati equation 6.5.6, as below.

$$\tilde{X} = 1.4\left(\sigma_1^{-2} + \sigma_2^{-2}\right)^{-1} = 1.4\left(1^{-1} + 1.5^{-1}\right)^{-1} = 0.969 \tag{6.5.11}$$

After sensor fusion, the variance of the fused estimate is reduced moderately in this case. However, if $A = 0.5$,

$$\tilde{X} = 1.0\left(\sigma_1^{-2} + \sigma_2^{-2}\right)^{-1} = 1.0\left(1^{-1} + 1.5^{-1}\right)^{-1} = 0.692 \tag{6.5.12}$$

In this case, the Kalman filter is able to reduce the fused signal variance by more than 30%.

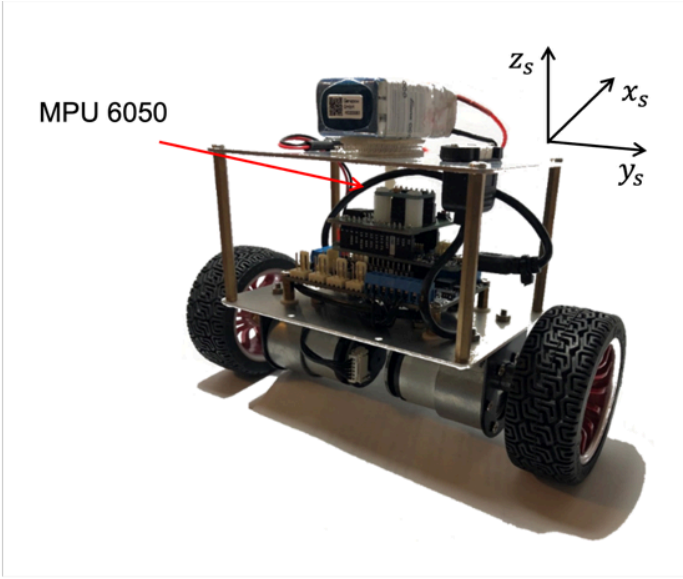


Figure 6.25: A mini-Segway prototype.

Now, we are ready to study another kind of sensor fusion, where the physical variable cannot be measured directly. Consider the mini-Segway in Figure 6.25 (see Chapter 8 for a more complete discussion). The mini-Segway is essentially an inversed pendulum driven by the two-wheel cart. To keep the mini-Segway in the upright position, we need to measure the pendulum angular position to help with feedback control, even when Segway wheels are moving. However, the mini-Segway has no fixed reference point. Therefore, it is impossible to install a physical angular position sensor to measure it.

One approach is to calculate the angular displacement θ_y^g of the inversed pendulum from the rate gyro signal, where the rate gyro is installed on the top of the pendulum in direction y_s parallel to the wheel driving axis. Angular displacement θ_y^g can be calculated from the gyro output (angular velocity $\dot{\theta}_y^g$) by integrating $\dot{\theta}_y^g$ in a digital sense with a sample time of T_s. Consider a Taylor expansion of the angular displacement θ_y^g as follows.

$$\theta_y^g(t+T_s)=\theta_y^g(t)+\frac{d\theta_y^g(t)}{dt}\cdot T_s+\varepsilon(t)=\theta_y^g(t)+\dot{\theta}_y^g(t)T_s+\varepsilon(t) \tag{6.5.13}$$

where $\varepsilon(t)$ represents the high order term in the Taylor expansion. Let $t=kT_s$ and, ignoring the high order term $\varepsilon(t)$, we have the following.

$$\theta_y^g((k+1)T_s)\approx\theta_y^g(kT_s)+\dot{\theta}_y^g(kT_s)T_s \tag{6.5.14}$$

Letting $\omega_y^g=\dot{\theta}_y^g$ be the rate gyro output and applying the z-transform to equation 6.5.14 yields the following.

$$\frac{\Theta_y^g(z)}{\Omega_y^g(z)}=\frac{T_s}{z-1}=G(z) \tag{6.5.15}$$

where $\Theta_y^g(z)$ and $\Omega_y^g(z)$ are z-transforms of θ_y and $\dot{\theta}_y$, respectively. From Table 4.3 in Section 4.3, we can find that transfer function $G(z)$ is a one-step delayed integration. Therefore, we can calculate the angular position by integrating the sampled angular speed signal. However, the rate gyro signal actually contains stochastic noise and bias that cannot be calibrated out because it varies as the sensor operational condition changes, such as temperature. Let ω_n and ω_b represent the sensor noise and bias, respectively. Equation 6.5.14 becomes as follows.

$$\theta_y^g((k+1)T_s)\approx\theta_y^g(kT_s)+\omega_y^g(kT_s)T_s+\omega_n(kT_s)T_s+\omega_b(kT_s)T_s \tag{6.5.16}$$

The associated z-transform is shown below.

$$\Theta_y^g(z)=\frac{T_s}{z-1}=G(z)\left(\Omega_y^{gc}(z)+\Omega_n(z)+\Omega_b(z)\right) \tag{6.5.17}$$

where $\Omega_n(z)$ and $\Omega_b(z)$ are the z-transform of ω_n and ω_b, respectively, and $\Omega_y^{gc}(z)$ is the z-transform of $\dot{\theta}_y^g$ without any noise. Assuming that ω_n is a zero-mean noise over the integration period, term $\theta_n(k)=Z^{-1}[G(z)\Omega_n(z)]$ is bounded and goes to zero as k goes to infinity. However, assuming that ω_b is a constant bias and letting $\theta_b(k)=Z^{-1}[G(z)\Omega_b(z)]$, we have the associated time response of $\theta_b(k)$ with bias input as follows, with $\theta_b(-1)=0$.

$$\theta_b(k)=\sum_{i=0}^{k-1}\omega_bT_s=k\omega_bT_s \tag{6.5.18}$$

Note that the term $k\omega_bT_s$ increases as the time index k increases, causing angular displacement drift due to the accumulated bias error. As a result, angular displacement calculated from the rate gyro signal is not accurate over time; a new method is needed. Sensor fusion could be a good choice by combining rate gyro and accelerometer sensor signals.

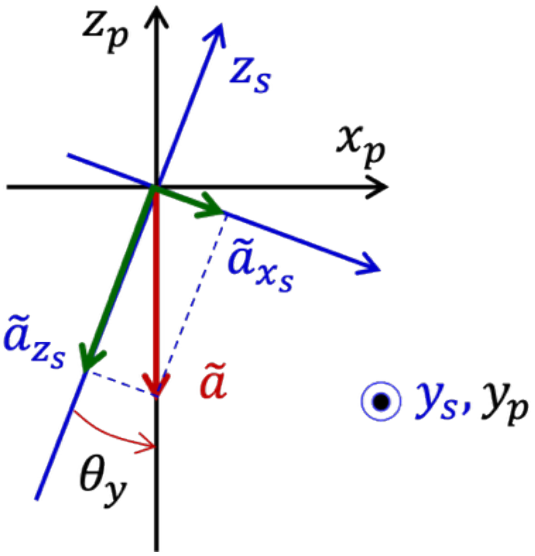


Figure 6.26: Sensor and physical coordinates.

The question then is how to calculate the mini-Segway pitch angle using accelerometer signals. Let $(x_s,\ y_s,\ z_s)$ and $(x_p,\ y_p,\ z_p)$

be the sensor and physical coordinates in Figure 6.26, respectively. The definitions of y_s and y_p are shown on Figure 6.26 with a direction out of the plane (arrow dot). The tilt angle of the mini-Segway is also defined, with anticlockwise rotation as positive.

We can assume that the acceleration magnitude of the mini-Segway cart in the x_p direction is much smaller than that of gravity ($-\tilde{a}$ in the physical coordinate). This is true for a cart like mini-Segway with limited actuation capability. In this case, the gravity vector can be factorized in the two components below.

$$\tilde{a}_{z_s} = -\tilde{a}\cos\theta_y \ , \quad \tilde{a}_{x_s} = -\tilde{a}\sin\theta_y \tag{6.5.19}$$

Because the rotation angle defined in Figure 6.26 is clockwise, θ_y is negative in this case. As a result, we have the following.

$$\tan\theta_y \ = \frac{\tilde{a}_{x_s}}{\tilde{a}_{z_s}} \qquad \rightarrow \theta_y^a = \theta_y = \operatorname{atan}\left(\frac{\tilde{a}_{x_s}}{\tilde{a}_{z_s}}\right) \tag{6.5.20}$$

Therefore, the mini-Segway pendulum angular displacement θ_y can be calculated using the accelerometer installed on the mini-Segway; see Figure 6.25 for the three-axis accelerometer and rate gyro (MPU 6050) located at the tip of the pendulum. With the angle defined in Figure 6.26, the calculated angle is negative because measured $\tilde{a}_{x_s}$ is positive and $\tilde{a}_{z_s}$ is negative. We can summarize the pros and cons of this method in Remark 6.4.

Remark 6.4

From the above derivation, we can conclude the following.

1. One key advantage of using the accelerometer signal to calculate angular displacement is that it does not use integration. Thus, the fused signal will not drift.
2. However, accelerometer signals are subject to high-frequency noise, making it inaccurate at high frequency. Therefore, the angular displacement calculation only works well at low frequency.

As Remark 6.4 summarizes, the accelerometer signals are good only for estimating angular displacements at low frequency. If a high-pass filter is used before integrating the rate gyro signal, the bias in the rate gyro signal can be eliminated to generate a good angular displacement signal at high frequency. The idea now is to combine the angular displacements calculated from accelerometer signals at low frequency and from rate gyro signal at high frequency using a so-called complementary filter [35] of both accelerometer and rate gyro signals. A complementary filter is a sensor fusion technique that uses high- and low-pass filter to combine data from multiple sensors, such as an accelerometer and gyroscope, to generate an estimate.

Let $\theta_y^a(kT_s) = \operatorname{atan}\left(\tilde{a}_{x_s}(kT_s)/\tilde{a}_{z_s}(kT_s)\right)$ be the discrete-time angular displacement calculated from the accelerometer signals ($\tilde{a}_{x_s}$ and $\tilde{a}_{z_s}$). Also let $\theta_y^g(kT_s) = Z^{-1}\left[\Theta_y^g(z)\right]$ denote the angular displacement calculated by integrating the rate gyro signal. The z -transforms of $\Theta_y^g(z)$ is shown below based on equation 6.5.15.

$$\Theta_y^g(z) = \frac{T_s}{z-1}\Omega_g(z) \tag{6.5.21}$$

Note that $\theta_y^a(z)$, $\Theta_y^g(z)$, and $\Omega_g(z)$ are associated z-transforms of θ_y^a, θ_y^g, and $\dot{\theta}_y^g$ (rate gyro measurement), respectively. Then, the sensor fusion based on the complimentary filter leads to the following sensor fusion transfer function.

$$\Theta_y^c(z) = \underbrace{\frac{(1-\alpha)z}{z-\alpha}}_{\text{low-pass}}\theta_y^a(z) + \underbrace{\frac{\alpha(1-z^{-1})}{z-\alpha}}_{\text{high-pass}}\Theta_y^g(z) \tag{6.5.22}$$

where α is the filter parameter. Substituting equation 6.5.21 into the above equation yields the following complimentary filter transfer function.

$$\Theta_y^c(z) = \frac{(1-\alpha)z}{z-\alpha}\theta_y^a(z) + \frac{\alpha T_s z}{z-\alpha}\Omega_y^g(z) \tag{6.5.23}$$

where $\Theta_y^c(z)$ is the z-transform of the fused (complimentary filtered) angular displacement signal based upon both accelerometer and rate gyro signals. Due to the complimentary filter, the transfer function associated with $\Omega_y^g(z)$ does not contain integration. Therefore, the bias in the sensor signal will not be integrated, leading to no drift in the estimated signal. Parameter α is related to the cutoff frequency of both low- and high-pass filters. Example 6.10 illustrates how to calculate the continuous-time equivalent

angular frequency based upon α.

Example 6.10

Recall Table 4.3 in Section 4.3. For the first-order low- and high-pass filters, $\alpha = e^{-aT_s}$, where T_s is the sample period and parameter a is the angular corner frequency of the equivalent continuous-time filter. Let $\alpha = 0.98$ and $T_s = 0.01$ second. Find the angular frequency of the equivalent continuous time low-pass filter. Because $\alpha = e^{-aT_s}$, the equivalent angular frequency can be calculated below.

$$a = -\frac{\ln \alpha}{T_s} = -\frac{\ln 0.98}{0.01} = 2.02 \ rad/s \tag{6.5.24}$$

Therefore, the cutoff frequency of the equivalent continuous time low-pass filter is 2.02 rad/s. Note that the continuous-time equivalent low-pass filter frequency is also a function of sample period.

Next, we will conduct a simulation study to show the effectiveness of the complementary filter for sensor fusion.

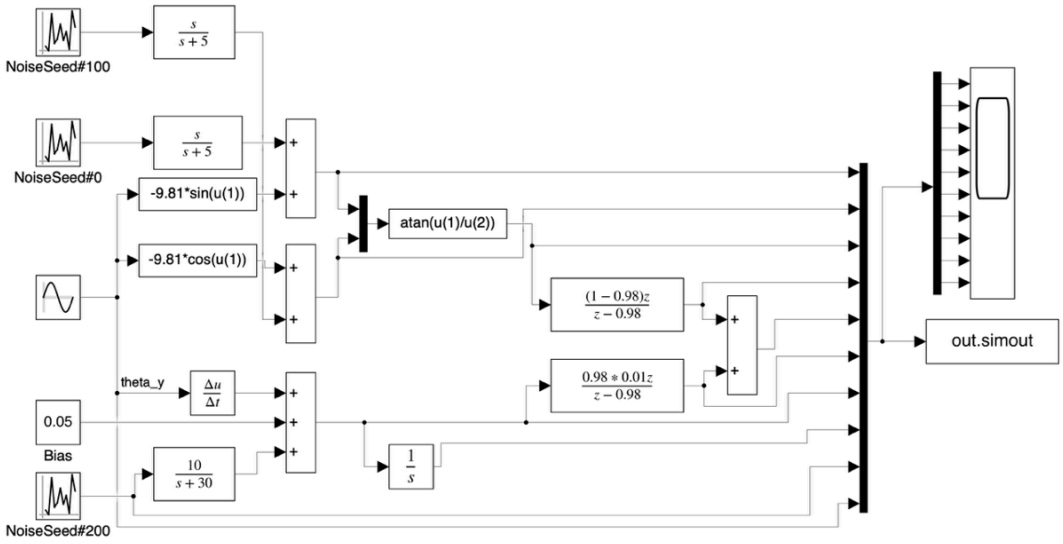


Figure 6.27: A Simulink model for studying sensor fusion.

Figure 6.27 shows a Simulink model used to study the effectiveness of the sensor fusion, where there are three uniform random number signal generators with a range between -1 and 1. Different random number seeds are used to distinguish the generated random signals. Here, we use a sinusoidal signal with magnitude of one and period of π as the reference angle, as well as a bias signal of 0.05 (5% of the sinusoidal signal). In order to simulate the actual noise characteristics, the noise signals for the accelerometer in x and z directions are filtered by high-pass filters with the transfer function defined below.

$$G_{hpf}(s) = \frac{s}{s+5} \tag{6.5.25}$$

The filter corner frequency is 5 rad/s. We apply a wide-bandwidth low-pass filter to the rate gyro sensor with the following transfer function.

$$G_{lpf}(s) = \frac{10}{s+30} \tag{6.5.26}$$

where the cutoff frequency of the low-pass filter is 30 rad/s.

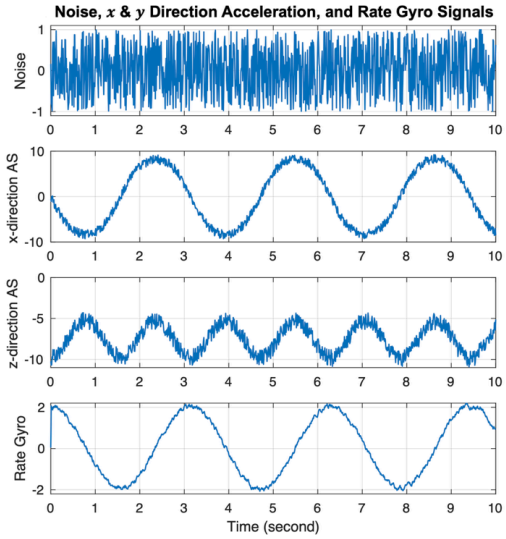


Figure 6.28: Simulink generated signals: noise, x and z direction accelerations, and angular rate.

Note that the sensor fusion coefficient is $\alpha = 0.98$. With sample period of $T_s = 0.01$ second, the associated cutoff frequency is about 2 rad/s; see equation 6.5.24. Figure 6.28 shows these signals generated by the Simulink model. The top plot shows the uniform random number generated by the model. The second plot from the top is the x-direction acceleration calculated from the reference angular position with gravity $\tilde{a} = 9.81\ m/s^2$ (see Figure 6.26 for the definition). In turn, the third plot from the top is the z-direction acceleration calculated from the reference angular position with gravity $\tilde{a} = 9.81\ m/s^2$. Finally, the bottom plot shows the output of the rate gyro signal obtained by taking derivative of the position signal.

With the required signals generated for fusion simulation (see Figure 6.27), the formula in equation 6.5.23 helps to calculate the fused angular position and an integrator helps for calculating the angular position directly from the rate gyro signal.

The top plot of Figure 6.29 shows the angular displacement calculated directly from the x_s and z_s direction acceleration signals using equation 6.5.20 (see Figure 6.26 for definition). The middle plot shows the angular displacement calculated from the angular velocity signal $\dot{\theta}_y$ (the rate gyro signal) by integrating $\dot{\theta}_y$. We can see that the accelerometer-based angular displacement signal is pretty noisy (see the top plot of Figure 6.29). The position calculated from the rate gyro signal is very smooth (almost noise free due to the integral action), but the calculated position drifts upward due to the 5% bias in the signal. As a result, the angular displacement calculated from accelerometer signals or rate gyro signal cannot be used directly.

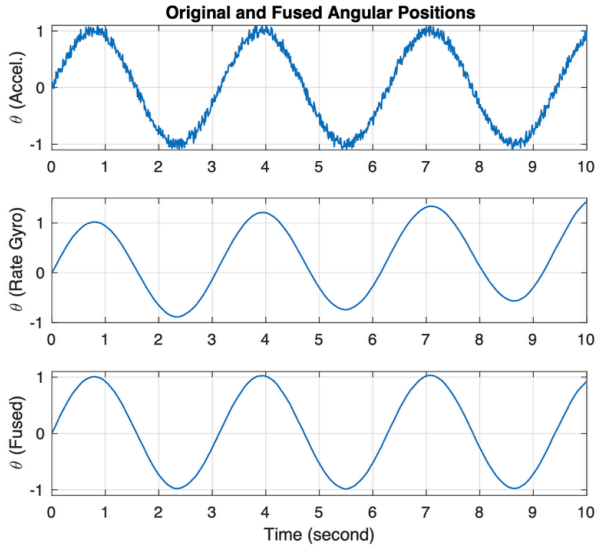


Figure 6.29: Calculated angular position by sensor fusion of accelerometer and rate gyro signals.

The bottom plot of Figure 6.29 shows the fused signal of the accelerometer and rate gyro signals based on equation 6.5.23. The fused signal recovers the original unit sinusoidal signal with a period of π without visible signal noise, compared with the integrated rate gyro single. The advantage of this fused signal is that the drift in the integrated rate gyro signal is eliminated by the sensor fusion. This simulation study shows the power of the sensor fusion.

Section Summary

In this section, the concept of the sensor fusion [7] combines multiple-sensor data or data derived from disparate sources to generate estimates that are more reliable than using these sources (sensors) individually. Two examples of fusing two sensor signals with different measurement noise levels are presented. They indicate that sensor fusion in this case is capable of significantly reducing the measurement noise. In addition, the section shows that the Kalman filter can also be deployed for sensor fusion. We provide an example of using both accelerometer and rate gyro signals to estimate the rotational angle based on complimentary filter technique. The example demonstrates how to use sensor fusion to improve estimation results, compared with using the integrated rate gyro signal (rotational speed signal) directly.

Section Keywords

Practice Problems

1. Let z_1 and z_2 be two measured signals from two independent sensors with noise variances $\sigma_1^2 = 2$ and $\sigma_2^2 = 2$, respectively. Calculate the variance of fused signal z_3 based on equation 6.5.1.
2. Consider system 6.5.3 with $A = 0.2$, $D = 1$, and $C^T = [1,\ 1]$. The input system covariance $W = 0$ and the measurement noise covariance matrix $V = diag\,[\sigma_1^2, \sigma_2^2]$. Let $\sigma_1^2 = 1$ and $\sigma_2^2 = 1.5$, respectively. Calculate the variance of the fused signal by the Kalman filter defined in equations 6.5.5, 6.5.6, and 6.5.10.
3. Let $\alpha = 0.96$ and $T_s = 0.01$ seconds. Find the angular frequency of the equivalent continuous-time low-pass filter.
4. Use the Simulink simulation model presented in Figure 6.27, and let $\alpha = 0.5$. Conduct a simulation study and compare the result with those in Figure 6.29.

This page titled 6.5: Sensor Fusion is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing

Learning Objectives

After completing this section, you will be able to:

1. Explain the working principle of a micro-electromechanical system (MEMS) device.
2. Introduce the MEMS sensor architecture and its communication capability.
3. Describe the three-axis MEMS accelerometer and rate gyro sensor (MPU 6050).

Sections 6.3 and 6.4 reviewed the mechanical architectures of single-axis accelerometer and rate gyro sensors. Now this section will describe the general architecture of a MEMS (micro-electromechanical system) sensor in detail, where the MPU 6050 is used as an example.

Micro-electromechanical systems are microscopic devices incorporating both electronic and mechanical moving parts in a single unit. MEMS are made up of components between 1 and 100 micrometers in size (i.e., 0.001 to 0.1 mm). MEMS devices generally range in size from 20 micrometers to a millimeter (i.e., 0.02 to 1.0 mm). However, MEMS components can be arranged in arrays with increased dimension. A MEMS sensor usually consists of a central processing unit, such as a microprocessor or microcontroller with several micromechanical sensing components detecting the changing environment. Figure 6.30 shows a general architecture of a MEMS sensor. It consists of a micromechanical sensing system, a microprocessor or microcontroller for signal processing, and output electronics to communicate with the main microcontroller in the information domain of a mechatronic system.

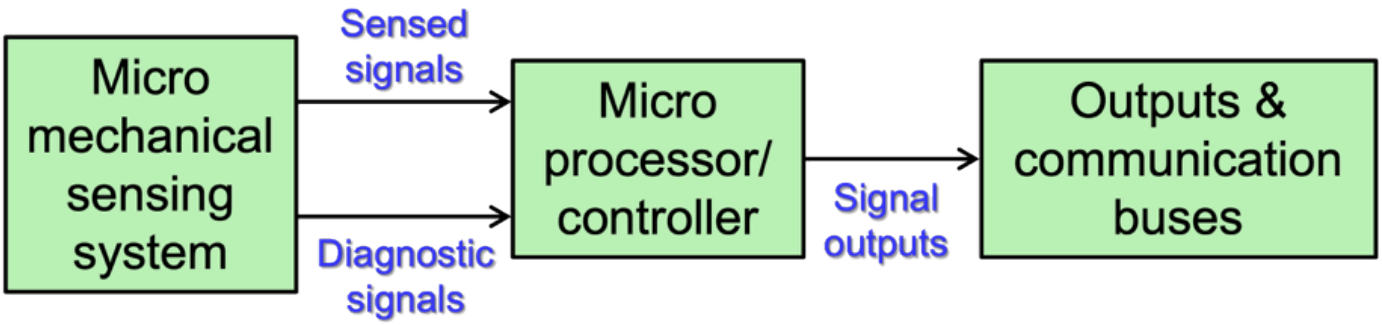


Figure 6.30: MEMS sensor architecture.

For the micromechanical sensing system, we have already discussed sensing elements such as accelerometers (Figure 6.15) and rate gyros (Figure 6.22) in the MEMS. The microprocessor or microcontroller processes the signals provided by the micromechanical sensing system, along with the health diagnostic results of the sensing system, and compensation for nonlinearity, temperature, and other factors. In order to ensure fast communication between the MEMS sensor and the microcontroller in the information domain of the mechatronic system, fast communication buses are often used.

Before getting to the details of MPU 6050, let's look further at MEMS-based sensors. MEMS became practical once they could be fabricated using modified semiconductor manufacturing technologies [36]. These technologies include molding and plating, wet and dry etching, electrical discharge machining (EDM), and other technologies capable of manufacturing small devices. There are two basic types of MEMS sensing technologies: capacitive and ohmic. A capacitive MEMS switch is developed using a moving plate or sensing element, which changes the capacitance [37] (see Figure 6.15 and Figure 6.22). Ohmic switches are controlled by electrostatically controlled cantilevers [38]. Note that ohmic MEMS can fail due to metal fatigue of the MEMS actuator and contact wear since cantilevers can deform over time.

The fabrication of MEMS evolved from the technology of semiconductor fabrication to the basic techniques of depositing material layers, patterning, and etching to produce the required shapes [39]. In this section, we will concentrate on silicon-based MEMS.

Silicon is the material used to create most integrated circuits widely used in consumer electronics due to readily available inexpensive high-quality materials and the ability to incorporate electronic functionality. Using silicon for MEMS is also due to its advantages of material properties. For example, single crystal silicon is an almost perfect Hookean material [40]. That is, when the silicon is flexed or bended, there is virtually no hysteresis. This leads to almost no energy dissipation or loss, which is perfect for the vibration-based rate gyros discussed in Section 6.4 due to its highly repeatable oscillations. Nowadays, silicon nanostructures [41] are gaining in importance in the field of microelectronics and MEMS technologies.

There are a few manufacturing approaches for MEMS, including bulk micromachining, surface micromachining, and wafer bonding. The bulk micromachining is the oldest paradigm of silicon-based MEMS, where the entire thickness of a silicon wafer is used for building the micromechanical structures [42] and the silicon is machined using various etching process. Surface micromachining uses layers deposited on the surface of a substrate as the structural materials, rather than using the substrate itself [43]. Surface micromachining was created in the late 1980s. It renders micromachining of silicon more compatible with planar integrated circuit technology to combine MEMS with the integrated circuit (e.g., microprocessor or microcontroller) on the same silicon wafer. Wafer bonding involves joining two or more substrates (usually having the same diameter) to one another to form a composite structure.

Both bulk and surface silicon micromachining are used in the industrial production of sensors, ink-jet nozzles, and other devices. Recently, these two approaches merged due to a new etching technology, called deep reactive ion etching [44].

For the rest of this section, we will concentrate on a widely used MEMS sensor, MPU 6050 [8] that is a three-axis accelerometer and a three-axis rate gyro sensor. This sensor will also be used in both Chapter 8's discussion of the mini-Segway and Chapter 9's discussion of the three-wheel cart. The MPU 6050 is the center chip on the MPU 6050 board for Arduino applications in the left plot of Figure 6.31. It is the world's first integrated six-axis motion tracking device that combines a three-axis gyroscope, a three-axis accelerometer, and a digital motion processor (DMP), all in a small $4 \times 4 \times 0.9$ mm package. This MEMS sensor is installed on the inversed pendulum of the mini-Segway (see Chapter 8). It mainly measures the x and z direction accelerations and y direction angular speed (see Figure 6.25), where the measured accelerations and angular speed are fused to obtain the angular position in y direction (Section 6.5). The circuit connected to the MPU 6050 is shown on the left of Figure 6.31.

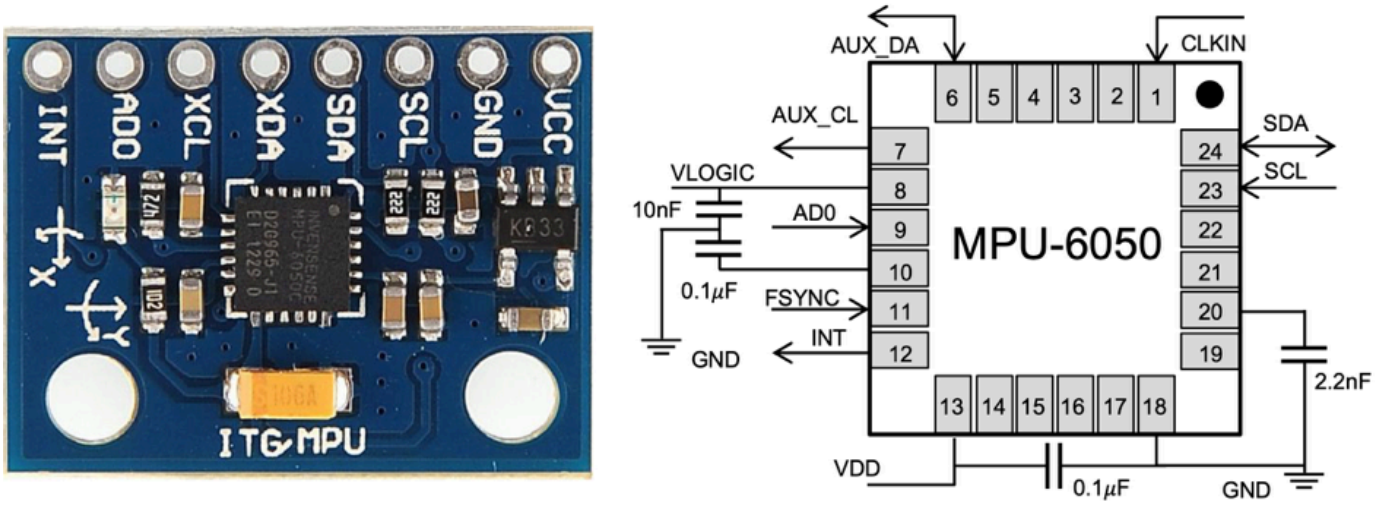


Figure 6.31: MPU 6050 board for Arduino applications.

Figure 6.32 shows a block diagram of the MPU 6050. The red-dashed box indicates the communication block from Figure 6.30, the blue-dashed box represents the micromechanical sensing elements, and the rest of the components are part of microcontroller. A detailed description can be found in the data sheet of MPU 60X0 [8].

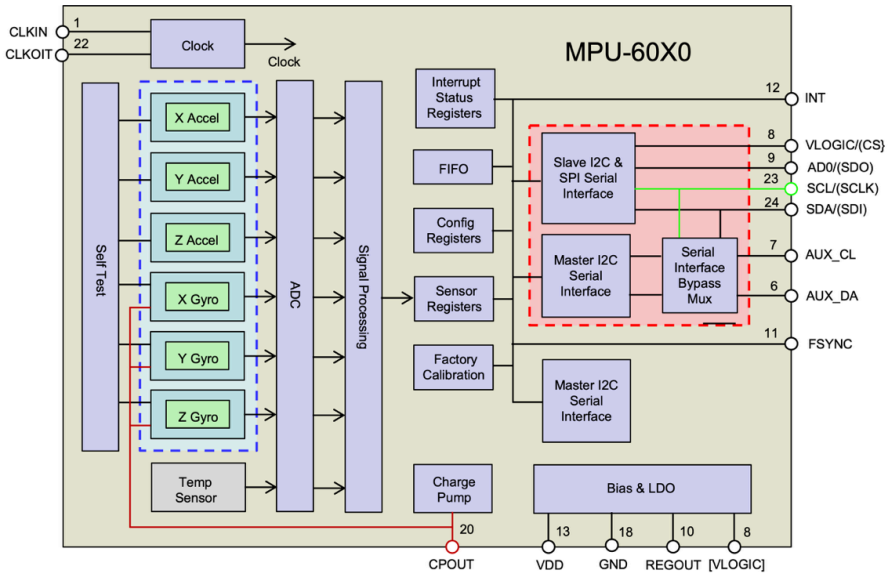


Figure 6.32: MPU 6050 block diagram.

For the micromechanical part (inside the blue-dashed line box), we can find the x, y, and z direction accelerometers (three) and rate gyros (three). These accelerometers and rate gyros are in the form presented in Figure 6.15 for the accelerometers and Figure 6.22 for the rate gyros.

For the microcontroller part, MPU-6050 features three 16-bit analog-to-digital converters (A/Ds) for digitizing the gyroscope outputs and three 16-bit A/Ds for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, this part features a user-programmable gyroscope full-scale range ($\pm$25, $\pm$500, $\pm$1000, and $\pm$2000 degree/sec) and a user-programmable accelerometer full-scale range ($\pm$2, $\pm$4, $\pm$8, and $\pm$16 g). Additional features include an embedded temperature sensor for temperature compensation and an on-chip oscillator with $\pm 1\%$ variation over the operating temperature range.

For the communication part, all device registers are performed using either I2C bus at 400kHz or SPI bus for both the MPU 6000 and the MPU 6050 at 1MHz (MPU 6000 only). For applications requiring faster communications, the sensor and interrupt registers can be read using SPI at 20MHz (MPU-6000 only).

Section Summary

This section addresses the general concept of the MEMS (micro-electromechanical system) regarding its dimension, system architecture, and how it is made. We introduce in detail a three-axis accelerometer and gyroscope integrated sensor system MPU 6050 [8], utilizing MEMS technology. The MPU 6050 is the world's first integrated six-axis motion tracking device that combines a three-axis gyroscope, three-axis accelerometer, and a digital motion processor (DMP) all in a small $4 \times 4 \times 0.9$ mm package. The mechanical acceleration and gyroscope sensors are integrated with the digital motion processor (DMP) capable of processing complex motion fusion algorithms and run-time calibration firmware to eliminate the costly and complex calibration process.

Section Keywords

Practice Problems

1. Select one correct statement from the following list.
   a. MEMS are made up of components between 1 and 100 micrometers in size (i.e., 0.001 to 0.1 mm).
   b. MEMS are made up of components between 20 micrometers and a millimeter (i.e., 0.02 to 1.0 mm).
2. List a few advantages of using silicon as the material for MEMS.

This page titled 6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing

Learning Objectives

After completing this section, you will be able to:

1. Describe the working principle of temperature sensors such as RTD (resistance temperature device), thermal couple, and air flow sensors.
2. Grasp the working principle of pressure sensors such as piezo-pressure sensors.
3. Explain electric current and voltage sensing principles.

In earlier sections, we reviewed a few mechatronic sensors including rotational hall-effect speed sensors, accelerometers, and rate gyros. In this section we will briefly review sensing principles of temperature, pressure, air flow, current, and voltage sensors. Our intension is to provide an overview and how to use these sensors in general.

First let's look at temperature sensing technology. Generally, there are four main kinds of electrical temperature sensors. The first kind is the thermistor [45], a resistor made of semiconductor materials; its resistance is strongly dependent on temperature. Note that the word *thermistor* combines *thermal* and *resistor*. Negative temperature coefficient (NTC) thermistors exhibit a decrease in electrical resistance as their body temperature increases. By contrast, positive temperature coefficient (PTC) thermistors have an increase in electrical resistance as body temperature goes up.

The second kind is a thermocouple [46], also known as a thermoelectrical thermometer. It is an electrical junction consisting of two dissimilar electrical conductors. A thermocouple produces a temperature-dependent voltage as a result of the Seebeck effect, that is, the electromotive force developed across two points of an electrically conducting material when there is a temperature difference between them. Note that the generated voltage can be interpreted to measure temperature.

The third kind is a resistance thermometer [9], also called a resistance temperature detector (RTD). It is used to measure temperature. Most RTDs consist of a length of fine wire wrapped around a heat-resistant ceramic or glass core. The RTD wire is a pure material, typically platinum, nickel, or copper. These materials have an accurate resistance and temperature relationship that is used to provide an indication of temperature. Because RTD elements are fragile, they are often housed in protective probes. Due to RTDs' high accuracy and repeatability, they are slowly replacing thermocouples in industrial applications below $600\ ^{\circ}\mathrm{C}$.

The last kind is a silicon bandgap temperature sensor in the form of thermometer. It is widely used in electronic equipment because it can be part of a silicon integrated circuit at very low cost. The principle of the sensor is that the forward voltage of a silicon diode (see Chapter 2), such as the base-emitter junction of a bipolar junction transistor, is temperature-dependent. It can be used to detect junction temperature [47].

We will concentrate on resistance temperature detector (RTD) for the rest of the temperature sensing discussion. Note that the resistance temperature detector is described by its temperature and resistance relationship. For most RTDs, their resistance increases as the temperature goes up within the measurement range. This is due to the fact that the atoms in the metal wire of the sensor vibrate more at higher temperatures; this causes more collisions with flowing electrons, thus increasing the resistance to electrical current flow.

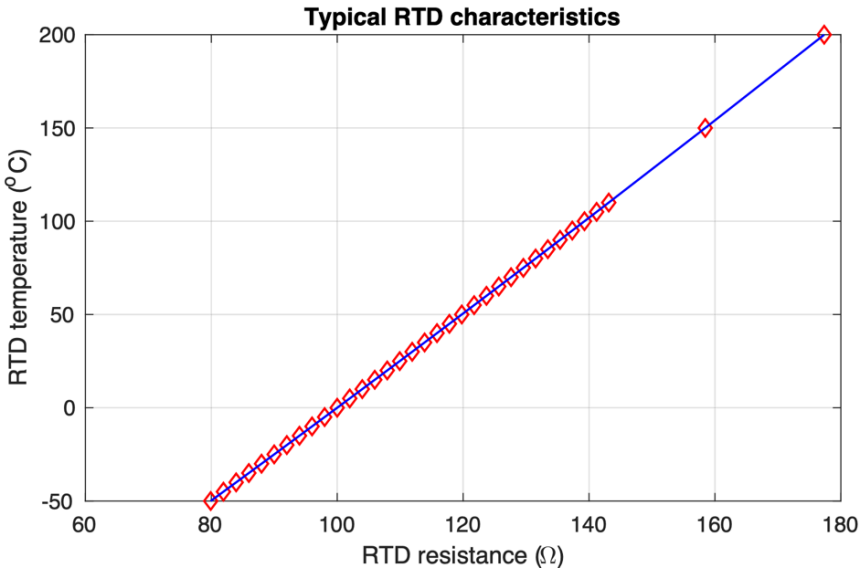


Figure 6.33: Typical RTD characteristics.

Figure 6.33 shows a typical RTD sensor characteristic chart representing the relationship between RTD temperature and associated resistance value. One can see that RTD resistance increases as temperature rises. This phenomenon is known as a positive temperature coefficient, where the resistance of the material directly increases with temperature. To use the RTD to measure temperature, the RTD resistance value needs to be converted into a voltage signal that can be sampled into the information domain. One approach is to connect the RTD (R_{RTD}) to a reference voltage through a reference resistor R_{ref}, as shown in Figure 6.34, where an operational amplifier is used to isolate the RTD circuit from the connected circuit.

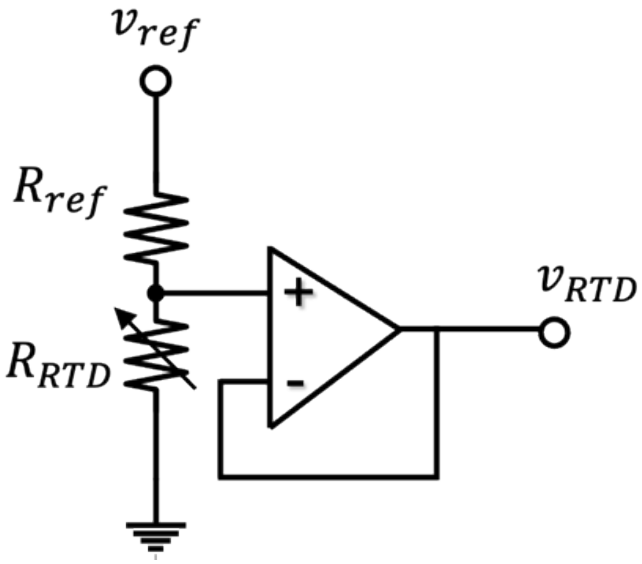


Figure 6.34: Serial-connected RTD sensor.

The simple derivation, utilizing the techniques discussed in Chapter 2, leads to the following relationship between the RTD and v_{RTD}.

$$v_{RTD} = \frac{R_{RTD}}{R_{RTD} + R_{ref}} v_{ref} \tag{6.7.1}$$

Another method is to use an H-bridge circuit, shown in Figure 6.35, with a differential operational amplifier circuit because the RTD voltage signal is referenced as being half of v_{ref}. The advantage of this circuit will be addressed later in this section.

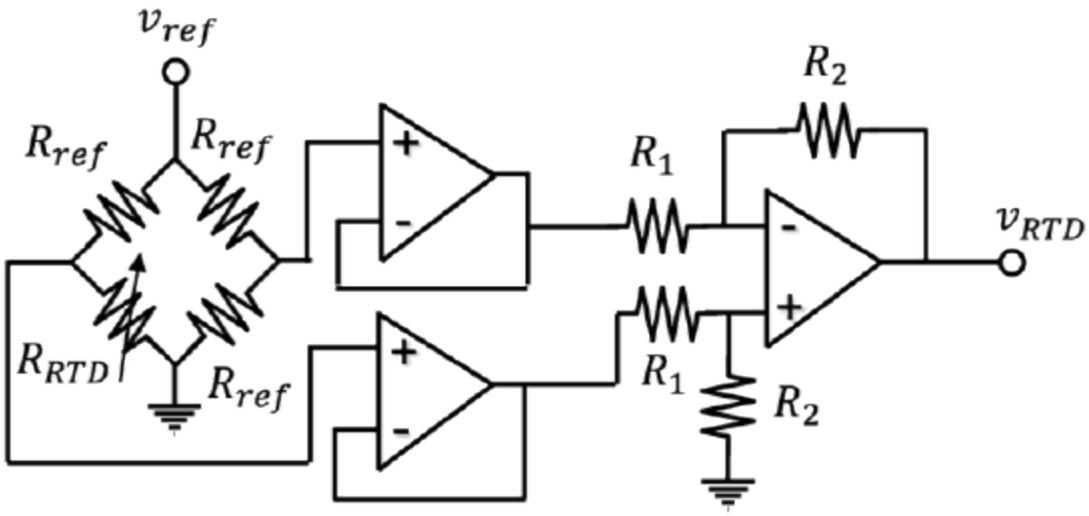


Figure 6.35: H-bridge RTD circuit.

Utilizing the techniques discussed in Chapter 2, the following relationship between RTD resistance R_{RTD} and circuit output voltage v_{RTD} can be derived.

$$v_{RTD} = \frac{R_2}{2R_1}\frac{(R_{RTD} - R_{ref})}{(R_{RTD} + R_{ref})}v_{ref} \tag{6.7.2}$$

Comparing equations 6.7.1 and 6.7.2, we find that using the H-bridge circuit leads to more freedom for tuning the output voltage v_{ref} in terms of the output voltage range and output voltage for a given temperature by adjusting R_1, R_2, and R_{ref}. We will use Example 6.11 and Problem 6.24 to illustrate.

Example 6.11

Let $R_1 = 5\ k\Omega$, $R_2 = 10\ k\Omega$, and $R_{ref} = 75\ \Omega$. Find the relationship from R_{RTD} to v_{RTD} and using the RTD relationship in Figure 6.33 to plot the relationship from measured voltage to RTD temperature, given $v_{ref} = 5\ V$. Substituting given parameters into equation 6.7.2 yields the following.

$$v_{RTD} = \frac{R_2}{2R_1}\frac{(R_{RTD} - R_{ref})}{(R_{RTD} + R_{ref})}v_{ref} = 5\frac{R_{RTD} - 75}{R_{RTD} + 75} \tag{6.7.3}$$

Using the formulae above, we can calculate the RTD resistance R_{RTD} for a given output voltage v_{RTD}. Then use the relationship shown in Figure 6.33 to find the associated RTD temperature. The relationship from the output voltage v_{RTD} to RTD temperature is shown in Figure 6.36. Note that the relationship is no longer linear.

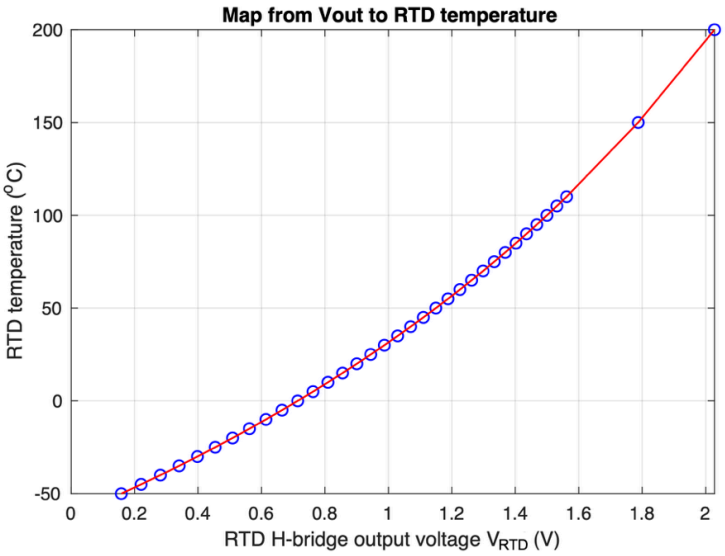


Figure 6.36: H-bridge RTD relation from v_{RTD} to RTD temperature.

Note that the map shown in Figure 6.36 can be used to find the RTD temperature based upon the H-bridge circuit output voltage v_{RTD} directly.

Now we will discuss the working principles and different types of pressure transducers and pressure sensors. Pressure is defined as the force applied by a liquid or gas on a surface. It is usually measured in units of force per unit of surface area, commonly in Pascal (Pa), Bar (bar), N/mm^2, or PSI (pounds per square inch). A pressure sensor is a device that measures the physical quantity (pressure) and translates it to an electrical or an optical signal. A pressure sensor consists of a pressure sensitive element to determine the actual pressure applied to the sensor and some components to convert this information into an output signal. As a result, pressure sensors are also known as pressure transmitters.

A variety of properties can be used to classify pressure sensors, including absolute, gauge, and differential pressure sensors. Absolute pressure sensors measure the pressure relative to a reference chamber (nearly vacuum). Gauge pressure sensors (also called relative pressure sensors) determine the pressure relative to the present atmospheric pressure. Differential pressure sensors find the difference between two pressure chambers for measuring pressure drops, fluid levels, and flow rates.

Pressure sensors can also be categorized by their sensing element types mainly as strain gauge [48], piezoelectric [49], and capacitive pressure [50] sensors. A strain gauge pressure sensor measures change in the ohmic resistance of a strain gauge with its deflection magnitude caused by applied pressure proportional to the change in resistance. A piezoelectric pressure sensor determines the electric charge due to applied stress resulted from pressure, bending, twisting, or vibrations. The electric charge is used to determine the amount of pressure. Capacitive pressure sensors use a diaphragm and pressure cavity to create a variable capacitor as a function of pressure applied to the diaphragm.

We will focus on the working principle of the strain gauge, piezoelectric, and capacitive pressure sensors. First, for the strain gauge pressure sensor, the resistance of the strain resistance absorbed on the base material changes with the mechanical deformation. Figure 6.37 illustrates this effect, which is commonly known as the resistance strain effect.

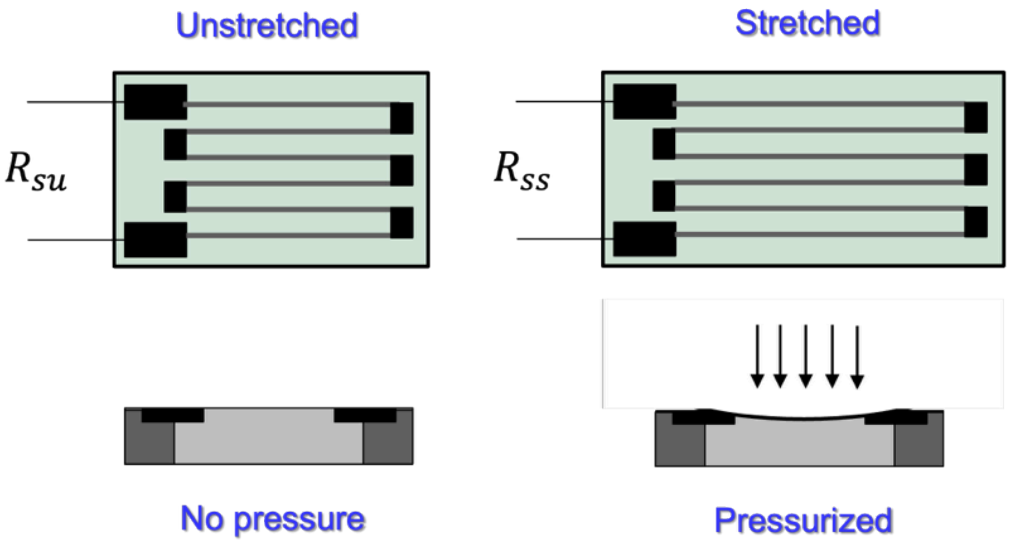


Figure 6.37: Unstretched and stretched strain gauges.

The top plots in Figure 6.37 show an unstretched strain gauge strip with resistance of R_{su} (top left) and a stretched strain gauge strip with resistance of R_{ss}, where $R_{ss} > R_{su}$ (top right). The bottom plots in Figure 6.37 show unstretched and stretched strain gauge strips installed on a pressure sensor diaphragm. When the pressure is applied, the sensor diaphragm is deformed downward (bottom right). The strain gauge is stretched so that its resistance is increased. Note that the resistance of the strain gauge also changes under different environments (e.g., temperature and electromagnetic field). If the resistance is too high, resulting in high impedance, the ability to resist external electromagnetic interference is poor. For most of the strain gauges, the resistance is around tens of ohms to tens of thousands of ohms.

Figure 6.38 shows a strain gauge pressure sensor with four strain gauges evenly distributed on the diaphragm. The associated resistance values are denoted by R_{vh}, R_{vl}, R_{hl}, and R_{hr}, marked on the left plot of Figure 6.38. We can define the following resistance values marked on the circuit part of Figure 6.38.

$$R_v = R_{vh} + R_{vl}, \quad R_h = R_{hl} + R_{hr} \tag{6.7.4}$$

Using the circuit on the right of Figure 6.38, we can find the change of resistance values due to applied pressure. Note that the formula for calculating the output voltage as a function of these resistance values is found in Problem P6.25 at the end of the section.

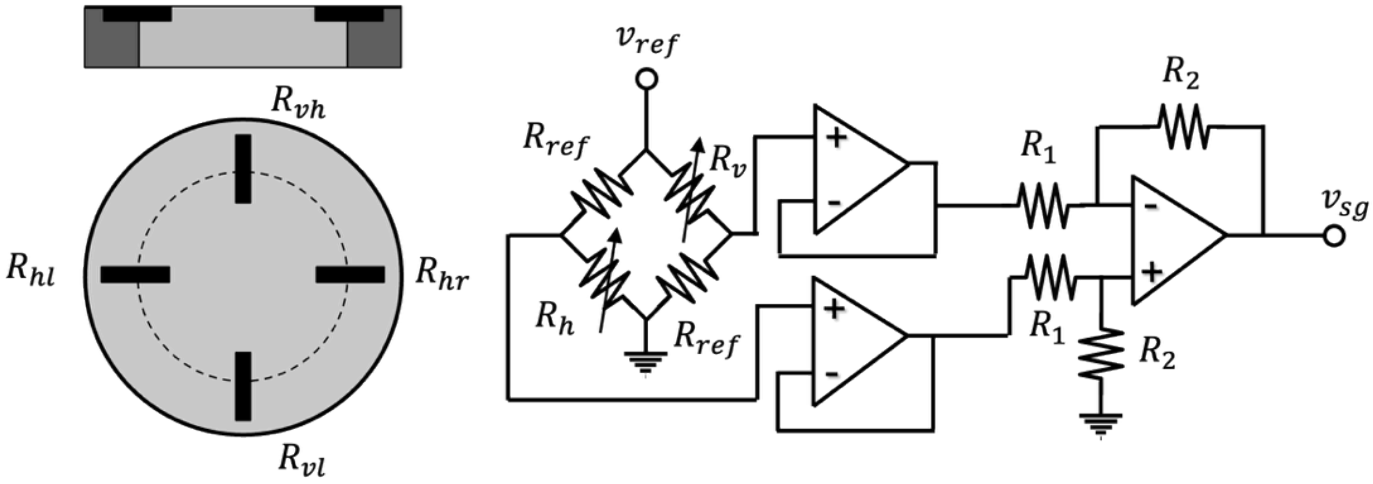


Figure 6.38: Strain gauge pressure sensor and detection circuit.

Next, we will discuss the working principle of a piezoelectric pressure sensor. The piezoelectric materials used in piezoelectric pressure sensors include quartz, potassium sodium tartrate, and dihydrogen phosphate. Among them, quartz is a natural crystal.

The piezoelectric effect is found in this crystal. Within a certain temperature range, the piezoelectric property always exists. After the temperature exceeds this range, the piezoelectric property completely disappears. When certain dielectrics inside the piezomaterial are deformed by external forces in a specific direction, polarization will occur inside them. Positive and negative charges appear on their two opposing surfaces, generating a positive voltage v_{ps} (see the right plot of Figure 6.39 for the opposition surfaces). When the external force is removed, it will return to the uncharged state with $v_{ps} = 0$. This phenomenon is called a positive piezoelectric effect.

When the direction of the applied force changes, the polarity of the charge changes accordingly. Conversely, when an electric field is applied in the polarization direction of the dielectric, these dielectrics will also deform. After the electric field is removed, the deformation of the dielectric will disappear. This phenomenon is called the inverse piezoelectric effect. The type of sensor developed based on the dielectric piezoelectric effect is called a piezoelectric sensor. The right plot of Figure 6.39 illustrates the sensor structure. The piezoelectric material is sandwiched between the fixed insulation (below the piezoelectric material) and the diaphragm on top of the piezoelectric material. As a result, when the diaphragm is deformed downward due to pressure, voltage v_{ps} (sensor signal) is produced. This signal is very weak. A charge amplifier is often used to obtain a signal with certain magnitude by isolating the piezoelectric material output signal and sensor output.

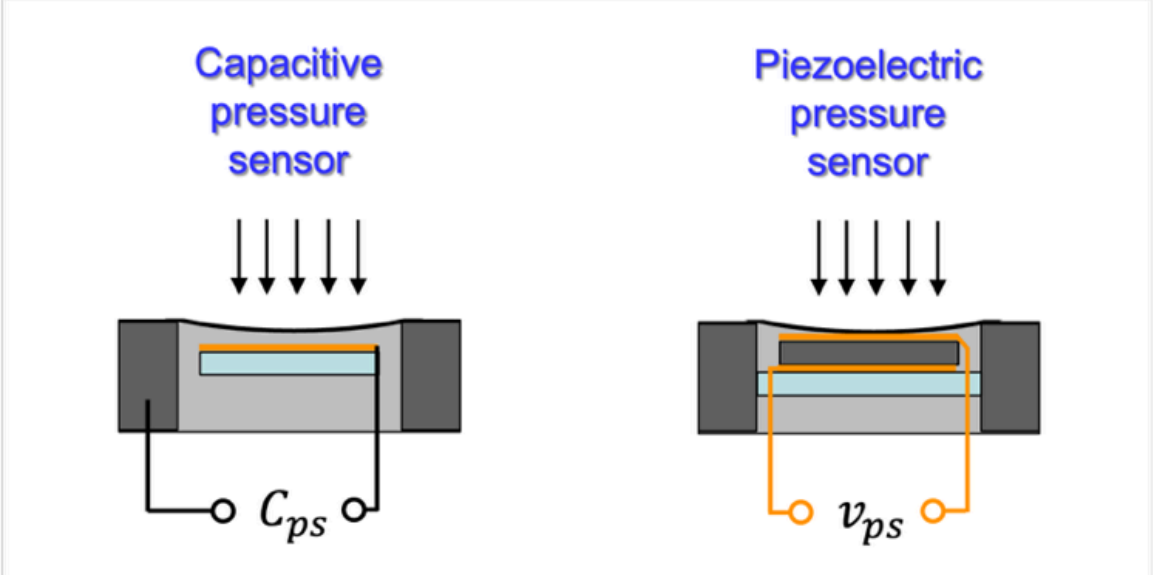


Figure 6.39: Piezoelectric and capacitive pressure sensors.

The next group are capacitive pressure sensors. They use a diaphragm and pressure cavity to create a variable capacitor to detect strain due to applied pressure (see the left plot of Figure 6.39), where capacitance increases as pressure deforms the diaphragm. The capacitance of the sensor is typically around 50 to 100 pF with the change being a few picofarads (pFs). The stiffness and strength of the diaphragm material can be chosen to provide a range of sensitivities and operating pressures. To get a large signal, the capacitive pressure sensor may need to be fairly large. This increases the capacitance, which can limit its operational frequency range. However, smaller diaphragms are more sensitive and have a faster response time. Common technologies use metal, ceramic, and silicon diaphragms. Capacitive pressure sensors can also be integrated into CMOS (complementary metal-oxide-semiconductor) technology [51].

Let's look now at the fluid and gas flow sensors. A flow sensor is a component that measures the flow of a fluid such as gas or liquid. Flow sensors utilize both mechanical and electrical subsystems to measure changes in the fluid's physical characteristics and calculate its flow. Gaseous and liquid fluids behave vastly different from one another. Thus, the methods we use to measure their flow must be different. We will mainly address the gear flow meter for measuring liquid flow and thermal mass flow meter for determining gases mass flow.

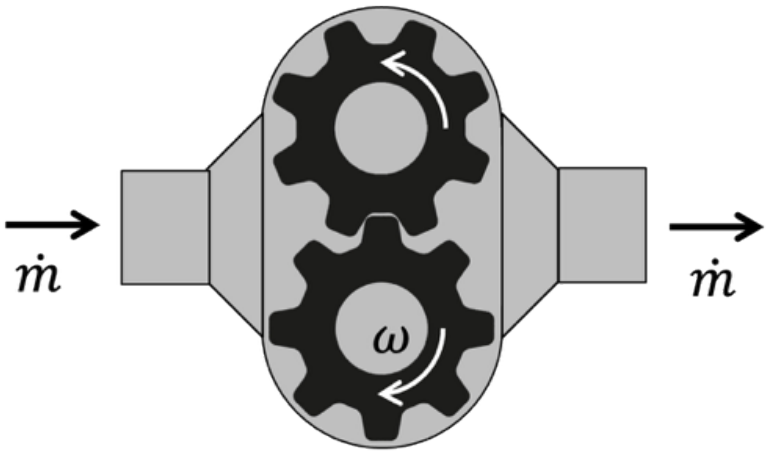


Figure 6.40: Fluid gear flow sensing illustration.

Figure 6.40 presents the principle of a fluid gear flow sensor. A pair of mechanical gears are used in the center of the plot. As the fluid mass flow into the gear flow meter, both gears rotate in an opposite direction with their speed as a function of the mass flow rate $\dot{m}$. To convert the rotational speed to the mass flow rate, a rotational speed measurement device, such as the one introduced in Section 6.2, can be used to obtain the gear rotational speed ω. A calibration lookup table can then be made to convert the measured speed ω to mass flow rate $\dot{m}$. There is another type of gear flow meter utilizing the oval-shape gears with improved performance by replacing circular gears with oval-shape gears.

For the mass air flow measurement, the hot-wire mass air flow (MAF) sensor converts the amount of mass air flow into a voltage signal v_{maf} (see Figure 6.41). The primary components of the hot-wire MAF sensor are a thermistor (represented by R_{th}), a platinum hot wire (represented by R_{hot}), and an electronic control circuit consisting of the transistor T and temperature regulator. The thermistor measures the temperature of the incoming air, and the hot wire is maintained at a constant temperature in relation to the thermistor (normally higher than the thermistor temperature) by an electronic control circuit. An increment in air flow will increase the heat loss of the hot wire. The electronic control circuitry will compensate by increasing the current flown through the wire.

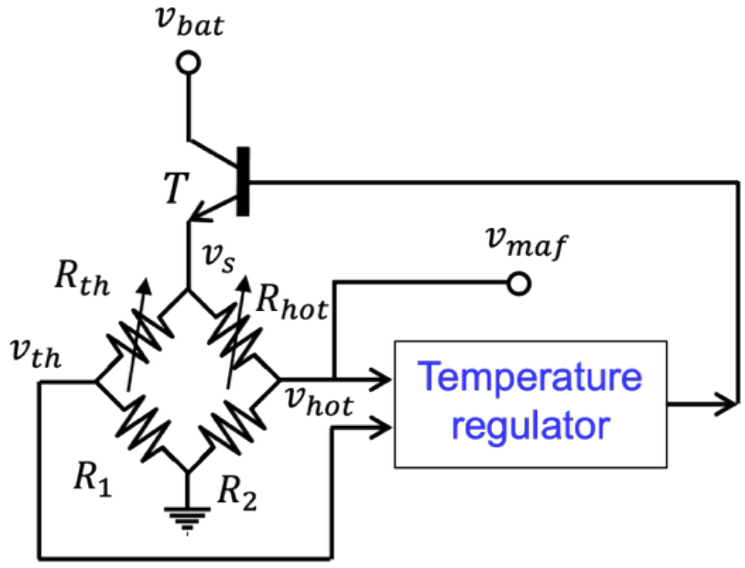


Figure 6.41: Hot-wire MAF sensor illustration.

Note from Figure 6.41 that the electronic control circuit measures voltages v_{th} and v_{hot} while the temperature regulator controls the voltage v_s to make the voltage difference $v_{hot} - v_{th}$ equal to zero (or a given difference). The goal is to maintain the

temperature of the hot-wire resistor R_{hot} greater than the thermistor one by a designed value. Note that setting $v_{hot} - v_{th} = 0$ leads to the following.

$$v_{th} = \frac{R_1}{R_{th} + R_1} = \frac{R_2}{R_{hot} + R_2} = v_{hot} \tag{6.7.5}$$

The above equation is equivalent to the following relationship.

$$\frac{R_{th}}{R_1} = \frac{R_{hot}}{R_2} \quad \rightarrow \quad R_{hot} = \frac{R_2}{R_1} R_{th} \tag{6.7.6}$$

Therefore, by selecting the proper properties of R_{th} and R_{hot} and maintaining $v_{hot} - v_{th} = 0$, the hot-wire resistor temperature can be maintained at a level above air temperature. Also, mass air flow rate can be calculated based up output $v_{maf} = v_{hot}$. Normally a calibration table is needed to convert the measured v_{maf} to the mass air flow rate. We will use Example 6.12 to show how to achieve the target hot-wire resistor temperature by maintaining $v_{hot} - v_{th} = 0$.

Example 6.12

Let the thermistor resistance and temperature have a relation of $R_{th} = 2\,(T_{th} + 50) + 1200\ \Omega$ for temperature $T \in [-50,\ 200]^\circ\mathrm{C}$, the hotwire resistor $R_{hot} = 0.2\,(T_{hot} + 40) + 120\ \Omega$ for temperature $T \in [-50,\ 200]\ ^\circ\mathrm{C}$, $R_1 = 100\ \Omega$, and $R_2 = 10\ \Omega$. Note that 10 times the difference between R_1 and R_2 is mainly due to the difference in R_{th} and R_{hot}. Assume that for the hot-wire MAP shown on Figure 6.41, $v_{hot} - v_{th}$ is maintained at zero and $T_{th} = 20℃$. Find the hot-wire resistor temperature T_{hot}.

Figure 6.42 plots the temperature and resistance relations of both the thermistor and hot-wire resistors. First, we draw a vertical line at 20℃ on the thermistor plot, map the intersection horizontally to the equivalent resistance hotwire plot, and draw the vertical line through the intersection on the right plot. We can then see that if $v_{hot} - v_{th} = 0$, the temperature of hot-wire resistor is 30℃, which is 10℃ higher than the thermistor's temperature.

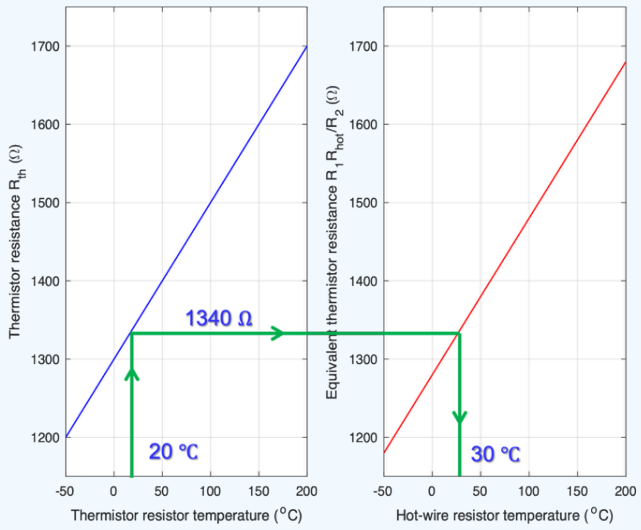


Figure 6.42: Temperature and resistance relations of a thermistor and a hot-wire resistor.

As a result, to make a hot-wire mass air flow sensor work, it is very important to properly match thermistor property with hot-wire resistor characteristics. Otherwise, the hot wire may not maintain at its temperature above the thermistor one. The hot-wire resistor temperature can also be calculated analytically. If the thermistor is held at 20℃, the corresponding resistance is $R_{th} = 2\,(25 + 50) + 1200 = 1340\ \Omega$. We can calculate the hot-wire resistance as follows.

$$R_{hot} = \frac{R_2}{R_1} R_{th} = \frac{10}{100} 1340 = 134\ \Omega \tag{6.7.7}$$

Solving the given equation $R_{hot} = 0.2\,(T_{hot} + 40) + 120\ \Omega$ leads to a hot-wire resistor temperature of 30℃.

Next up are techniques for current and voltage sensing. For a mechatronic system, the current sensor signal will eventually be converted into voltage signals. A/D converters can then sample them into the information domain. Let's first address how to detect a voltage signal that is not referenced to the ground (also called floating voltage signal). One example is the H-bridge RTD circuit shown in Figure 6.35. The voltage difference over two sample points is the measuring voltage. An operational amplifier circuit is used to convert the voltage signal referenced to the ground. The A/D converter in the information domain is usually referenced to ground. As a result, the operational amplifier circuit in Figure 6.35 is widely used.

In certain applications, the distance from the sensor (or converted voltage) to the A/D converter in the information domain is fairly long. In these cases, the sensor ground and A/D converter ground may be slightly different due to ground noise or bias, especially for these high-power (current) applications. We recommended converting the voltage signal into a current source signal. Then use a burn resistor to convert the current signal back to voltage for the A/D converter in the information domain; see Figure 6.43.

For Figure 6.43, v_s is the sensor output voltage. An operational amplifier is used to isolate the input from the rest of the circuit and convert the voltage to current with $i_s = v_s/R_2$ with the help of transistor T_3. This is because the voltage drop between positive and negative inputs of the operational amplifier is zero. Inside the red-dashed box is a current mirror circuit, making the current flow through collectors of transistors T_1 and T_2 the same. In order to have an accurate current mirror, transistors T_1 and T_2 needs to be manufactured on the same silicon so that their characteristics are the same.

Finally, the sensor output voltage v_s is recovered at the microcontroller side with $v_{s-A/D} = R_2 i_s$. As indicated in Figure 6.43, the sensor and microcontroller ground are different. If v_s is directly connected to the A/D converter, there is significant ground noise, leading to noisy sensor signal sampling by the A/D converter. This is the main reason to use the current source signal to connect the sensor signal to the A/D converter of the microcontroller in the information domain.

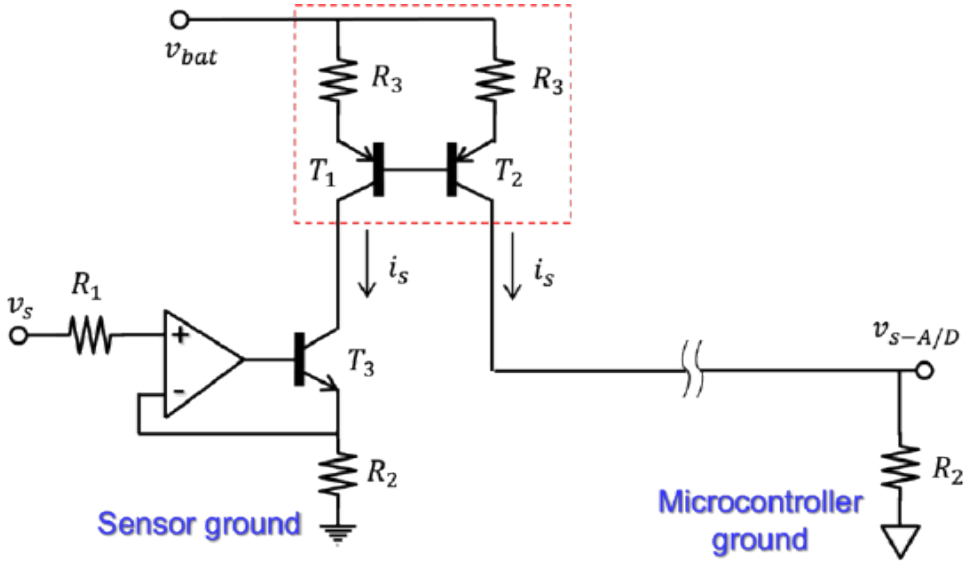


Figure 6.43: Current source voltage conversion circuit.

Now we'll address current measurement. Information regarding current flow is required in a wide range of mechatronic systems. Often, current information needs to be available in the digital form in the information domain for feedback control. This requires the current signal to be converted into a digital signal by an analog-to-digital (A/D) converter. For example, to measure current, we can use a shunt resistor (e.g., resistor R_1 of the circuit in P6.27) connected to the load source. Shunt resistors are widely used in power electronics due to their low cost, small size, and simplicity with reasonable accuracy. This method converts the current measurement problem into a voltage measurement problem. A disadvantage of using the shunt resistor for current sensing is the high energy consumption for applications with a very large current. For example, if the current flow through the shunt resistor is $i_{sh} = 10\ A$ and the resistor is $R_{sh} = 0.05\ \Omega$, the power consumed by the shunt resistor is $P = i_{sh}^2 R_{sh} = 5\ W$.

Let's discuss a current measurement approach based on the Rogowski coil principle [52], [53]. A Rogowski coil is a toroid of wire used to measure an alternating current $i(t)$ through a cable encircled by the toroid. Figure 6.44 shows a Rogowski coil encircling a current-carrying cable. The output of the coil, $v(t)$, is connected to a lossy integrator circuit to obtain a voltage $v_{out}(t)$ that is proportional to $i(t)$. The Rogowski coil displayed in Figure 6.44 is a typical application example using Faraday's induction law.

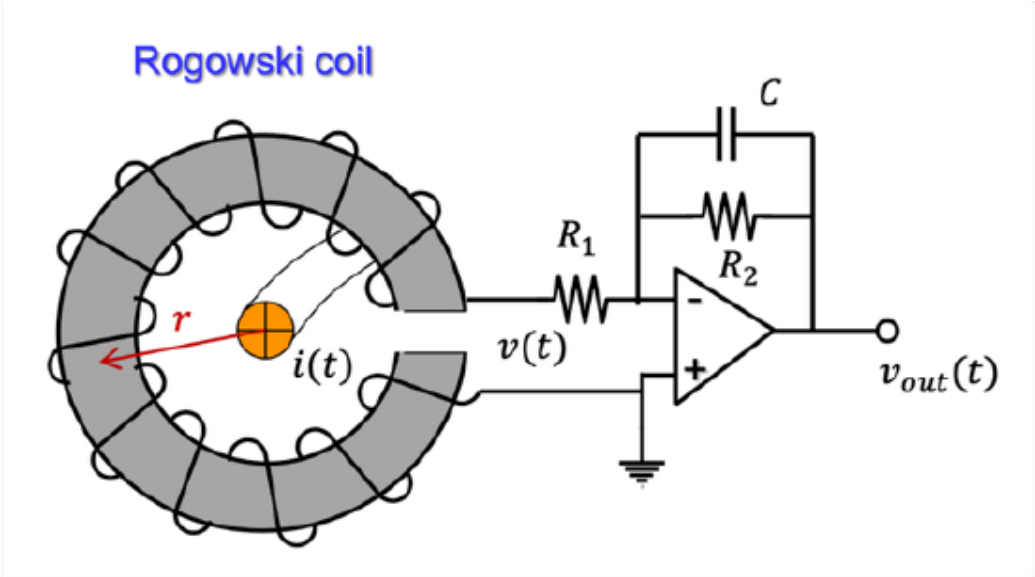


Figure 6.44: Rogowski coil and its amplifier circuit.

By applying Faraday's law, we can have the relationship shown in the following equation (see [53] for detailed derivation).

$$v\left(t\right)=-N\frac{A\mu_0}{2\pi r}\frac{di(t)}{dt} \tag{6.7.8}$$

where N is the number of turns; A is the cross-sectional area of the coil body formed by the windings; r is the radius of the toroid, and $i(t)$ is the current to be measured. Parameter μ_0 [54] is the coefficient of *permeability of free space*, which is a measure of how much magnetic field is produced per unit current in vacuum with its value shown below.

$$\mu_0=1.256\times10^{-6}\ \ (N/A^2) \tag{6.7.9}$$

To recover the current flowing through the toroid $i(t)$, an integration with integral constant (gain) k and with infinitely high input impedance can yield the exact result as follows.

$$v_{out}\left(t\right)=-kN\frac{A\mu_0}{2\pi r}\int_0^t\left(\frac{di\left(t\right)}{dt}\right)dt+v_{out}\left(0\right)=-kN\frac{A\mu_0}{2\pi r}i\left(t\right)+v_{out}\left(0\right) \tag{6.7.10}$$

so that the current can be calculated by the following equation.

$$i\left(t\right)=\frac{2\pi r}{kNA\mu_0}\left(v_{out}\left(0\right)-v_{out}\left(t\right)\right) \tag{6.7.11}$$

Although equation 6.7.11 indicated that the Rogowski coil can also be used to measure direct current (DC), its basic principle is based on detecting a flux change proportional to a current change. Without knowing current $i(0)$ represented by $v_{out}\left(0\right)$ in equation 6.7.11, it is impossible to reconstruct the DC current component. In practice, integration operation is always limited by the signal bias. A low-pass filter is always used to approximate the integral to get rid of the effect of bias (see Section 6.5 for the case of integrating a rate gyro signal to get angular position). The signal conditioning circuit presented in Figure 6.44 uses a low-pass filter with the following transfer function from $v(t)$ to $v_{out}(t)$ as shown in equation 6.7.12.

$$\frac{V_{out}(s)}{V(s)}=-\frac{1}{R_1}\frac{R_2}{R_2Cs+1} \tag{6.7.12}$$

As resistance R_2 goes to infinity, the transfer function reduces down to an integrator $(1/s)$ with gain $(1/R_1C)$. Normally, a fairly large R_2 is used. To measure the DC current, it has been recently proposed to combine the Rogowski coil with an open-loop magnetic field sensor that provides the DC information [55].

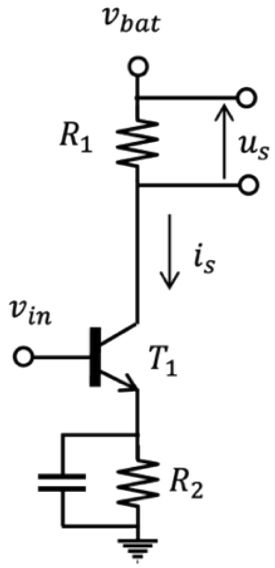
Section Summary

In this section, sensors that were not mentioned in Sections 6.2, 6.3, and 6.4, are briefly discussed with respect to their operational principles. These include temperature, pressure, air flow, current, and voltage sensors. The RTD (resistance temperature device) temperature [9] and pressure sensing [10] are discussed first, followed by the mass air flow [11] sensor using hot-wire air flow sensing technology. Current and voltage sensing technologies widely used in electronics are then addressed with examples, along with the current source technique. These sensing technologies are important for many mechatronic applications.

Section Keywords

Practice Problems

1. Let $R_1 = 5\ k\Omega$, $R_2 = 20\ k\Omega$, and $R_{ref} = 70\ \Omega$. Find the relationship from R_{RTD} to v_{RTD}. Use the relationship shown in Figure 6.33 to plot the relationship from measured voltage to RTD temperature, given $v_{ref} = 5\ V$.
2. For the circuit defined in the right plot of Figure 6.38, derive the relationship from R_v and R_h to output voltage v_{sg}.
3. Let the thermistor resistance and temperature have a relation of $R_{th} = 2\,(T_{th} + 50) + 1200\ \Omega$ for temperature $T \in [-50,\ 200]\ ^\circ\mathrm{C}$, the hotwire resistor $R_{hot} = 0.2\,(T_{hot} + 45) + 120\ \Omega$ for temperature $T \in [-50,\ 200]\ ^\circ\mathrm{C}$, $R_1 = 100\ \Omega$, and $R_2 = 10\ \Omega$. Assume the hot-wire MAP shown in Figure 6.41. If $v_{hot} - v_{th}$ is maintained at zero and $T_{th} = 20\ ^\circ\mathrm{C}$, find the hot-wire resistor temperature T_{hot}.
4. For the circuit below, design a circuit to convert the floating output voltage u_s to the one reference to the ground. Use the circuit shown in Figure 6.35 and then convert the voltage into a current source similar to that in Figure 6.43.



This page titled 6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6.8: Vision-Based Sensing

Learning Objectives

After completing this section, you will be able to:

1. Use the IR camera to detect agricultural spray system fault.
2. Explain the automotive image-based lane detection process.

In Section 6.1, we reviewed an image-based in-cylinder temperature sensing method. We used an example related to internal combustion engine research for auto-ignition control of the combustion process for a homogenous charge compression ignition (HCCI) [14], [15]. In this section, we will provide two more examples for image-based diagnostics and detection.

The first example is a solid-set canopy delivery system (SSCDS). It was developed to deliver water and chemical solutions to high-density fruit trees. For the purpose of detecting SSCDS faults, we propose using the thermal image-based detection method; see Chapter 5 of reference [12]. In the experimental setup, a FLIR thermal camera was mounted on a modified commercial unmanned aerial vehicle (UAV). Thus, any gusher fault, or an emitter nozzle, whether partially plugged or plugged, can be captured by the thermal infrared (IR) camera during the water and chemical spray processes. Due to the nature of the SSCDS, the fault detection mechanism is necessary. Over time, the emitters are easy to clog (plug) or fall off (gusher). To help growers locate faulty emitters in the field without checking emitters one-by-one in person, the proposed thermal fault detection mechanism is based on a thermal IR camera mounted on a UAV. Live thermal videos of the faulty sections can be sent to the user with a video transmitter mounted on the drone.

Because the camera is mounted on a drone, the thermal IR camera should be light in weight and small in size with low operational power. These requirements led to selection the FLIR VUE Pro R, a thermal IR camera for professional drones. This thermal camera, along with a conventional camera, is used for recording the spray process. The thermal IR videos are broken into frames and stitched into panorama pictures for post analysis.

For experiments conducted during a summer morning, the water temperature is usually lower than the atmosphere. Due to the transpiration effect, the trees are slightly cooler than the atmosphere prior to the spray. During the spray, the sprayed area temperature may be even lower than the area not being sprayed. When the outside temperature is low (below $50℉$), the water is hotter than the atmosphere; the tree temperature condition is reversed prior to and during spraying. Note that the thermal camera has a resolution of $\pm 0.2℃$ in distinguishing the temperature differences between objects. The thermal camera is used to pick up the hot or cold spots on trees. These spots will be identified as fault regions. The corresponding thermal footage is then referred to as feasible thermal footage. The flight and spray patterns are the key for a successful fault detection.

For the thermal footage in a black and white pattern, temperature differences are indicated by the level of darkness: darker color means lower temperature and lighter color higher temperature. In this case, experiments were conducted on a cold day with almost no wind. Therefore, the water was hotter than the atmosphere. In this orchard, faults were also manually created on purpose; Table 6.3 lists the locations.

Table 6.3: Emitter faults created.

Fault location in Figure 6.45	Emitter fault type
B1	Plugged
B2	Gusher
B3	Plugged
B4	Gusher

Because these emitters were too close to each other, only the faulted emitters were marked with small pieces of white stripes on top of the irrigation line, as seen in Figure 6.45 (e). The faulted emitters are also marked from B1 to B4 at the corresponding locations in Figure 6.45 (d); see Table 6.3 for the type of fault. In this experiment, thermal plate fusion was used; the associated results are shown on Figure 6.45 (a) and (b) associated with the spray with and without created faults, respectively. A black-white

thermal plate was also used for fault and no-fault spray situations. These are shown in Figure 6.45 (c) and (d), respectively. The fusion video was taken first. The black-white video was taken two hours later after all the sprayed solution was evaporated and the canopy temperature recovered back to normal. The experiment parameters were exactly the same for these two thermal plates.

Comparing Figure 6.45 (a) with Figure 6.45 (c), we can see that even though the experiment parameters were exactly the same for the two thermal plate runs, the figures indicate a huge difference in these two thermal plates. The black-white footage shows a clearer result, with a larger faulted area than the thermal fusion one, indicating that for this thermal camera, the black-white plate has a better sensitivity than the fusion plate.

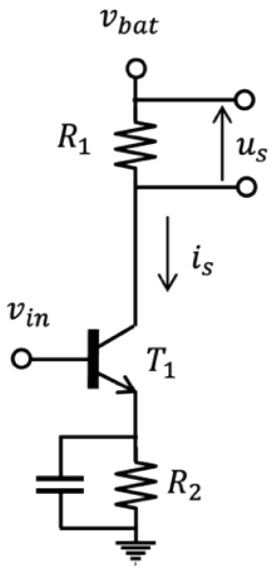


Figure 6.45: Thermal and regular images of the orchard.

From Figure 6.45, we can easily detect the gusher effects using the thermal camera by observing the hot areas due to higher flow rate of hot solution (water) with normal emitter. However, the plug effects are not even visible in the thermal footage. But with known locations for the plugged emitters, these plugged (missing) emitters can be detected.

The second image-based sensing is for road lane detection based on forward video images. Lane centering is also known as lane centering assist, lane assist, auto steer, or autosteer. As a feature in automated driving, it is an advanced driver-assistance system used to keep a moving vehicle remaining at the center of the lane, relieving the driver of the task of steering. Lane centering keeps the car centered in the lane. It is similar to lane-departure warning and lane-keeping assistance that warn the driver and keep the car away from the lane edge, respectively. Together with adaptive cruise control (ACC), keeping a safe distance from the vehicle in front, this feature could enable unassisted driving for some length of time. It is also part of automated lane keeping system.

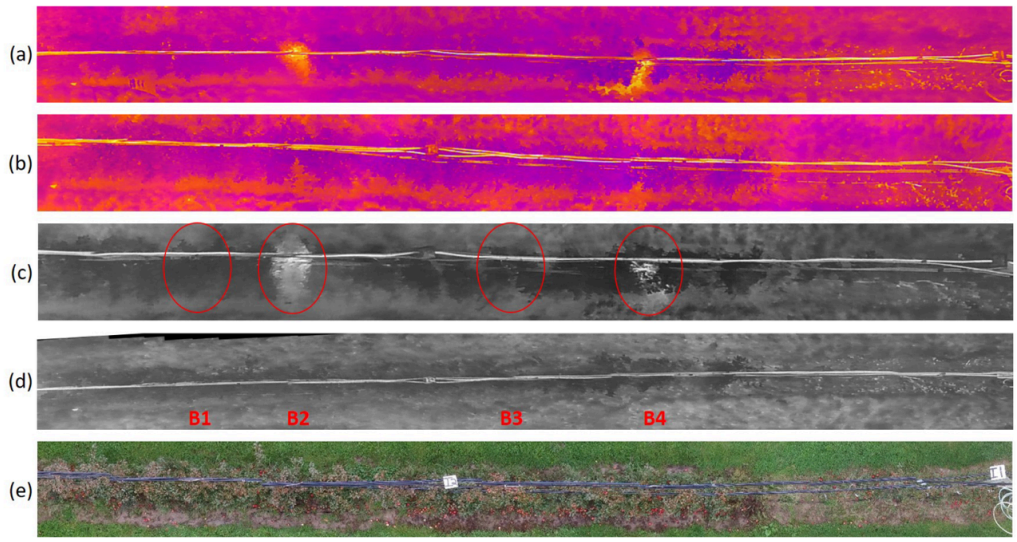


Figure 6.46: Framework of the minimalistic lane detection approach.

We will use the framework of the minimalistic approach described in [13] to explain the process of lane detection shown in Figure 6.46. In reference [13], color segmentation is used to detect objects or parts of the image that are of a particular color. The algorithm processes only sections of the image that have the specified colors. This approach is particularly useful for extracting information for lane markings; around the world, lane markings are dominantly painted in either yellow or white. The image-processing involves loading the image, then converting it from its original RGB color space to a hue, saturation, and lightness (HLS) color space. Note that RGB is an additive color model, meaning it adds red, green, and blue light together to create different colors. The process further splits the space into its color bands of hue, lightness, and saturation. Canny edge detector [56] is then used to produce the best edge image. Considering noise contained in any real-world image, smoothing filter is used to reduce image noise. Then, the Hough transform, a popular computer vision technique, is used for detecting lines, and linear regression is employed to obtain smooth lines (detected lane); see [13] for a detailed approach.

Section Summary

This section addresses a class of sensing technology that is image-based. For example, camera images and videos are used to diagnose water and chemical spray system in an orchard [12] and to detect road lanes for autonomous driving [13]. These applications use one or more sensing components (e.g., a camera or a camera plus IR video camera). The resulting images and videos can be used to sense multiple objects such as spray situations, lane locations, and type of front-moving objects. Sensor fusion technology is often used to combine multiple sensing sources (multiple cameras plus other sensors such as laser and radar) to estimate one sensing object (such as lane locations) to improve estimation accuracy.

Section Keywords

Image/video-based sensor: This device detects and converts light waves or other electromagnetic radiation into electrical signals to form a digital image or video sequence in the information domain.

Infrared (IR) camera: Also called a thermal camera, this device detects and captures images of infrared radiation emitted by objects, which is invisible to the human eye.

Infrared (IR) image: This visual representation is created by capturing infrared radiation, which is a form of electromagnetic radiation invisible to the human eye.

Lane detection: This computer vision process identifies and localizes lane markings on a road, typically for autonomous or semi-autonomous driving systems.

Practice Problems

1. Select one correct statement from the following list. The thermal camera has a resolution of
   a. $\pm 0.2℃$ in distinguishing the temperature differences for the solid-set canopy delivery system.
   b. $\pm 0.5℃$ in distinguishing the temperature differences for the solid-set canopy delivery system.
2. Read reference [13] and describe the minimalistic lane detection process in detail.

This page titled 6.8: Vision-Based Sensing is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

6.9: Chapter References

References

1. M. Chelly and N. Samama, “Detecting visibility in heterogeneous simulated environments for positioning purposes,” in *Proc. 2010 Int. Conf. on Indoor Positioning and Indoor Navig. (IPIN),* Hoenggerberg, Switzerland, Sep. 2010.
2. R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 12th ed., Englewood Cliffs, NJ, USA: Prentice Hall, 2011, ISBN-10: 0-13-602458-0.
3. C. F. Winder, “Shaft angle encoders afford high accuracy,” *Electron. Industry,* vol. 18, no. 10, pp. 76–80, Oct. 1959.
4. F. I. T. Petrescu, “Improving medical imaging and blood flow measurement by using a new Doppler effect relationship,” *Amer. J. of Eng. Appl. and Sci.,* vol. 8. No. 4, pp. 582–588, Oct. 2015, doi: 10.3844/ajeassp.2015.582.588.
5. IEEE Std 1431-2004, “Coriolis Vibratory Gyroscopes,” Accessed: Dec. 29, 2005. [Online] available: https://ieeexplore.ieee.org/abstract/document/1394849.
6. V. Vali and R. W. Shorthill, “Fiber ring interferometer,” *Appl. Opt.,* vol. 15, no. 5, pp. 1099–1100, May 1976, doi: 10.1364/AO.15.001099.
7. W. Elmenreich, “Sensor fusion in time-triggered systems,” Ph.D. thesis, Vienna University of Technology, Vienna, Austria, p. 173, 2002.
8. MPU-6000 and MPU-6050 Product Specifications (Rev. 3.4), Accessed: Nov. 22, 2024. [Online] Available: https://invensense.tdk.com/wp-conten...Datasheet1.pdf
9. D. P. Jones, *Sensor Technology Series: Biomedical Sensors*, New York, NY, USA: Momentum Press, 2010, ISBN-13: 978-1-6065-0056-9.
10. B. W. Bequette, *Process control: modeling, design, and simulation,* Hoboken, NJ, USA: Pearson, 2003, p. 735, ISBN-13: 978-0-1335-3640-9.
11. Mass air flow sensor (Internet Archive: Wayback Machine), Accessed: Nov. 22, 2024. [Online] Available: https://web.archive.org/web/20100114.../forms/h34.pdf
12. Y. Wang, “Sample and hold inputs theory and application, fault detection for SSCDS,” Ph.D. dissertation, Michigan State University, MI, USA, 2019.
13. R. Muthalagu, A. Bolimera, and V. Kalaichelvi, “Lane detection technique based on perspective transformation and histogram analysis for self-driving cars,” *J. of Comput. and Elect. Eng.,* vol. 85, p. 106653, Jul. 2020.
14. Y. Yoon, Z. Sun, S. Zhang, and G. Zhu, “A control-oriented two-zone charge mixing model for HCCI engines with experimental validation using an optical engine,” *ASME J. of Dyn. Syst., Meas., and Control,* vol. 136, no. 4, p. 041015, Jul. 2014, doi: 10.1115/1.4026660.
15. S. Zhang, R. Song, G. Zhu, and H. Schock, “Model-based control for mode transition between SI and HCCI combustion,” *ASME J. of Dyn. Syst., Meas., and Control,* vol. 139, no. 4, p. 041004, Feb. 2017, doi:10.1115/1.4035093.
16. GT-Power — Industrial Leading Engine Simulation Software, Accessed: Nov. 3, 2024. [Online] Available: https://www.gtisoft.com/gt-power/
17. J. D. Bronzino, “Infrared Camera Characterization,” in *Medical Devices and Systems*, Boca Raton, FL, USA: CRC Press, 2006, ch. 38, ISBN-13: 978-0-4291-2304-7.
18. R. Schiessl, A. Dreizler, and U. Maas, “Comparison of Different Ways for Image Post-Processing: Detection of Flame Fronts,” *SAE Tech. Paper 1999-01-1178,* 1999.
19. A. Jain, M. Murty, and P. Flynn, “Data Clustering: A Review,” *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.
20. E. Ramsden, *Hall-Effect Sensors: Theory and Applications*, 2nd ed., Burlington, MA, USA: Elsevier, 2006, ISBN-13: 978-0-7506-7943-3.
21. K. Matsubara, W. Stork, A. Wagner, J. Drescher, and K. D. Müller-Glaser, “Simultaneous measurement of the velocity and the displacement of the moving rough surface by a laser Doppler velocimeters,” *Appl. Opt.,* vol. 36, no. 19, pp. 4516–4520, 1997.
22. J. Cimbala, “Linear velocity measurement (pages 4–5),” *Penn State College of Engineering*, Accessed: Nov. 5, 2024. [Online] Available: https://www.me.psu.edu/cimbala/me345...easurement.pdf
23. P. Regtien and E. Dertien, *Sensors for Mechatronics*, 2nd ed., Amsterdam, Netherland: Elsevier, 2018, IBSN-13: 978-0-1281-3810-6.
24. R. F. Harrington, *Introduction to Electromagnetic Engineering*, 1st ed., Mineola, NY, USA: Dover Publications, 2003, p. 43, ISBN-10: 0-48-643241-6.
25. R. Bergh, H. Lefevre, H. Shaw, “An overview of fiber-optic gyroscopes,” *J. of Lightw. Technol.,* vol. 2, no. 2, pp. 91–107, Mar. 1984.
26. W. A. Gill, I. Howard, I. Mazhar, and K. McKee, “A review of MEMS vibrating gyroscopes and their reliability issues in harsh environments,” *MDPI J. of Sens.,* vol. 22, p. 7505, Sep. 2022, doi: 103390/s22197405.

27. C. M. Graney, “Coriolis effect, two centuries before Coriolis,” *Phys. Today,* vol. 64, no. 8, pp. 8–9, Aug. 2011, doi: 10.1063/PT.3.1195.
28. A. K. Singh, “Piezoelectric gyro sensor technology,” *Def. Sci. J.,* vol. 57, no. 1, pp. 95–103, Jan. 2007.
29. N. Beverini, A. Di Virgilio, J. Belfi, A. Ortolan, K. U. Schreiber, A. Gebauer, and T. Klügel, “High-accuracy ring laser gyroscopes: earth rotation rate and relativistic effects,” *J. Phys.: Conf. Ser.,* vol. 723, p. 012061, 2016, doi: 10.1088/1742-6596-723-1-012061.
30. R. Paschotta, “Ring Laser” in *Encyclopedia of Laser Physics and Technology,* Hoboken, NY, USA: Wiley, 2008, ISBN-13: 978-3-5274-0828-3.
31. E. J. Post, “Sagnac effect,” *Rev. of Mod. Phys.,* vol. 39, no. 2, pp. 475–493, 1967, doi: 10.1103/RevModPhys.39.475.
32. S. Maybeck, *Stochastic Models, Estimating, and Control*., 1st ed., River Edge, NJ, USA: Academic Press, Jun. 1979, ISBN-13: 978-0-1248-0701-3.
33. R. E. Kalman, “A new approach to linear filtering and prediction problems,” *ASME J. of Basic Eng.,* vol. 82 (Series D), pp. 35–45, 1960.
34. P. Zarchan and H. Musoff, *Fundamentals of Kalman Filtering: A Practical Approach*, Reston, VA, AIAA (Amer. Inst. of Aeronaut. and Astronaut., Inc.), Aug. 2001, ISBN-13: 978-1-5634-7455-2.
35. L. Ryzhkov, “Complementary filter design for attitude determination,” in *Proc. 2018 (5th) IEEE Int. Conf. Methods Syst. Navig. Motion Control (MSNMC)*, Kiev, Ukraine, Oct. 2018, pp. 214–217, doi: 10.1109/MSNMC.2018.8576177.
36. J. B. Angell, S. C. Terry, and P. W. Barth, “Silicon micromechanical devices,” *Sci. Am.,* vol. 248, no. 4, pp. 44–55, Apr. 1983.
37. A. Beliveau, G. T. Spencer, K. A. Thomas, and S. L. Roberson, “Evaluation of MEMS capacitive accelerometers,” *IEEE Des. Test of Comput.,* vol. 16, no. 4, pp. 48–56, Dec. 1999, doi: 10.1109/54.808209.
38. J. Iannacci, *RF-MEMS Technology for High Performance Passives: The Challenge of 5G mobile applications*, Bristol England: IPO Publishing, Nov. 2017, ch. 1, doi: 10.1088/978-0-7503-1545-6ch1 and ISBN-13: 978-0-7503-1545-6 [Online].
39. R. Ghodssi and P. Lin, *MEMS Materials and Processes Handbook*, New York, NY, USA: Springer, 2011, ISBN-13: 978-0-3874-7316-1.
40. H. Petroski, *Invention by Design; How Engineers Get from Thought to Thing*, Cambridge, MA, USA: Harvard University Press, Sep. 1996, ISBN-13: 978-0-6744-6368-4.
41. M. Liu, J. Peng, et al., “Two-dimensional modeling of the self-limiting oxidation in silicon and tungsten nanowires,” *Theor. Appl. Mech. Lett.*, vol. 6, no. 5, pp. 195–199, 2016, doi: 10.1016/j.taml.2016.08.002.
42. G. T. Kovacs, N. I. Maluf, and K. E. Petersen, “Bulk micromachining of silicon,” *Proc. IEEE,* vol. 86, no. 8, pp. 1536–1551, Aug. 1998, doi: 10.1109/5.704259.
43. J. M. Bustillo, R. T. Howe, and R. S. Muller, “Surface micromachining for microelectromechanical systems,” *Proc. IEEE,* vol. 86, no. 8, pp. 1552–1574, Aug. 1998, doi: 10.1109/5.704260.
44. M. Tilli, M. Paulasto-Krockel, M. Petzold, H. Theuss, T. Motooka, and V. Lindroos, *Handbook of Silicon-Based MEMS Materials and Technologies,* 3rd ed., Amsterdam, Netherlands; Elsevier, 2020, ch. 6, ISBN-13: 978-0-1281-7786-0.
45. J. A. Becker, “Properties and uses of thermistor-thermally sensitive resistors,” *Bell Syst. Tech. J.,* vol. 26, no. 1, pp. 170–212, Jan. 1947, doi: 10.1002/j.1538-7305.1947.tb01314.
46. C. Goupil, H. Ouerdane, K. Zabrocki, W. Seifert, N. F. Hinsche, and E. Müller, “Thermodynamics and thermoelectricity,” in *C. Goupil (ed.), Continuum Theory and Modeling of Thermoelectric Elements,* New York, NY, USA, Wiley-VCH, Jan. 2016, pp. 2–3, ISBN-13: 978-3-5274-1337-9.
47. R. J. Widlar, “An exact expression for the thermal variation of the emitter-based voltage of bipolar transistors,” *Proc. IEEE,* vol. 55, no. 1, pp. 96–97, Jan. 1967, doi: 10.1109/PROC.1967.5396.
48. J. Martin, W. Bacher, O. F. Hagena, and W. K. Schomburg, “Strain gauge pressure and volume-flow transducers made by thermoplastic molding and membrane transfer,” in *Proc. IEEE Eleventh Annu. Int. Workshop on Micro Electro and Mech. Syst.*, Heidelberg, Germany, Jan. 25–29, 1998, pp. 361-366, doi: 10.1109/MEMSYS.1998.659783.
49. S.-Y. Kim, K.-H. Kim, and Y. Kim, “Comparative study on pressure sensors for sloshing experiment,” *J. of Ocean Eng.,* vol. 95, pp. 199–212, Jan. 2015, doi: 10.1016/j.oceaneng.2014.11.014.
50. L. Chen and M. Mehregany, “A silicon carbide capacitive pressure sensor for in-cylinder pressure measurement,” *Sens. and Actuators A: Phys.,* vol. 145, pp. 2–8, Jul. 2008, doi: 10.1016/j.sna.2007.09.015.
51. T. Nagata, H. Terabe, S. Kuwahara, S. Sakurai, O. Tabata, S. Sugiyama, and M. Esashi, “Digital compensated capacitive pressure sensor using CMOS technology for low-pressure measurements,” *Sens. and Actuators A: Phys.*, vol. 34, no. 2, pp. 173–177, Aug. 1992, doi: 10.1016/0924-4247[GrindEQ__92_]80189-A.
52. D. G. Pellinen, M. S. DiCipua, S. E. Sampayan, H. Gerbracht, and M. Wang, “Rogowski coil for measuring fast, high level pulsed currents,” *Rev. of Sci. Instrum.,* vol. 51, pp. 1535–1540, Nov. 1980, doi: 10.1063/1.1136119.
53. S. Zieler, R. C. Woodward, H. H-C Iu, and L. J. Borle, “Current sensing techniques: a review,” *IEEE Sens. J.,* vol. 9, no. 4, pp. 354–376, Apr. 2009, doi: 10.1109/JSEN.2009.2013914.
54. 2022 CODATA value: vacuum magnetic permeability from the NIST reference on constants, units, and uncertainty, Accessed: December 30, 2025. [Online] Available: https://physics.nist.gov/cgi-bin/cuu/Value?mu0
55. L. Dalessandro, N. Karrer, M. Ciappaz, A. Castellazzi, and W. Fichtner, “Online and offline isolated current monitoring of

parallel switched high-voltage multi-chip IGBT modules," in *Proc. 39th IEEE Power Electron. Specialists Conf. (PESC'08)*, Rhodes, Greece, Jun. 15–19, 2008, pp. 2600–2606.
56. X. Yan and Y. Li, "A method of lane edge detection based on Canny algorithm," in *Proc. 2017 Chinese Automation Congress (CAC)*, Jinan, China, October 20-22, 2017, pp. 2120–2124.

Solutions to Practice Problems

1.

$$i_a = \frac{\tau}{K_\tau} = \frac{50}{2.0} = 25\ (A) \tag{6.9.1}$$

As a result, the resulting energy consumed by the current sensing resistor R_c can be calculated as follows.

$$e_c = R_c i_a^2 = 0.02 \times 25^2 = 12.5\ (W) \tag{6.9.2}$$

Therefore, the energy wasted on the current sensing resistor is significantly reduced, compared with Example 6.1.

1.
2. The following is the plot of signals A and B of the incremental encoder rotating in the anticlockwise direction.

1.
2. The answer is c).

1.
2. The answer is c).

1.
2. The answer is b).

1.
2.

$$f_c = 2 \times 0.0025 \times 10 \times 10^{-5} \times 2\pi \times 10^5 \sin 2\pi 10^5 t \ = 0.1\pi \sin 2\pi 10^5 t \quad (Nm) \tag{6.9.3}$$

1.

1.

After sensor fusion, the variance of the fused estimate is reduced significantly in this case.

1.

Therefore, the cutoff frequency of the equivalent continuous time low-pass filter is 4.08 rad/s. In addition, the equivalent cutoff frequency reduces as α approaches one.

1. The simulation results with $\alpha = 0.5$ is shown below. The estimated angular displacement through sensor fusion becomes noisier, compared with that in Figure 6.29. Therefore, in this case, properly selecting the value of α is very important for the sensor fusion.
2. The correct statement is a).

1. When the silicon is flexed or bended, there is virtually no hysteresis, leading to almost no energy dissipation or loss. This is perfect for the vibration-based rate gyros discussed in Section 6.4.
2. Substituting given parameters into equation ??? yields the following map. We can see that both output voltage and the minimal voltage are moved.

1.
2. The following figure plots the temperature and resistance relations of both the thermistor and hot-wire resistor. First, we draw a vertical line at 20\(\mathtt
ParseError: invalid DekiScript (click for details)

```
Callstack:
    at (Bookshelves/Introductory_Engineering/Mechatronics:_Fundamentals_Design_Integration_
```

\) on the thermistor plot, map the intersection horizontally to the hot-wire plot, and draw the vertical line through the intersection on the right plot. We then have that if $v_{hot} - v_{th} = 0$, the temperature of the hot-wire resistor is 25\(\mathtt
ParseError: invalid DekiScript (click for details)

```
Callstack:
    at (Bookshelves/Introductory_Engineering/Mechatronics:_Fundamentals_Design_Integration_
```

\), which is 5\(\mathtt
ParseError: invalid DekiScript (click for details)

```
Callstack:
    at (Bookshelves/Introductory_Engineering/Mechatronics:_Fundamentals_Design_Integration_
```

\) higher than the thermistor's temperature

3. The following circuit converts the floating ground voltage v_s to grounded voltage v_{sg}, to the current source $i_s = v_{sg}/R_4$, and then back to $v_{s-A/D}$. The red circuit is provided by P6.27.
4. The solution is a).

1. Discuss in detail regarding these five blocks in Figure 6.46 by reading reference [13]. Explain the inputs and outputs of these five blocks: color segmentation, Canny edge detection, smoothing filter, Hough transform, and linear regression.

This page titled 6.9: Chapter References is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

7: MECHATRONIC CONTROL SYSTEMS

Learning Objectives

1. Describe the mechatronic system software and hardware architecture along with the concept of open- and closed-loop control systems. Show that a mechatronic system may contain multiple open- and closed-loop systems.
2. Explain the difference between microprocessors and microcontrollers. Introduce the general architecture (structure) of a mechatronic microcontroller. Be able to select a proper microcontroller for the mechatronic system application.
3. Discuss the Arduino family of microcontrollers, especially Arduino MKR 1000 Wi-Fi microcontrollers. Review MATLAB and Simulink software used to program the microcontroller in addition to IDE (integrated development environment) software.
4. Describe the control system hardware and software architectures from an application example of DC-motor speed control utilizing an Arduino MKR 1000 Wi-Fi microcontroller board, H-bridge pulse-width modulation motor control hardware, and hall-effect speed sensor input.
5. Provide the concept of the iterative development process, called V-design, used for developing a mechatronic system through both model-based simulation and prototyping validation.

This chapter provides an introduction to the mechatronic system architecture. It also reviews the associated system hardware, including sensors, actuators, and both open- and closed-loop controls. The chapter describes the mechatronic system in a control system setup, the associated control and communication software used in a mechatronic system, and a design process for mechatronic systems.

7.0: Prelude

7.1: Mechatronic System Architecture

7.2: Microcontrollers Used for Mechatronic Systems

7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi

7.4: Mechatronic Control System Architecture- Speed Control Example

7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process

7.6: Chapter References

This page titled 7: Mechatronic Control Systems is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

7.0: Prelude

In earlier chapters, we discussed that a mechatronic system consists of the physical plant, sensors, actuators, and any open- and closed-loop controllers. The physical system (plant), sensors, and actuators belong to physical domain. The control strategy and data processing mechanism are inside the information domain. In this chapter, our focus will be the control systems. We will utilize the techniques introduced in Chapters 2, 3, and 4 to design closed-loop controllers, analyze the closed-loop mechatronic system performance, and then implement the controller into the microcontroller.

Section 7.1 reviews the architecture of a mechatronic system from both structural and hardware points of view. The mechatronic system often uses open- or closed-loop control systems to regulate certain system outputs to its target values [1], [2] or it controls some system outputs to track the desired trajectory [3], [4], respectively. The open- and closed-loop control system architecture and its associated hardware will be discussed by using a few examples. Most mechatronic system components have been introduced in early chapters; this section mainly addresses system level requirements and architectures.

Section 7.2 reviews the microcontroller architecture [5] for implementing the control software. First, it addresses the difference between a microcontroller and a microprocessor. A microcontroller unit (MCU) [5] is essentially a small computer on a single chip. It is designed to manage specific tasks within an embedded system without requiring a complex operating system. By contrast, a microprocessor [6] is a computer processor with the data processing logic and control included on a single integrated circuit (IC) or a small number of ICs. The microprocessor contains the arithmetic, logic, and control circuitry required to perform the functions of a central processing unit (CPU).

Next, the general architecture of a mechatronic microcontroller is discussed. Microcontrollers are the core of mechatronic systems. They can be categorized commonly based on their architecture, data bus width, instructions set, and application scenarios. Last, the section presents the process of selecting a proper microcontroller for designing a mechatronic system. Selecting an appropriate microcontroller for a mechatronic system requires a systematic approach. It is essential to first establish a comprehensive understanding of the mechatronic system requirements and architecture before proceeding with the selection of the microcontroller.

Section 7.3 provides detailed information of a sample microcontroller: Arduino MKR 1000 Wi-Fi [7]. First, we discuss the Arduino family microcontrollers, along with their IDE (integrated development environment) software [8] and the program environment under MATLAB and Simulink [9]. Next, we introduce in detail the Arduino MKR 1000 Wi-Fi microcontroller, including its circuit board, pin layout, detailed electronic circuit, surrounding electronic components, and its core, ARM 32-bit Cortex-M0+ MCU [10]. The intension is to familiarize readers with the microcontroller and its circuit from the hardware point of view and to compares the sample microcontroller (Arduino MKR 1000 Wi-Fi) with the microcontroller architecture discussed in Section 7.2. Last, the MATLAB and Simulink Arduino MKR 1000 Wi-Fi software is discussed in brief along with Arduino UNO Q. A detailed Simulink discussion is provided in Appendices A and B.

Section 7.4 addresses the control system architecture of mechatronic systems. The section will use an example of DC-motor speed control, building on the overall mechatronic system architecture of both hardware and software that are introduced in Section 7.1. We start with the hardware of the DC-motor speed control system, utilizing the Arduino MKR 1000 Wi-Fi microcontroller. In order to make this control system work, an additional DC-motor drive board, the Arduino MKR motor carrier [12], is needed to drive the DC motor and collect the hall-effect speed sensor signal. The Arduino MKR motor carrier board is open source. Its detailed information can be found in [12]. We then present the Simulink motor control software. We show how the motor drive and hall-effect sensor input blocks in the Arduino Simulink support library can be used to develop closed-loop control software utilizing the PID (proportional-integral-derivative) control technique [13].

Section 7.5 introduces a model-based mechatronic system development process, called the V-design process. This process starts with the required mechatronic system specifications or requirements and ends with a design leading to a life-cycle optimized product. For developing the mechatronic system models, this design process starts with a rough mathematical model that leads to a more detailed model, also called a digital twin of the target mechatronic physical system. This model can be used to evaluate closed-loop mechatronic system performances by adding actuator and sensor dynamics, and then considering the detailed physical system dynamics and nonlinearities. A successful design that is validated through simulation now moves to generating a prototype of the mechatronic system, on that is optimized for the system's life cycle.

Orienting Questions

1. What are the software and hardware architectures of a mechatronic system?
2. What is the general system architecture for a microcontroller used in a mechatronic system?
3. What is the Arduino MKR 1000 Wi-Fi microcontroller and its features?
4. What is the V-design process for a mechatronic system?

This page titled 7.0: Prelude is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

7.1: Mechatronic System Architecture

Learning Objectives

After completing this section, you will be able to:

1. Describe the mechatronic system software and hardware architectures and their relationship to the control system design and analysis.
2. Explain the architecture of mechatronic system software.

Chapter 1 introduced a general mechatronic system architecture, which is reproduced in Figure 7.1. The physical and information domains are separated by the domain interface. The control system covers the whole mechatronic system shown in Figure 7.1, where the digital control law is normally implemented in the information domain and executed by the digital computer (microprocessor). The analog control law can be implemented in the physical domain, but it is rarely used nowadays.

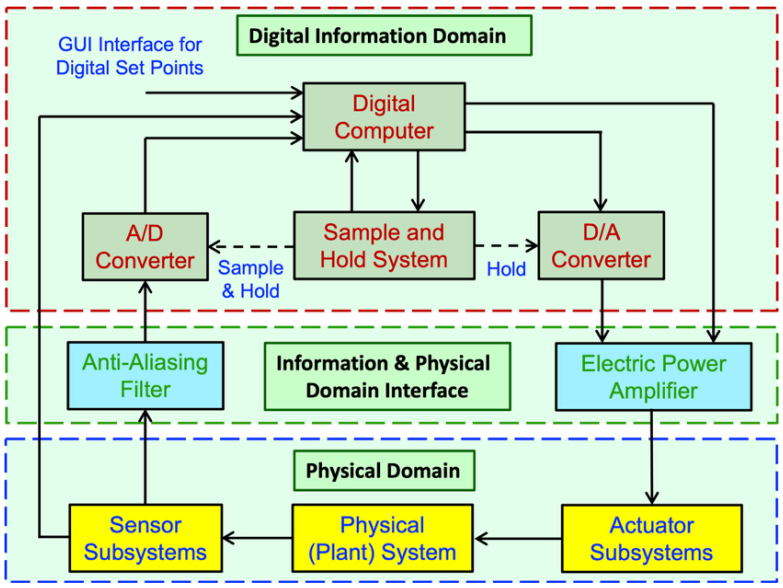


Figure 7.1: General mechatronic system architecture.

We will divide the mechatronic system architecture into two parts: hardware and software. A mechatronic system integrates hardware in the physical domain with software in the information domain. Without a system concept of integrated hardware and software, we cannot design a good mechatronic system with optimal performance.

Let's first discuss the hardware representation inside a mechatronic system and its relation to the mechatronic control system architecture. A mechatronic system may have multiple control systems (open- or closed-loop). We will demonstrate it with an internal combustion engine mechatronic system.

An internal combustion engine can be considered as a complex mechatronic system; see Figure 1 in reference [14]. The engine control module (ECM) controls many engine subsystems. These include the engine air-intake throttle, fuel injection, ignition, variable valve timing, and EGR (exhaust-gas-recirculation) valve position. The ECM receives feedback signals such as engine manifold temperature and pressure, intake air flow, exhaust air-to-fuel ratio and oxygen content, engine position, and speed sensor signals. As a result, the simplified engine mechatronic system consists at least five control systems (see Figure 7.2).

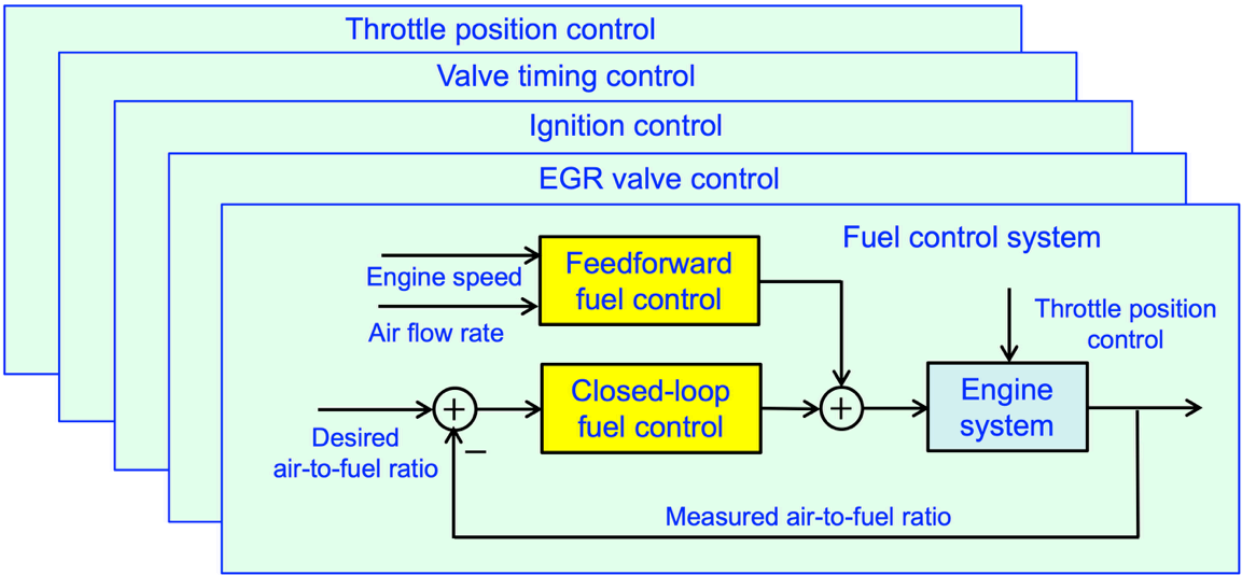


Figure 7.2: Multiple control systems inside an engine mechatronic system.

These five control subsystems could have different update (sample) rates. For example, the ignition and fuel control loops are often event-based depending on the engine crank position (or speed). Meanwhile, the intake and exhaust valve timings, EGR rate, and throttle position controls are time-based with a fixed sample timing. The sensor signals can also be divided into groups. For time-based sensor and event-based signals, the time-based signals are sampled with specific (fixed) sample rates. For event-based signals such as hall-effect sensors, the engine crank and cam shaft positions are sampled (trigged) based on the shaft position. Therefore, the sensor signals used inside a mechatronic control system can have quite different update rates. Most existing discrete-time control theories assume that all sensor signals are updated in the same rate. However, in most practical applications, the sample rates of sensor signals in a mechatronic system are different.

Some sensor signals are sampled through analog-to-digital (A/D) converters. Some are sampled through digital channels directly by counting the number of pulses in a given time window or measuring the time duration for an active pulse. Others are sampled through serial communications such as SPI (serial peripheral interface) and I2C (inter-integrated circuit). As a result, different sensor sample rates in a control feedback loop can lead to system modeling error; most available analysis tools are for a single sample rate.

Remark 7.1

As a rule of thumb, for a mechatronic system with multiple sample rates for sensor signals, the slowest sample rate should satisfy the Shannon-Nyquist sample theory (see Section 4.5). For practical applications, the slowest sample frequency should be at least ten times the highest frequency component in the mechatronic system in the physical domain. easy to remember and apply in solving problems, as long as you get the **proper values a**

Example 7.1

Consider the engine valve timing control mechatronic subsystem shown in Figure 7.2. Assume that the valve timing actuator is an electric cam phaser [15]. In reference [16], closed-loop system identification, based on the system identification approach described in Section 2.6, was conducted to obtain a linearized system model. Table 6 of reference [16] provides an identified system model in the following form.

$$G(s) = \frac{\Theta_{cam}(s)}{U_m(s)} = \frac{1.6 \times 10^{-5} s^2 + 0.0032s + 1.0}{s^2 + 1.0s + 0.99} \tag{7.1.1}$$

where $\Theta_{cam}(s)$ and $U_m(s)$ are the actual cam phase angle and electric cam phaser control input, respectively. The above model can be simplified as follows by setting the coefficient of the square term in the numerator of the transfer function to zero.

$$G(s) = \frac{\Theta_{cam}(s)}{U_m(s)} \approx \frac{0.0032s + 1.0}{s^2 + 1.0s + 0.99} \tag{7.1.2}$$

Based upon the transfer function shown in equation 7.1.2, the bandwidth of the electric cam phase system can be calculated based upon the second-order system transfer function described in Section 3.3.

$$\omega_n = \sqrt{0.99} = 0.995 \ \ (rad/s) = 0.158 \ (Hz) \tag{7.1.3}$$

The position sensor in the cam phase system is based on the hall-effect sensing principle (see Section 6.2) with a wheel that generates four pulses per revolution (n_c). Let's calculate the position update (sample) rate as a function of engine speed. Each crank-shaft rotation results in a half rotation of the engine cam shaft for four-stroke engines. Therefore, the number of pulses (n_p) generated by a cam position sensor can be calculated below as a function of engine speed n_e in RPM (rotation per minute).

$$n_p = \frac{4 \times n_e}{2 \times 60} \ \ (Hz) \tag{7.1.4}$$

In this case, n_p represents the position sample rate. Consider an engine operated between 200 and 6000 RPM. the slowest sample rate in this case ($at \ n_e = 200 \ RPM$) can be calculated as follows.

$$n_p \, (200 \ RPM) = \frac{n_c \times n_e}{2 \times 60} = \frac{4 \times 200}{2 \times 60} = 6.665 \ \ (Hz) \tag{7.1.5}$$

which is more than 42 times the system bandwidth. This indicates there is no issue on sampling frequency over the entire engine operational speed range. As the engine speed increases, the sample frequency also goes up.

The engine combustion system is in fact a multi-input and multi-output (MIMO) mechatronic system; see Section 3.5. For such a MIMO system, we can have individual single-input and single-output (SISO) control loops to control individual targets by ignoring the coupling effect within the physical system. The advantage of using a single loop SISO controller is its simplicity. In this case, however, the control performance could be degraded due to lack of coordination among control signals. Yet, MIMO control can also be used to improve the control performance at the expense of control system design and tuning complexity. In Chapters 8 and 9, we will compare the performances of SISO individual loops and MIMO systems for mini-Segways and three-wheel carts.

Remark 7.2

For a complex mechatronic system with interacted dynamics, both SISO and MIMO control strategies can be used. The advantage of the SISO strategy is its simplicity and the feasibility of adopting a PID (proportional-integral-derivative) control strategy. For highly interactive mechatronic systems, MIMO control strategy improves closed-loop control performance by considering the interactions inside the physical system.

Next, we will connect the control system diagram (e.g., Figure 7.2) and the hardware architecture of a mechatronic system. Consider the control system diagram on the top layer of Figure 7.2. We can see that three sensors and one actuator are used for the closed-loop control system. The engine mass air flow sensor and engine speed sensor signals are used for generating the feedforward control. The air-to-fuel ratio sensor (also called oxygen sensor) is used for feedback control to regulate the engine air-to-fuel ratio. Figure 7.3 shows a control system hardware architecture of the engine air-to-fuel ratio control shown in Figure 7.2 as a feedback control system block diagram.

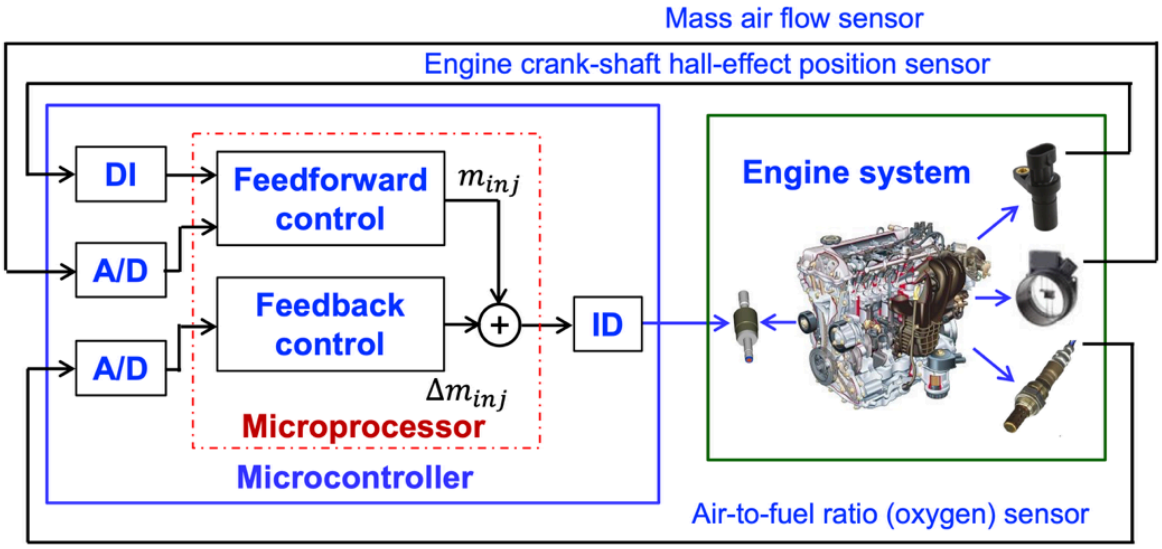


Figure 7.3: Hardware architecture of the engine air-to-fuel ratio control system.

From Figure 7.3, we can see three sensors and one actuator inside the engine system block. The microcontroller block contains a microprocessor, two A/D (analog-to-digital) conversion channels, one digital input (DI) channel, and one injection drive (ID) channel. Let's consider their functionalities.

The engine crank-shaft hall-effect position sensor generates a square-wave pulse train (see Section 6.2). This pulse train is input to the microcontroller through the digital input channel. The digital channel uses a digital counter to count the number of clock pulses enabled when the position pulse train is at "high." The counted number is then converted to engine speed using the method described in Section 6.2. In addition, the engine position signal can be calculated using both crank- and cam-shaft position sensors. The output of the mass air flow sensor is an analog voltage signal. It can be sampled into the microcontroller by its A/D channel. The engine speed signal is calculated from the digital-input channel (based on the crank-shaft position sensor) and the mass-air-flow signal from the A/D channel (based on the mass air flow sensor). The feedforward control signal m_{inj} can then be calculated inside the microprocessor; the fuel flow rate $\dot{m}_{fuel}$ is calculated to maintain the normalized air-to-fuel ratio equal to one based on the measured engine mass air flow rate. The calculated $\dot{m}_{fuel}$ is converted into the fuel injection mass m_{inj} based on the current engine speed and number of engine cylinders. Maintaining the normalized air-to-fuel ratio at one is critical so that the three-way catalyst can be used for emission reduction.

For the feedback control, the analog voltage signal of the air-to-fuel ratio (oxygen) sensor is converted into the digital signal in the microprocessor. An error signal is generated by subtracting the measured normalized air-to-fuel ratio signal from the desired one. A PID controller is used to generate the closed-loop fuel control signal Δm_{inj}.

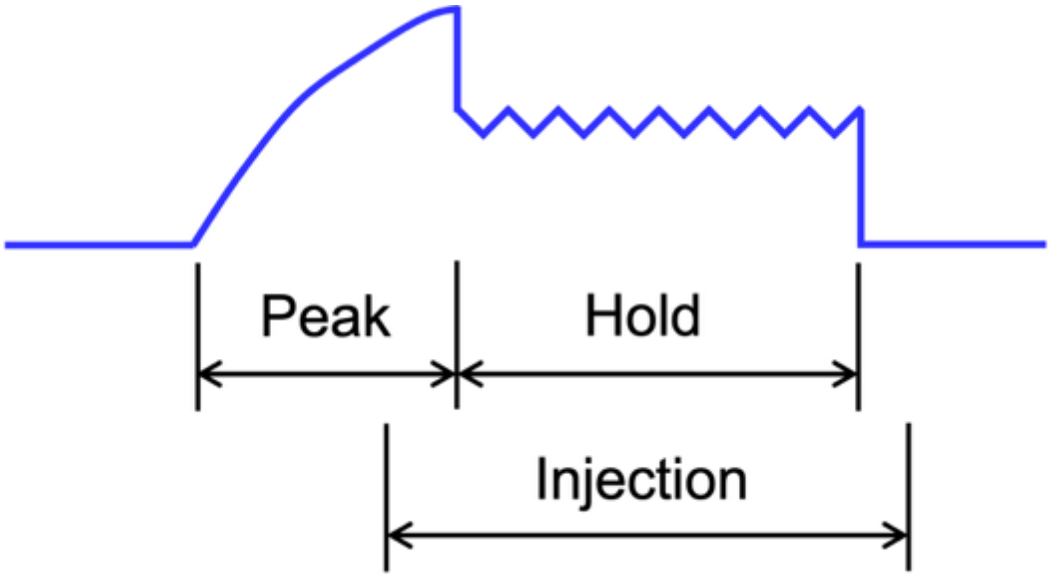


Figure 7.4: Peak and hold current profile of an injection drive.

We calculate the total fueling signal by taking the summation of injection mass m_{inj} and closed-loop correction Δm_{inj}. For an internal combustion engine, the fuel injection is event-based; that is, it is based upon the engine crank-shaft position (not based on time). As a result, an event-based fuel injection drive is required (see Figure 7.3). The current profile of an injection drive often has peak and hold characteristics, shown in Figure 7.4, where the fuel injection mass is a function of the injection spray duration. As a result, in the injection drive, the desired injection fuel mass is converted into the injection spray duration and then into the peak and hold durations. Converting a control signal, such as injection fuel mass, into the actuator control signal (injector solenoid current) is a complicated process.

Now, we are ready to discuss the software architecture of mechatronic systems. In general, the software for a mechatronic microcontroller can be divided into three tiers (see Figure 7.5). These are low-level signal input and output software, middle-level signal processing software, and high-level control feature software.

The low-level signal input and output software shown in the lower block of Figure 7.5 is used mainly to program and calibrate a microcontroller's input and output devices (A/D, D/A, digital I/O). The signal input devices have three groups: direct digital inputs, analog inputs through A/D converters, and digital inputs through serial communication buses.

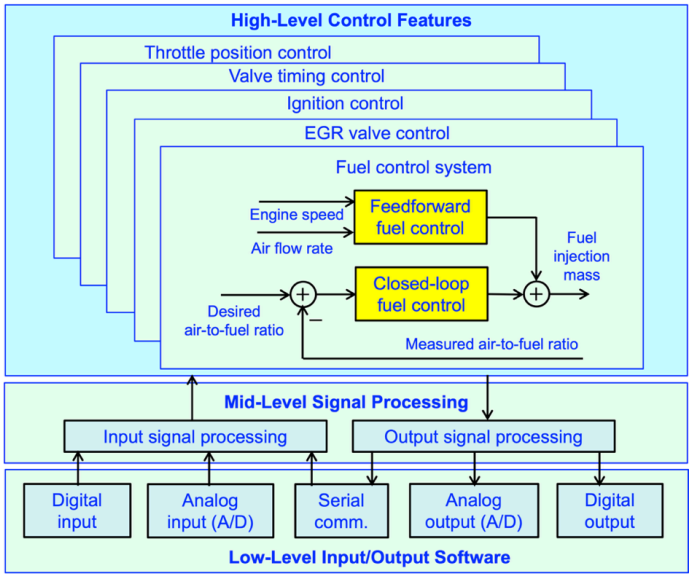


Figure 7.5: Mechatronic system software architecture.

The direct digital input has three configurations. The first is digital signal high and low detection with output of 1 bit. The second

is the number of pulses counted from a pulse train with output of a digital number. The last is pulse train duration counting (or pulse high duration counting) with output of a digital number (see Section 6.2). For each kind of input, we need to define signal resolution (e.g., number of bits for digital counters). These low-level software outputs are used as inputs to the middle level software to generate the sensor signals used in the high-level control feature software. The last input group is the software for serial communication. For example, the three-axis accelerometer and three-axis rate gyro MEMS sensor, MPU 6050, sends the detected accelerations and angular rates through the I2C serial bus; see Section 6.6.

The second part of the low-level software includes the system signal outputs. Similar to the inputs, there are three groups of hardware electronics needed to be programmed and calibrated. They are digital outputs, analog outputs, and serial communication. Digital outputs are normally in the form of low-side (or high-side) analog or PWM (pulse-width modulation) outputs or they are in the form of H-bridge PWM control outputs (see Section 4.6). In this last case, the PWM signal frequency needs to be calibrated in software, where converting a digital output to the corresponding duty cycle is handled in the middle-level software.

For the analog output, the D/A converters need to be calibrated. Output update rates also need to be specified through low-level software. Last, for some microcontrollers, certain calculated outputs can be sent to the actuators through serial communication protocols. For example, the Arduino MKR 1000 microcontroller used in the mini-Segway (see Chapter 8) allows the high-level control feature algorithms and middle-level signal processing algorithms in the host computer. All sensor and actuator signals are sent through a Wi-Fi network in the sense of serial communication. In this case, the data update speed (sample frequency) and digital resolution (data length) need to be configured through low-level software. Chapter 8 details this configuration in the mini-Segway application.

Normally, the microcontroller's manufacturer provides the low-level drive software. The microcontroller input and output hardware and software are programmed during the design and development process. The main low-level programing calibrates and configures the low-level software. If any low-level software needs added features, the user needs to coordinate with the microcontroller's manufacturer.

Next, we will address the middle-level software for signal processing and conditioning. As with the low-level software, the signal conditioning in the middle level software can be divided into two parts: input and output data conditioning software. The input data conditioning has two goals. It converts the digital and analog input signals from the low-level software into variables that can be used in the upper-level software. It also provides conditioned sensor signals for all control features.

In most cases, the lower-level signal is a fixed-point number (see Section 4.2). For example, a twelve-bit A/D converter may produce a 12-bit fixed point number. If the microprocessor adopts the floating-point calculation, the fixed-point number needs to be converted into the floating one. This is also true for the digital input produced by counting the number of pulses. Certain input signals may be needed by control feature algorithms in the upper level. It is efficient to condition the signal in the middle level to eliminate repeated signal conditioning in multiple control features. For example, in the internal combustion engine microcontroller, the filtered engine speed signal is used by most control features. If it were filtered within individual control features, repeated engine speed filtering calculations would exist in each feature algorithm.

Inside the middle-level software, the output signal conditioning converts the control outputs from all control features into variables with proper format so that the lower-level output hardware can handle them. For example, the engine throttle control feature generates a two-way control signal (positive or negative) for the H-bridge PWM drive as part of the low-level drive (see Section 5.3). Assume that the floating-point calculation is used in the throttle control feature to generate a floating control variable. The middle-level software needs to convert the floating number into a fixed-point number with the sign identified. The sign of the converted number is used to determine which pair of switches should be turned on. The magnitude is used to determine the PWM signal duty-cycle for the H-bridge. In addition, the PWM signal period (or frequency) needs to be provided in the middle level software. In short, the middle-level software needs to convert the control feature output variables to the proper format for the low-level software/hardware.

As for the upper-level software, its main task is to complete the control feature calculations based on certain control strategies such as PID control and model-based control strategies. In upper-level control software, each feature deals with different dynamics. The algorithm update rate (control system sample rate) could be quite different. Consider the multiple control features illustrated in Figure 7.5. Some control features need to be updated in a relatively fast rate (e.g., 3–5 ms). These include the engine air-to-fuel ratio, engine ignition control, throttle, and EGR (exhaust-gas-recirculation) valve control. As discussed earlier, the engine cam valve time had relatively slow dynamics; see equation 7.1.3. Its associated control strategy can have a relatively slow update rate (e.g., 10–20 ms). Noticed that all control feature algorithms are sampled (updated) at a fixed rate. Thus, all implemented controllers should be in digital form. If a continuous-time controller is designed, it must be discretized before implementation. However, it is preferred to design the discrete-time controller directly using the discretized system model.

Remark 7.3

For upper-level software used for programming the control features, the sample rate of the control features can be different and selected based upon the bandwidths of the associated mechatronic subsystem dynamics.

Section Summary

This section reviews the architecture of a mechatronic system from both structural and hardware points of view. Mechatronic systems often use open- or closed-loop control systems to regulate certain system outputs to their target values [1], [2] or to control some system outputs to track the desired trajectories [3], [4], respectively. Most mechatronic system components have been introduced in earlier chapters. This section mainly addresses system level requirements and architecture.

Section Keywords

Closed-loop control: It uses feedback signals to automatically adjust system inputs to maintain desired output by continuously monitoring and comparing actual outputs against set points.

Mechatronic system: It consists of mechanical, electrical/electronic, and mechatronic subsystems. A microcontroller (or multiple microcontrollers) in the information domain is often used to collect the sensor and reference signals from a mechatronic system. It then makes decisions based upon the collected signals inside the microcontroller, converts these decisions into actuator commands, and sends them to the related actuators for execution in a closed-loop manner.

Microcontroller: This compact, self-contained integrated circuit (IC) functions as a miniature computer on a single chip, housing a central processing unit, memory, and input/output peripherals designed to control specific functions of an electronic device or embedded system.

Open-loop control: In this non-feedback control method, the control action is independent of the resulting process output. That is, the control does not use output feedback to adjust its operation.

Practice Problems

1. Find the correct answer from the list below to make the statement correct. For a mechatronic system,
    1. Sensor sample rates must be the same.
    2. Sensor sample rates can be different, with the slowest sample rate satisfying the Shannon-Nyquist theorem.

1. Use the information provided in Example 7.1 and assume there are three pulses generated per cam-shaft rotation. Calculate the sampling frequency at 200 RPM.

1. Find the correct answer from the list below to make the statement correct. The internal combustion engine fuel injection current profile can be divided into
    1. Peak and hold periods.
    2. Peak and spray periods.
    3. Spray and hold periods.

1. Find the correct answer from the list below to make the statement correct. The mechatronic microcontroller software can be divided into the following levels:
    1. Low- and high-level.
    2. Low-, middle-, and high-level.
    3. Low- and middle-level.

1. Justify whether the following statements are correct.
    1. The upper-level control feature algorithms shall have the same sample rate.
    2. There are three typical input and output approaches in the low-level software.

This page titled 7.1: Mechatronic System Architecture is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

7.2: Microcontrollers Used for Mechatronic Systems

Learning Objectives

After completing this section, you will be able to:

1. Explain the difference between a microcontroller and a microprocessor.
2. Describe the general architecture of a mechatronic microcontroller.
3. Select a microcontroller for a specific application.

Microcontroller units (MCU) and microprocessor units (MPU) are two kinds of integrated circuits. MCUs are single-chip processors that are valuable in the development of mechatronic systems. They replace antiquated multicomponent central processing units (CPUs) with separate logic, digital and analog I/O units. Microcontrollers and microprocessors differ significantly in component structure, chip architecture, performance capabilities, and applications.

There are several key differences between the two. Microcontrollers combine all the necessary elements of a microcomputer system, along with necessary digital and analog I/O hardware, into a single piece of hardware (silicon). A microcontroller normally does not require additional peripherals or complex operating systems to function whereas a microprocessor does. Both MCUs and MPUs contain at least one CPU. However, microcontrollers also integrate memory, I/O components, and other varied peripherals. Certain microcontrollers include one or more FPGA units (field programmable gate arrays) whereas the MPU does not. FPGA units handle tasks requiring fast signal processing capability. They are integrated circuits providing customers the ability to reconfigure the hardware to meet specific-use case requirements after the manufacturing process.

Cost-effective, small-in-size, and low-power microcontrollers are optimized for all-in-one (single chip) functionality. They are best used for specific mechatronic applications like automotive engine control units, infotainment systems, and Internet-of-things (IoT). Meanwhile, general-purpose microprocessors are typically more powerful. They are designed to be supported by specialized hardware for increased performance in high-demand applications such as personal computing and graphics processing.

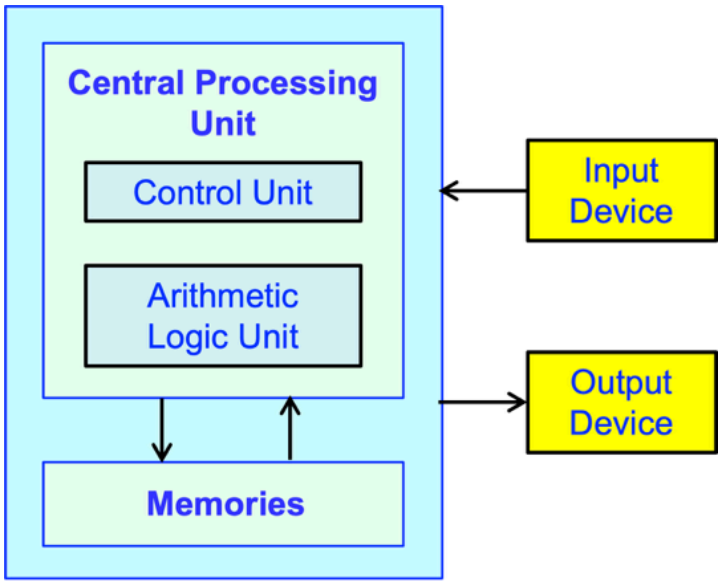


Figure 7.6: Microprocessor with von Neumann architecture.

Microprocessor hardware is based on the classical von Neumann architecture [17]. It consists of a CPU with both an arithmetic logic unit (ALU) and processor registers (small amounts of fast memory storage for quick data access). It also includes a control unit, memory for data and instructions, external memory for mass storage, and I/O mechanisms. Figure 7.6 provides a rough illustration. This methodology uses the same set of interconnecting wires (known as a bus) to transmit instructions and perform operations. Microprocessors cannot perform these actions simultaneously. However, modern devices use various mitigation techniques to avoid data transmission bottlenecks.

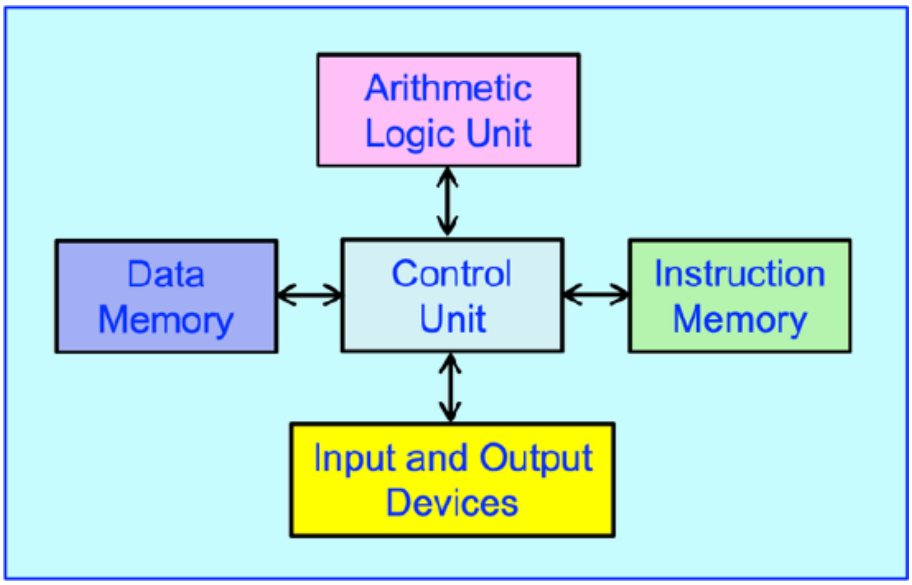


Figure 7.7: Microprocessor with Harvard architecture.

By contrast, microcontrollers with real-time applications use the complex Harvard architecture [18]; see Figure 7.7 for an illustration. They have a dedicated set of data buses and address buses for reading and writing data to memory. They have an additional set to fetch instructions for performing operations. Because the CPU can read an instruction and access data memory at the same time, the Harvard architecture can perform basic operations faster than the von Neumann architecture. As a result, the Harvard architecture has an advantage for real-time and high-speed computing applications. In addition, the unified data and instructions memory space of the simplified von Neumann architecture can improve reliability and scalability. Therefore, von Neumann-based microprocessors are often used for high demanding applications such as high-performance computing (HPC) and gaming, whereas Harvard architecture is typically used in microcontrollers to handle fast signal processing.

As Figure 7.7 shows, the Harvard architecture is a computer architecture with separated data memory and instruction memory pathways. Contrasted with the von Neumann architecture, in which instruction and data memories sharing the same bus, the Harvard architecture uses two independent buses for instruction and data memories. It is suitable for real-time and low-power applications. The ALU in Figure 7.7 is a combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers.

A modified Harvard architecture has a relaxed separation between instruction and data memory buses to allow the CPU or ALU concurrent access to two (or more) memory buses. The most common modification includes separate instruction and data caches using a common address space. When the CPU or ALU executes from cache, it acts as a pure Harvard machine. When accessing backing memory, it acts like a von Neumann machine. As a result, code (instruction) can be moved around like data, leading to a powerful technique. This modification is widespread in modern processors, such as the widely used ARM architecture [19] and x86 processors.

Remark 7.4

Microprocessors and microcontrollers are quite different, as summarized below.

1. Both microcontroller and microprocessor units contain at least one central processing unit (CPU). However, microcontrollers also have integrated memory, input/output components, and other varied peripherals.
2. The CPUs have different architecture. The CPUs in a microprocessor often utilize the von Neumann architecture whereas those in a microcontroller use the Harvard architecture.

Let's now concentrate on the architecture of microcontrollers for mechatronic systems. These microcontrollers have three key component groups.

a. **Central processing unit (CPU):** Often referred to as the "brain" of the microcontroller, the CPU serves as the core component responsible for executing instructions and controlling operations of all peripherals.

b. **Memory:** Microcontrollers contain two types of memory. Volatile memory, also called random-access memory (RAM), stores temporary data that may be lost if the system loses power. Nonvolatile flash memory (such as EPROMM or flash memory) stores the programming instruction set (firmware) of the microcontroller.
c. **Peripherals:** Depending on the intended application, a microcontroller might contain various auxiliary components. I/O interfaces include timers, counters, A/D and D/A signal converters, and communication protocols (UART, SPI, I2C). Auxiliary peripherals might also include LCD screens, Ethernet connectivity ports, and interfaces for such types of modules.

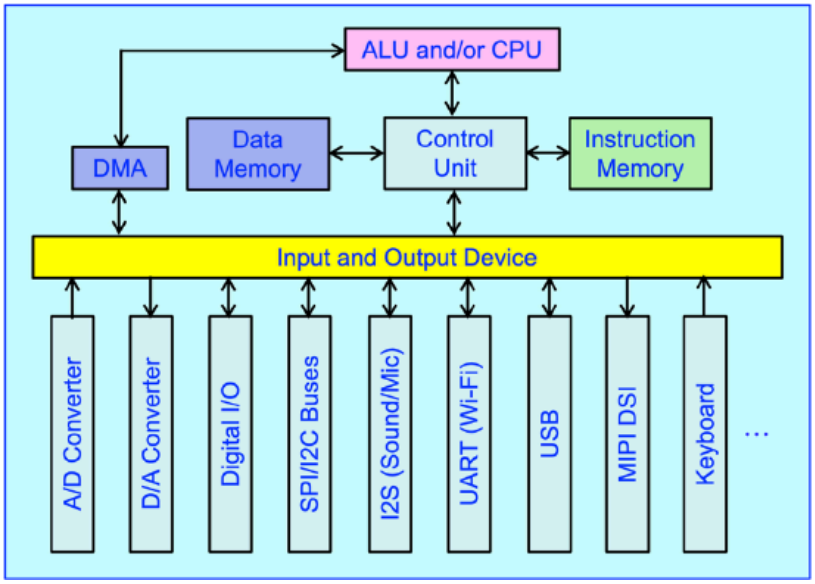


Figure 7.8: Typical microcontroller interface architecture.

Figure 7.8 shows a detailed interface architecture of a microcontroller. At the microprocessor level, a DMA (direct memory access) block at the upper-left corner was added to provide a direct path for the main microprocessor unit (CPU) to access the data written into the DMA. Certain sensors, such as an accelerometer and rate gyro sensor MPU 6050 (see Section 6.6) can be connected to the microcontroller through the SPI (serial peripheral interface) or I2C (inter-integrated circuit) serial communication link and written into the DMA. For these input and output devices connected to the CPU through its control unit, the first group includes A/D and D/A converters. These are mainly used for analog sensor inputs (e.g., pressure sensors) and analog outputs (e.g., analog armature voltages for controlling DC-motors).

The second group has the digital I/O for handing the digital sensor signals (e.g., hall-effect position pulse trains) and output control signals (e.g., the PWM signal for DC-motor control; see Section 5.3). The third group includes the serial communication buses (SPI and I2S) used to handle sensors and actuators equipped with such capability. A microcontroller that needs to input sound signal and output graphics often uses the I2S serial communication for this purpose.

The fourth I/O group includes the UART and the USB. The UART (universal asynchronous receiver/transmitter) serial communication link is used mainly for microcontroller Wi-Fi communication. The USB (universal serial bus) is used for accessing external memory devices and meeting other I/O requirements. Note that the sample microcontroller (Arduino MKR 1000 Wi-Fi) introduced in the next section utilizes Wi-Fi to communicate with the host computer. The last I/O group is for user interfaces. For example, the keyboard is a typical input interface, and output interface includes monitor display through MIPI (mobile industry processor interface) and DSI (display serial interface).

The rest of this section presents a process for selecting a proper microcontroller when designing a specific mechatronic system. This is a critical task, requiring a systematic approach. It is important not to select your microcontroller prematurely. At the same time, it is essential to have a comprehensive understanding of your mechatronic system requirements and architecture before selecting the microcontroller.

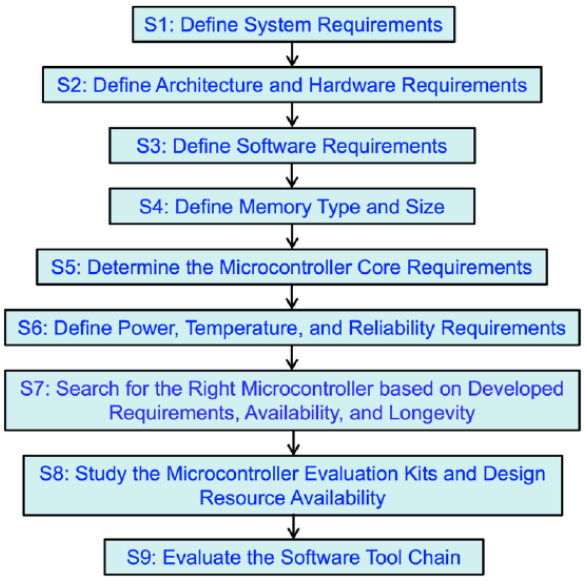


Figure 7.9: Block diagram of selecting a microcontroller for a specific application.

Figure 7.9 presents a nine-step process to guide your microcontroller selection. These nine steps will help you make an informed decision, meeting provided requirements and constraints of the target mechatronic system. Let's look at these steps in greater detail.

- **Step 1) Define system requirements.** To start, we must clearly define the high-level requirements and specifications of the target mechatronic system. We can then understand the requirements on the microcontroller. Four aspects must be considered and accomplished. First, determine the main functions and capabilities of the target mechatronic system by identifying the core tasks and operations handled by the microcontroller. Next, define the performance expectations for the target system including processing speed (e.g., required sample rate), response time, and overall efficiency. Consider the level of performance required to meet the demands of the target system to ensure that the selected microcontroller can deliver the desired performance. Then identify the types of required input and output interfaces by considering various devices and peripherals to be connected to the microcontroller. Determine the necessary interface protocols and connections. Last, define the power supply or battery voltage range and power consumption limits. Select the microcontroller power source, whether battery-powered or connecting to a power outlet.
- **Step 2) Define system architecture and hardware requirements.** Develop a detailed block diagram of the target mechatronic system (e.g., Figure 7.8). Analyze this diagram with various interfaces and the system's I/O requirements. Doing so should help you find the interface requirements for A/D and D/A converters, digital I/O channels, I2C, Ethernet, SPI, UART, and USB. These interfaces will enable communication between the microcontroller and any external devices (sensors and actuators), networks, or peripherals. Determine if the target system requires wireless communication capabilities such as RF (radio-frequency) interfaces; these would include Wi-Fi, Bluetooth, or other wireless protocols. Be sure to show these requirements in the block diagram. For example, the mini-Segway mechatronic system discussed in Chapter 8 require a Wi-Fi connection. Then estimate the number of pins required for the microcontroller. Consider the identified interfaces and I/O components. Each of them typically requires a certain number of pins for connectivity and data transfer. In addition, an additional number of I/O pins should be reserved for future expansion.
- **Step 3) Define software requirements.** Consider your processing requirements such as word-length and speed. What are your requirements for the operating system, data bus width, speed, and software code size estimation? Carefully evaluate software architecture, processing power, and performance requirements. Determine how much data can be transferred between the microprocessor and memory or other components at once. Select the microprocessor size, such as 8-bit, 16-bit, or 32-bit data bus width. Study the algorithm complexity, computational tasks, and real-time processing requirements. What are your specific performance expectations such as response time or overall system efficiency? Determine the processing power needed to handle the workload efficiently. Generate requirements of the real-time operating system, including specific microcontrollers. In addition, determine if the target mechatronic system design requires supporting multiple operating systems. (A powerful microprocessor and additional memory may be needed).
- **Step 4) Define memory type and size.** Carefully consider the memory requirements for software and data when selecting a microprocessor system. These include determining the size and type of memory needed to store data and run your software code. Consider three key factors. First, select a microcontroller configuration with enough memory capacity to handle these mentioned requirements effectively. Factor in both the size of boot or code memory (e.g., flash memory) and the dynamic

memory (RAM) requirements. Second, consider the type of memory interface required by the mechatronic system, including flash memory and DMA (direct memory access). Ensure that the selected microcontroller supports any desired interfaces, such as serial memory. Third, in order to optimize the mechatronic system's performance, utilize a small yet high-speed RAM integrated directly into the microprocessor. Evaluate the system's specific software requirements along with its data access patterns. These will help determine the appropriate cache memory size, which typically ranges from a few KB to several MB.

- **Step 5) Determine the microcontroller core requirements.** Carefully assess the requirements from Steps 1 through 4. Select a microcontroller core (central processing unit) by determining the number of cores with or without FPGA. Identify the appropriate memory interface and memory sizes, and then select the suitable clock speed for the target microprocessor to ensure the optimal performance and functionality for the mechatronic system. Next, identify the appropriate type of microprocessor core, such as ARM Cortex-M, Intel x86, RISC-V, or Cortex-M0+, based on your application's specific needs. Evaluate if an FPGA processor is needed. Determine the number of cores (single- or multi-core) for the selected microprocessor, depending on the level of parallel processing or multitasking required. Evaluate the maximum memory size that the target microprocessor can address. Ensure that sufficient memory can be added to accommodate the software code and data requirements. Determine the optimal clock speed for the target microprocessor to operate, achieving the desired performance level. Meanwhile, consider factors such as power consumption, real-time requirements, and the complexity of computational tasks.
- **Step 6) Define power, temperature, and reliability requirements.** When selecting your microcontroller, it is important to consider power requirements. Factor in processing core voltage, power dissipation, power management, junction temperature, thermal management, and low-power modes. Carefully analyze and compare power consumption vs. performance among the microcontrollers. Evaluate their low-power mode, especially when the mechatronic system operates on battery power. It is crucial to ensure that the selected microcontroller meets requirements under different power modes so that power consumption can be optimized. Consider a thermal management system, including heat sinks and ventilation, to maintain a safe junction temperature. Most microcontroller devices are manufactured in multiple temperature grades to accommodate different application requirements. Therefore, select a proper temperature grade for the microcontroller. The commercial grade is between 0° and 70°C, industrial grade between -40° and 85°C, automotive grade between -40° and 125°C, and military grade between -55° and 125°C. Estimate the microcontroller's expected reliability over a specific period of time. Use the manufacturer datasheet and documentation, including metrics such as mean time between failures (MTBF) and failure in time (FIT) rate, to provide helpful information.
- **Step 7) Search for the right microcontroller based on developed requirements, availability, and longevity.** By this step, all the necessary specifications for choosing the target microcontroller are gathered from Steps 1 through 6. However, there are still a few important points to keep in mind. Explore suppliers such as Arrow, Avnet, Digkey, Mouser, Future Electronics, or similar companies to find pricing and availability information. Talk to your supplier's FAE (field application engineer). Discuss your application and specific requirements. Visit the websites of microcontroller manufacturers, such as Intel, AMD, Renesas, Micron, NXP, Texas Instruments, Microchip, and Arduino. Confirm the data and information you have obtained from suppliers or other sources. Make sure that the selected microcontroller is readily available and can be easily procured. Evaluate the microcontroller's longevity support provided by the manufacturer (important for developing a mechatronic system). Some manufacturers release their products with long-term support and availability, ensuring that the chosen part will be supported and available for an extended period of time.
- **Step 8) Study the microcontroller evaluation kits and design resource availability.** Once the options for microcontrollers are narrowed down, assess them based on the availability of evaluation kits, design resources, and support from the chip manufacturers. Compare the available design resources, including physical symbols, logical symbols, schematics and layout files, application notes, and white papers. Make an informed decision, testing and validating the microcontrollers' performance and features in a real-world setting. Select a microcontroller that not only meets your system's technical requirements, but also offers the resources needed to streamline the design process and ensure successful integration into the target system. For any microcontroller that meets all the required specifications, assess the level of design and technical support that is available. Avoid any microcontroller that lacks sufficient design resources and technical assistance.
- **Step 9) Evaluate the software tool chain.** It is important to consider the availability and compatibility of software compilers and tools, as well as the presence of debugging and development resources. Collectively, they ensure a smooth and efficient development process for the chosen microcontroller. As you move forward, evaluate the availability and compatibility of compilers and tools that are specifically designed for the chosen microcontroller. Consider the range of compiler options and their compatibility. Ensure that the microcontroller supports a variety of debugging tools, such as debuggers, emulators, integrated development environments (IDEs), MATLAB, and Simulink tools that are essential for diagnosing and resolving issues during the development and testing phases. In addition, verify that all the necessary tools required for development—such as programming tools, software libraries, and development kits—are readily available for the chosen microcontroller. These ready-to-use resources simplify programming, testing, and diagnosing tasks.

This nine-step microcontroller selection process can be used as a first round for selecting a few candidate microcontrollers. Then,

a few iterations may be needed to finalize the selection process.

Section Summary

This section discusses microcontroller architecture [5]. First, it addresses the difference between a microcontroller and microprocessor. A microcontroller unit (MCU) [5] is essentially a small computer on a single chip. It is designed to manage specific tasks within an embedded system without requiring a complex operating system. A microprocessor unit (MPU) [6] is a computer processor with the data processing logic and control included in a single integrated circuit (IC) or a small number of ICs. An MPU contains the arithmetic, logic, and control circuitry required to perform the functions of a central processing unit (CPU). Next, the section addresses the general architecture of a mechatronic microcontroller. These are the core of mechatronic systems. They can be categorized based on their architecture, data bus width, instruction set, and application scenarios. Last, the section presents the process for selecting a proper microcontroller to design a mechatronic system. Making this selection requires a systematic approach. It is essential to establish a comprehensive understanding of the system's requirements and architecture before proceeding with the selection of a microcontroller.

Section Keywords

Practice Problems

1. Identify whether each of the following statements is true or false.
    a. Both microprocessor units (MPU) and microcontroller units (MCU) have a central processing unit (CPU) or an arithmetic and logic unit (ALU).
    b. The central processing units in both microprocessors and microcontrollers utilize the same architecture
    c. In addition to any central processing units, a microcontroller also contains integrated memory, input/output (I/O) components, and other various peripherals.
2. Select which answer best completes the following sentence. The central processing unit inside a microcontroller utilizes:
    1. Harvard architecture.
    2. Both Harvard and von Neumann architectures.
    3. von Neumann architecture.
3. For a microcontroller, what are the set of typical I/O channels?
4. List all nine steps in the process of selecting a microcontroller.

This page titled 7.2: Microcontrollers Used for Mechatronic Systems is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi

Learning Objectives

After completing this section, you will be able to:

1. Describe the Arduino MRK 1000 Wi-Fi microcontroller architecture.
2. Explain the Arduino MKR 1000 Wi-Fi microcontroller's external wiring diagram and its I/O channels.
3. Discuss the available software for programming the Arduino MKR 1000 Wi-Fi.

This section provides detailed information of a sample microcontroller, the Arduino MKR 1000 Wi-Fi [7], used in the mini-Segway and three-wheel cart mechatronic systems described in Chapters 8 and 9. Arduino (https://www.arduino.cc/) has released over 100 hardware products: boards, shields, carriers, kits, and other accessories. Figure 7.10 shows a few Arduino microcontrollers from entry to enhanced levels. Note that Arduino microcontrollers have open-source electronics platform with easy-to-use hardware and software. Much of the hardware is available, including DC-motor drive boards, accelerometers, and rate gyro sensors. Regarding the software used for programming and implementation, Arduino microcontrollers are able to work with MATLAB and Simulink in addition to Arduino's IDE software. Thus, Arduino's microcontrollers have a wide scope of applications, such as IoT (Internet of things), smart home, and DIYs (do-it-yourself).

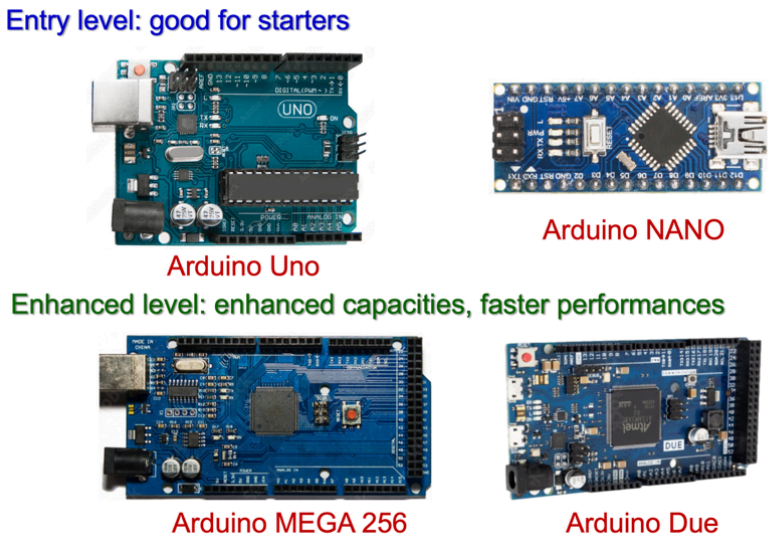


Figure 7.10: Sample Arduino microcontrollers.

Let's first provide details about programming Arduino microcontrollers. Arduino's IDE software can be used to program based on associated hardware specifications. It generates user-specified control sequences with fast compiling speed and computational codes. The detailed application information regarding this software can be found in references [20] for Arduino applications and [21] for Arduino's application hub. Meanwhile, MATLAB and Simulink are graphic programming tools used to generate the target code with the help of MATLAB's real-time code generation. The code is then compiled into the Arduino microcontroller. Using MATLAB and Simulink to program Arduino microcontrollers has several advantages. They are more interactive by visualizing programming code in the form of block diagrams. They generate real-time visible control sequences and responses. They provide physics-based programming such as implementation of low- and high-pass filter transfer functions. They are also compatible with control design, signal processing, and offline simulations (such as software-in-the-loop (SiL) simulations).

Using MATLAB and Simulink support packages for Arduino hardware, we can interactively communicate with an Arduino microcontroller board. These add-on packages enable us to acquire analog and digital sensor data from an Arduino board, and control other devices (actuators) with digital and PWM outputs. We can drive DC, servo, and stepper motors (also support the Adafruit motor shield). We can access peripheral devices and sensors connected over I2C or SPI and also communicate with an Arduino microcontroller either over a USB (or micro-USB) cable or wirelessly over Wi-Fi in real-time. We can also build custom

add-ons to interface with additional hardware and software libraries, and calibrate the control parameters in real time.

Now let's dive into a specific Arduino microcontroller, the Arduino MKR 1000 Wi-Fi [7]. Figure 7.11 shows a picture of the microcontroller board. The Arduino MKR 1000 Wi-Fi is featured with a low power MCU with 32-bit computational capacity, low-power 2.4G Hz IEEE 802.11 b/g/n Wi-Fi connection, and both Simulink and IDE for software development. Here, we will focus on the Simulink-based programming software.



Figure 7.11: Arduino MKR 100 Wi-Fi microcontroller board.

The Arduino MKR1000 Wi-Fi is a microcontroller board with Wi-Fi connectivity, making it possible to connect the host computer to the microcontroller wirelessly. It is based on the Atmel ATSAMW25 SoC (system on chip), which is part of the SmartConnect family of Atmel Wireless devices. These were specifically designed for (IoT) projects and devices. The ATSAMW25 is composed of three main blocks: a) SAMD21 Cortex-M0+ 32bit low-power ARM MCU, b) WINC1500 low-power 2.4GHz IEEE 802.11 b/g/n Wi-Fi (supporting SHA-256), and c) ECC508 Crypto-Authentication for security.

The board design includes a LiPo charging circuit. This circuit board allows the microcontroller to run on battery power or external 5V. It charges the LiPo battery while running on external power. Switching from one power source to the other is automatic. The circuit board has a good 32-bit computational power similar to the Arduino Due board (see Figure 7.10), the usual rich set of I/O interfaces, and low-power Wi-Fi with a Crypto-chip for secure communication. It easily uses Arduino software (IDE) and Simulink for code development and programming.

These features make this microcontroller board the preferred choice for the emerging IoT battery-powered mechatronic projects in a compact form. The micro-USB port can be used to supply power (5V). The Arduino MKR 1000 Wi-Fi is able to run with or without the LiPo battery and has limited power consumption.

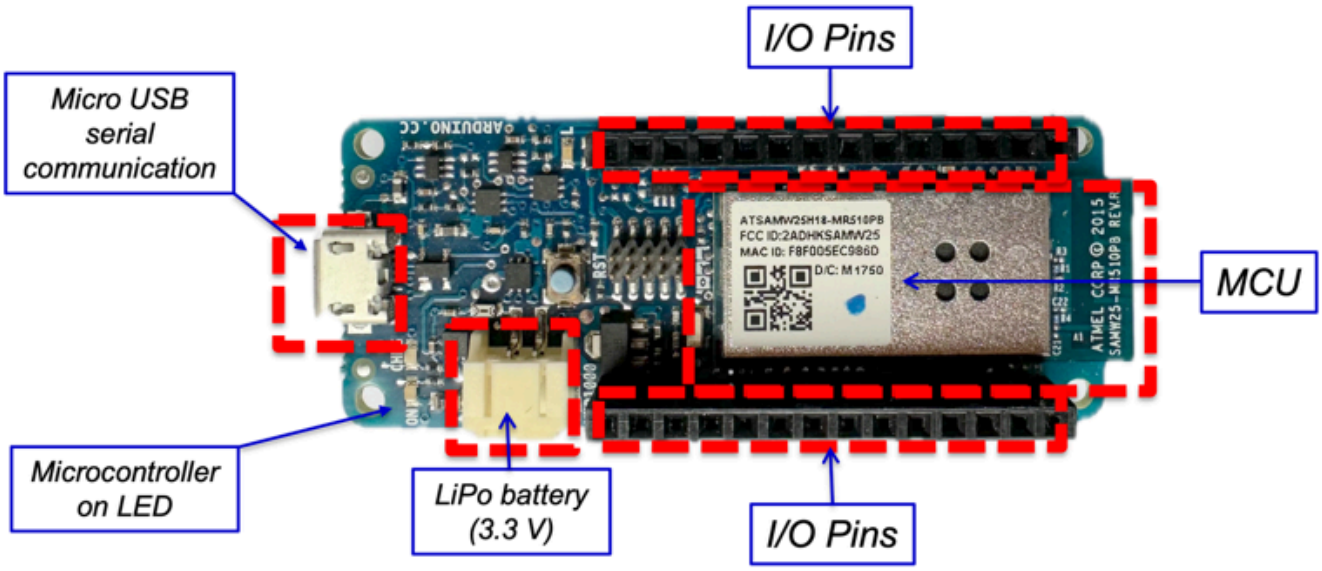


Figure 7.12: Arduino MKR 1000 Wi-Fi board layout (side A).

Figure 7.12 shows side A of the Arduino MKR 1000 Wi-Fi board. The I/O pins are located on the upper and lower sides. The micro-USB on the left provides power to the microcontroller and serial communication. The microcontroller unit (MCU) is on the right with integrated Wi-Fi antenna. The LiPo battery port is located next to the left of the I/O lower pins. There is also an LED power display located at the lower left corner.

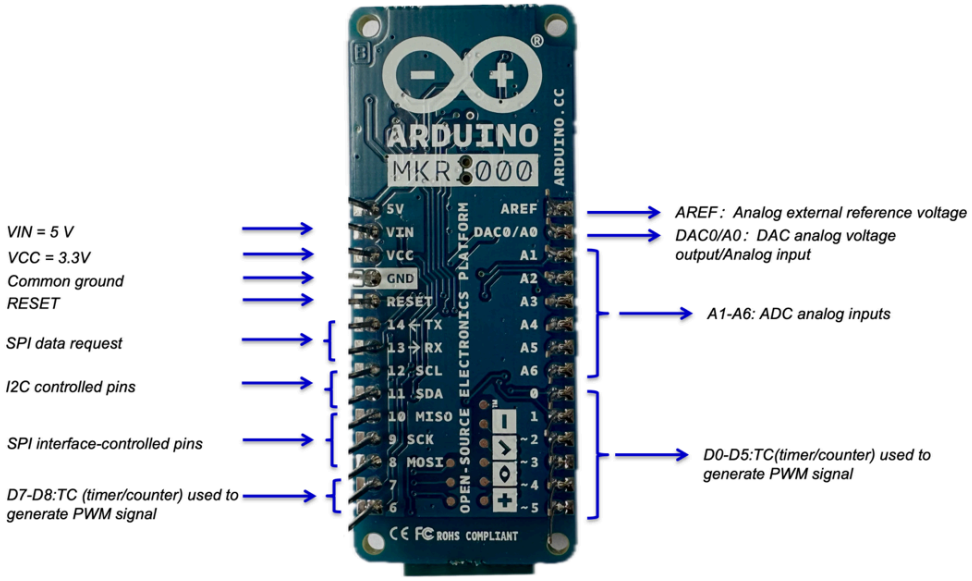


Figure 7.13: Arduino MKR 1000 Wi-Fi board layout (side B).

Figure 7.13 shows the output pins of the Arduino MKR 1000 Wi-Fi board. On the left side, from top down, are the power and ground pins; the SPI and I2C serial communication pins; and two channels of digital input or output pins used to count pulses or to generate pulse trains such as PWM (pulse width modulation) signals. On the right side, from top down, are the analog reference voltage pins; one D/A converter output (analog) or one A/D converter input pin; six channels of A/D converter inputs; and six additional channels of digital input or output pins. These pins of the Arduino MKR 1000 Wi-Fi board contain only a subset of those from the SAMD21 Cortex-M0+ 32bit low power ARM MCU. They reduce the number of output pins for the Arduino MKR 1000 Wi-Fi board and meet the requirements for most of the target mechatronic applications.

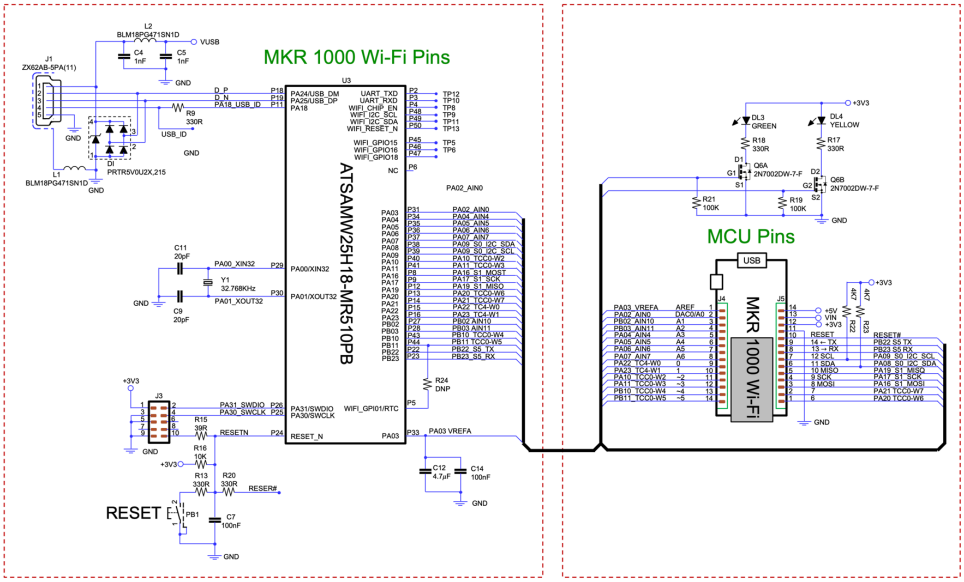


Figure 7.14: Arduino MKR 1000 Wi-Fi circuit and pin map between the extension board and the ARM microprocessor unit (MPU).

To understand the details of microcontroller boards, we must understand its electronic circuit. Figure 7.14 details the circuit of the MKR 1000 Wi-Fi board. The left side of circuit is for the MKR 100 Wi-Fi board. The right side is for connections of SAMD21 Cortex-M0+ 32bit low power ARM MCU. On the upper-left corner is the micro-USB connector (J1). It has five pins and a USB shield grounded through an inductor. The Zener diode and four regular diodes are used to protect the board for those inputs outside the signal range. The circuit on the middle-left is used to generate clock pulses (32.768 kHz) for the SAMD21 Cortex-M0+ 32bit low power ARM MCU. It utilizes one quartz (Y1) and two capacitors. The LiPo battery power input connecter (J3) is on the lower-left corner. If it is connected, the microcontroller will be powered by the battery. The other circuit on the lower-left corner is for the MCD (microcontroller device) reset used to reset the MCU if needed. On the top-right are connections for the UART Wi-Fi communication to the ARM MCU. On the middle-right are those output pins of the MKR 1000 Wi-Fi board connected to the ARM MCU. The last pin on the right is the reference voltage output from the ARM MCU. For the right MCU pins in Figure 7.14, there are two digital LED display circuits on the top. One indicates the ARM MCU's normal operation; the other is for customized display. Table 7.1 summarizes these specifications of the MKR 1000 Wi-Fi board.

Table 7.1: Specifications of the Arduino MKR 1000 Wi-Fi board.

Microcontroller	SAMD21 Cortex-M0+ 32bit low power ARM MCU
Board Power Supply (VIN)	5 V
Circuit Operating Voltage	3.3 V
Digital I/O Pins	8
Analog Input Pins	7 (ADC resolution 8/10/12 bit); max 350,000 samples per second
Analog Output Pins	1 (DAC resolution 10 bit); max 350,000 samples per second
Flash memory	256 KB
SRAM	32 KB
Clock Speed	32.768 kHz (RTC), 48 MHz
Length	61.5 mm
Width	25 mm
Weight	32 grams

Example 7.2

This example dives into the LED display DL3 (see the red boxed LED in the photo within Figure 7.15). It uses the Simulink block diagram to show how to program the circuit that will light the display.

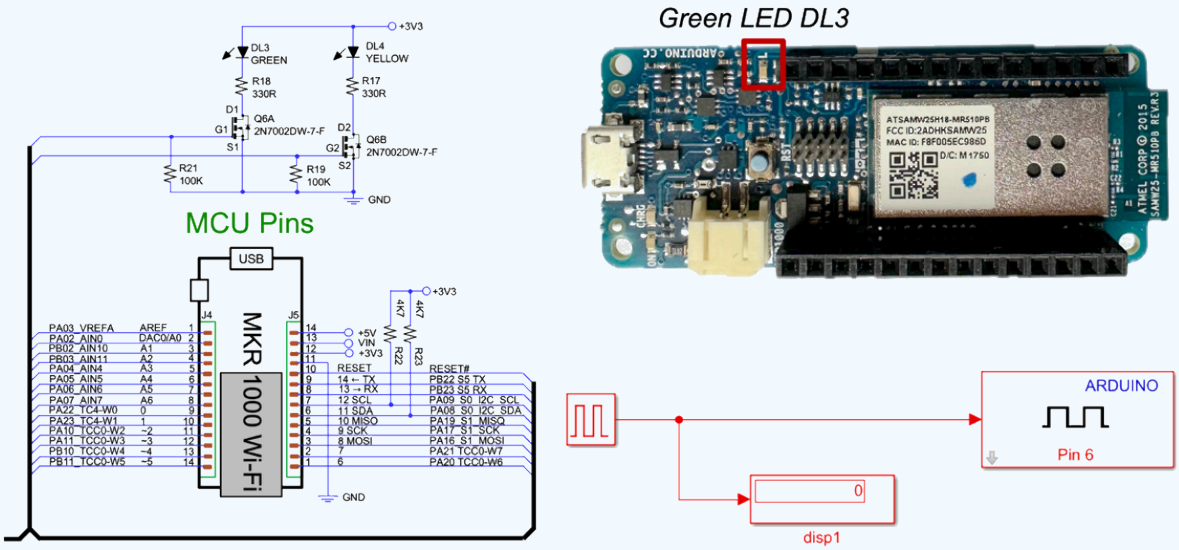


Figure 7.15: LED (DL3) control for the Arduino MKR 1000 Wi-Fi board.

The LED display DL3 is connected to the digital timer/counter PA20, labeled as TCC0-W6. The microcontroller technical document for TCC0-W6 [10] indicates that TC stands for timer/counter, which can be set to count clock pulses or generate waveform signals such as pulse-width modulation (PWM) signals. The timer/counter controls DC motors and other devices including the LED display. It can also be used as a direct digital output.

To control the LED DL3, the MATLAB and Simulink Arduino support package needs to be installed; see Section B.0 of Appendix B. After this package is installed, select the Arduino digital output block, shown in the lower-right part of Figure 7.15. (The Simulink support package will be in the Arduino hardware library.) Configure the digital output channel to Pin 6 (associated with TCC0-W6).

The circuit on the left of Figure 7.15 indicates that when the digital output of channel Pin 6 is high, transistor Q6A will be turned on and LED DL3 will be lit. When the output of Pin 6 is low, the transistor Q6A will be turned off; as a result, LED DL3 will be off. A square pulse wave must be generated to control the digital output (Pin 6) between high and low states. Simulink's "Pulse Generation" block can be utilized for that purpose. We can also specify the magnitude of pulse to one, period to 2 second, and duty-cycle to 50%. In this case, the LED DL3 will be on (lit) for one second and off for one second. If the period is reduced to 0.02 second with 50% duty-cycle, the human eye will not be able to distinguish the on and off events. Instead, the LED DL3 will be lit with a dimmer of 50%. One may increase or decrease the duty-cycle to adjust the dimmer level.

Remark 7.5

Example 7.2 shows the simplicity of using Simulink to program the Arduino MKR 1000 Wi-Fi.

Figure 7.16 shows the block diagram of the SAMD21 Arm Cortex-M0+ microcontroller unit (MCU) from Microchip Technology. It is used in the Arduino MKR 1000 Wi-Fi. This microcontroller is a low-power MCU with up to 256KB flash and 32KB of SRAM memory. Based upon its peripheral capability, the microcontroller chip can be packaged from 32- and 48- to 64-pins. The Microchip Technology's SAM D21 MCUs have a maximum clock frequency of 48MHz. The SAMD21 Arm Cortex-M0+ Microcontroller is designed for easy migration with identical peripheral modules. Note that not all the microcontroller input and output pins are available on the MKR 1000 Wi-Fi board.

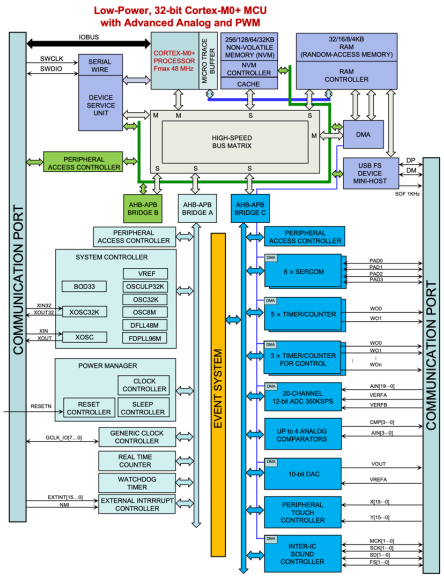


Figure 7.16: Block diagram of the SAMD21 Cortex-M0+ 32bit low power ARM MCU.

In addition to the characteristics of the microprocessor unit (MPU), the microcontroller peripherals' capability is also extremely important for mechatronic systems. Let's consider four groups of peripherals for the SAMD21 Arm Cortex-M0+ MCU. The first is the capability of the direct memory access (DMA). This microcontroller has a 12-channel direct memory access controller (DMAC); it is important for quickly accessing certain data such as sensor data provided through the serial communication.

The second group includes the digital input and output (I/O) channels. The SAMD21 Arm Cortex-M0+ MCU is equipped with a 12-channel event system. There are up to five 16-bit timer/counters (TCs). They can be reconfigured as either 8- or 16-bit TCs with two compare/capture channels or as 32-bit TCs with two compare/capture channels by using two TCs. These counters are extremely important for a mechatronic system. They can be used to generate PWM signals for DC-motor control or to count pulse train signals of hall-effect sensors for calculating the motor shaft speed (see Section 6.2). In addition, there are watchdog timer (WDT), 32-bit real time counters (RTC) with clock/calendar function, and three 24-bit timer/counters for control (TCC) with extended functions.

The third group provides the serial communication capability. The SAMD21 Arm Cortex-M0+ MCU has up to six serial communication interfaces (SERCOM). Each is configurable to operate as either SPI (serial peripheral interface), I2C (inter-integrated circuit) up to 3.4 MHz, LIN (local interconnect network), or USART with full-duplex and single-wire half-duplex configuration. These flexible serial communication capabilities are very important for those sensors and actuators utilizing serial communication for data exchange. For example, the MPU 6050 acceleration and rate gyro MEMS is able to use SPI and I2C serial communications to send acceleration and angular rate signals to the MPU. The monitor and keyboard utilize the USART channels. There are also one full-speed (12 Mbps) USB 2.0 interface, two inter-integrated sound (I2S) channels, and a cyclic redundancy check (CRC)-32 generator. For the mechatronic applications described in Chapters 8 and 9, the I2C serial communication is used to send the PWM command of armature voltage to the MKR motor carrier board and to read the motor speed of the hall-effect sensor from the same board.

The last group is for analog input and output channels. The SAMD21 Arm Cortex-M0+ MCU has one 12-bit, 350 kHz sample rate, A/D converter. It can be configured for up to 20 A/D channels using an equipped analog switch; one 10-bit, 350 kHz sample rate D/A converter that cannot be reconfigured to multiple channels; up to four analog comparators (AC), and a peripheral touch controller (PTC) with 256-channel capacitive touch and proximity sensing. Note that a single channel of D/A conversion should not be an issue. Most of the actuators are controlled by the PWM signals through digital outputs.

There is a relatively new Arduino microcontroller board, the UNO Q. It is a single-board computer combining the Qualcomm Dragonwing QRB2210 microprocessor; a quad-core Arm Cortex-A53 running Debian Linux OS, with the STMicroelectronics STM32U585 microcontroller; and an Arm Cortex-M33 running Arduino Core on Zephyr OS. The Linux system and the microcontroller communicate through Bridge, Arduino's remote procedure call library. This allows Arduino to access Linux services for high-level tasks. The UNO Q also features dual-band Wi-Fi 5 and Bluetooth 5.1 for wireless connectivity; a USB-C connector with power delivery input and video output; and Arduino-compatible headers for easy expansion with its shields, carriers, and accessories. Details of the Arduino UNO Q can be found in its data sheet [11].

Section Summary

This section provides detailed information of a sample microcontroller, the Arduino MKR 1000 Wi-Fi [7]. First, it discusses the Arduino family of microcontrollers, along with its IDE (integrated development environment) software [8] and programs under MATLAB and Simulink [9]. Next, we introduce the Arduino MKR 1000 Wi-Fi microcontroller in detail, including its extension board and pin layout, circuit and pin map, surrounding electronic components, and its core, the ARM 32-bit Cortex-M0+ MCU [10]. The intention is to familiarize readers with the microcontroller and its extension circuits from the hardware point of view. You can then compare the sample microcontroller (Arduino MKR 1000 Wi-Fi) with the microcontroller architecture discussed in Section 7.2. Last, we briefly discuss the MATLAB and Simulink Arduino MKR 1000 Wi-Fi software with an example. Appendix B provides more detailed discussion.

Section Keywords

Arduino IDE software: Arduino IDE (integrated development environment) is an open-source software application designed for writing, compiling, and uploading code to Arduino microcontroller boards. It serves as the primary interface for users to interact with Arduino hardware.

Arduino microcontroller: This microcontroller provides accessibility to beginners for building interactive electronic projects, allowing them to read inputs from sensors and control outputs like LEDs, motors, and other devices family.

Arduino MKR 1000 Wi-Fi: This microcontroller board is designed for internet of things (IoT) projects, featuring a built-in WiFi module for internet connectivity, an ARM Cortex-M0+ processor, a cryptographic chip for secure communications, and an onboard battery charger for easy portability.

Practice Problems

1. What microcontroller unit is used in the Arduino MKR 1000 Wi-Fi board? Select the correct answer.
    1. Atmel's SAMD21 MCU
    2. SAMD21 Cortex-M0+ 32bit low power ARM MCU
    3. ATmega328P microcontroller (MCU)

1. How many D/A channels are available for the Arduino MKR 100 Wi-Fi board? Select the correct answer.
    1. One
    2. Two
    3. Three

1. How can the Arduino MKR 1000 Wi-Fi board be powered? Select the correct answer.
    1. 5V power from mini-USB only
    2. 3.3V power supply from J3 LiPo battery connector only
    3. Either 5V from mini-USB or 3.3V from J3 LiPo battery connector

1. Develop a Simulink diagram to manually turn the LED DL3 on and off, using the example presented in Figure 7.15.

This page titled 7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

7.4: Mechatronic Control System Architecture- Speed Control Example

Learning Objectives

After completing this section, you will be able to:

1. Discuss the relationship between mechatronic system hardware and its architecture.
2. Explain the Arduino MKR motor carrier board functionality and know how to use it.
3. Describe the Simulink software for DC-motor speed control for the Arduino MKR 1000 Wi-Fi.

Section 7.1 introduced the overall hardware and software architectures of mechatronic control systems. In this section, we will use an example—mechatronic DC-motor speed control system—to relate each component to the physical domain, interface, and information domain in a mechatronic system. In addition, control system analysis and design block diagrams, as shown in Figure 7.2, present a simplified system architecture. Because only mathematical relationships are considered, they are good for control system performance analysis. Using the DC-motor speed control mechatronic system as an example, this section demonstrates how to convert a mechatronic system architecture to a control system block diagram.

Next, let's look at details of the Arduino MKR motor carrier board. It will be used in the DC-motor speed control system and other mechatronic system hardware as presented earlier. The MKR motor carrier board is an open-source add-on board designed to control servo, DC, and stepper motors; see Figure 7.17 for its picture and pin definitions. The board's circuit and Eagle PCB (printed circuit board) files are provided in references [23] and [24], respectively.

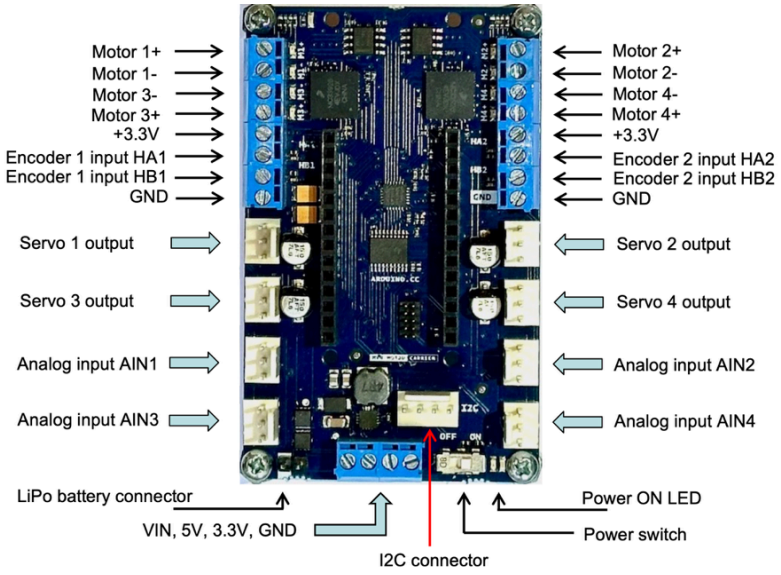


Figure 7.17: Arduino MKR motor carrier board and its pin definitions.

The Arduino MKR motor carrier board is compatible with all the boards in the MKR family. It has four servo motor outputs, four DC-motor outputs (two high performance and two standard performance), current feedback signals for the high-performance motors, two encoder sensors for pulses A and B, four analog inputs for sensors, an on-off switch with LED, a LiPo battery connector, and an I2C connector. Figure 7.17 shows the associated pins for the inputs and outputs mentioned. The board also features two H-bridge MC33926 motor drivers for high-performance DC-motor control with direct connection to the Arduino MKR 1000 Wi-Fi board through the two connectors in the center of the board, as well as current feedback, and capacity for up to 5 Amps (peak) motor drive current. In addition, there are two DRV8871 drivers that are controlled by the plug-in Arduino MKR 1000 Wi-Fi via I2C. The Arduino MKR 1000 Wi-Fi can be used to control the servo and DC motors, read the encoder signals, and read the battery voltage. For extended use or high-current motors, an extra heat sink might be required for the drivers.

When plugging the Arduino MKR 1000 Wi-Fi into the MKR motor carrier board, some of the Arduino MKR 1000 pins will no longer be available; they are used to control features of the carrier board. For example, the current feedback from the two MC33926 drivers is connected directly to some of the analog input pins on the MKR 1000 Wi-Fi board. To use the carrier board, we must plug an MKR board (the Arduino MKR 1000 Wi-Fi, MKR Zero, or other) onto the headers at the center of the Wi-Fi board, making sure it is connected in the proper direction. Check that the information printed on the side of the headers matches for both the Wi-Fi board and the carrier board.

Now, we can discuss the interface hardware for a DC-motor speed control mechatronic system. Figure 7.18 shows the hardware located in the information and physical domains, based on the mechatronic system architecture presented in Figure 7.1. The Arduino MKR 1000 Wi-Fi board is located in the information domain. Its 3.3V power is supplied by the Arduino MKR motor carrier board. This connection is completed by inserting the Wi-Fi board onto the carrier board.

In the information domain, a host computer with Simulink GUI is connected to the Arduino MKR 1000 Wi-Fi board through the local Wi-Fi network. The GUI can be calibrated through control buttons in real time. Both the host computer and the Wi-Fi board are connected. In addition, the host computer can also be connected to the Wi-Fi board through the mini-USB connection (not shown). This connection is used mainly to load the Simulink software to the Wi-Fi board.

To control the DC motor in a closed-loop, the Wi-Fi board sends the desired PWM duty and sign signals to the carrier board through the I2C communication (orange lines). The I2C communication is connected to the carrier board after the Wi-Fi board is inserted onto the carrier board. The motor speed encoder counts are also sent by the carrier board to the Wi-Fi board via an I2C communication protocol (orange lines).

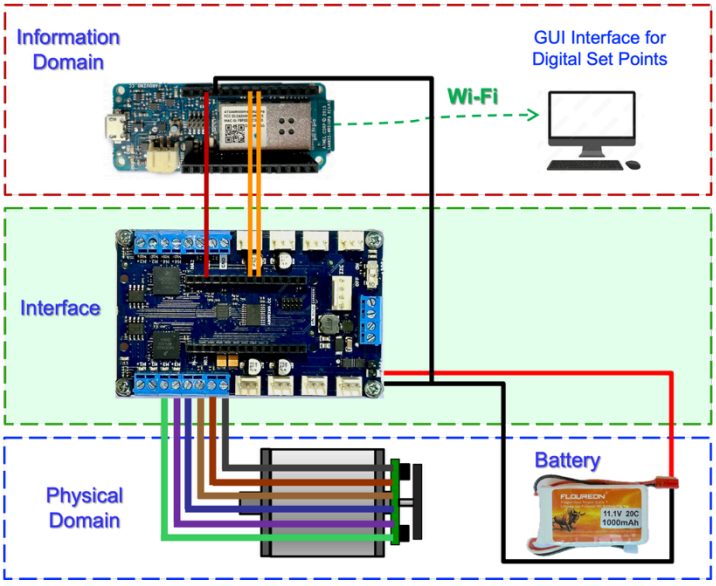


Figure 7.18: Arduino MKR hardware architecture for a DC-motor speed control system.

The interface hardware is formed by the carrier board. The dedicated microcontroller on the carrier board reads the two-encoder pulse trains. It then transmits them to the Wi-Fi board through the I2C pins (SDA and SCL) on the designated pins. This allows the Wi-Fi board to access and process the encoder information.

The other task for the carrier board is to use the PWM motor control signals sent via the I2C communication to precisely control both the motor speed and direction. The signals are sent to the H-bridge drive of MC33926. The carrier board also converts the battery voltage to 3.3V power supply for the Wi-Fi board.

The physical domain has two main components: the battery and a DC motor with encoder. The battery supplies power to both the Wi-Fi and carrier boards. The DC motor with encoder has six pins. Two pins are associated positive and negative sides of the DC-motor armature winding. The other four pins help measure the motor shaft speed and direction. Within these four pins, two are for ground and 3.3V power supply. The other two are for the encoder pulse trains A and B (see Section 6.2). The Arduino MKR motor carrier board pin definition can be found in Figures 7.17 and 7.18 with detailed signal flow and wire connection definitions.

The control system block diagram essentially presents the mathematical relation of a mechatronic system without considering its power flow. As a result, details such as power amplification, signal transmission through serial communication, and signal

A/D digital and D/A conversion errors due to digitization, are often ignored. However, when analyzing a digital control problem, consider the sample and hold process for the cases using zero-order sample and hold and other sample and hold situations [26], [27].

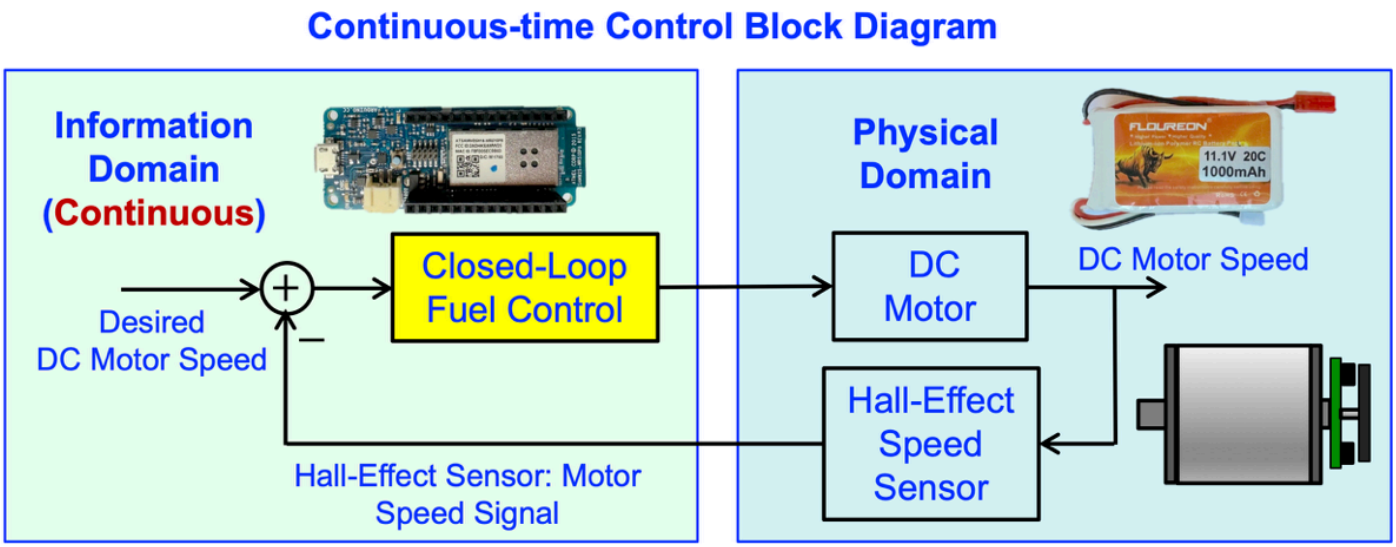


Figure 7.19: Continuous-time system block diagram for continuous-time control design and analysis.

The control system block diagram of mechatronic systems can be divided into two groups. One is a continuous-time system block diagram; see Figure 7.19 for DC-motor speed control system. The other is a hybrid control system block diagram; see Figure 7.20 which combines a continuous-time physical system model with a discrete-time controller in the information domain. The continuous-time block diagram synthesizes the continuous-time controller and then discretizes it for implementation to the microcontroller. The hybrid block diagram enables designing a discrete-time controller directly.

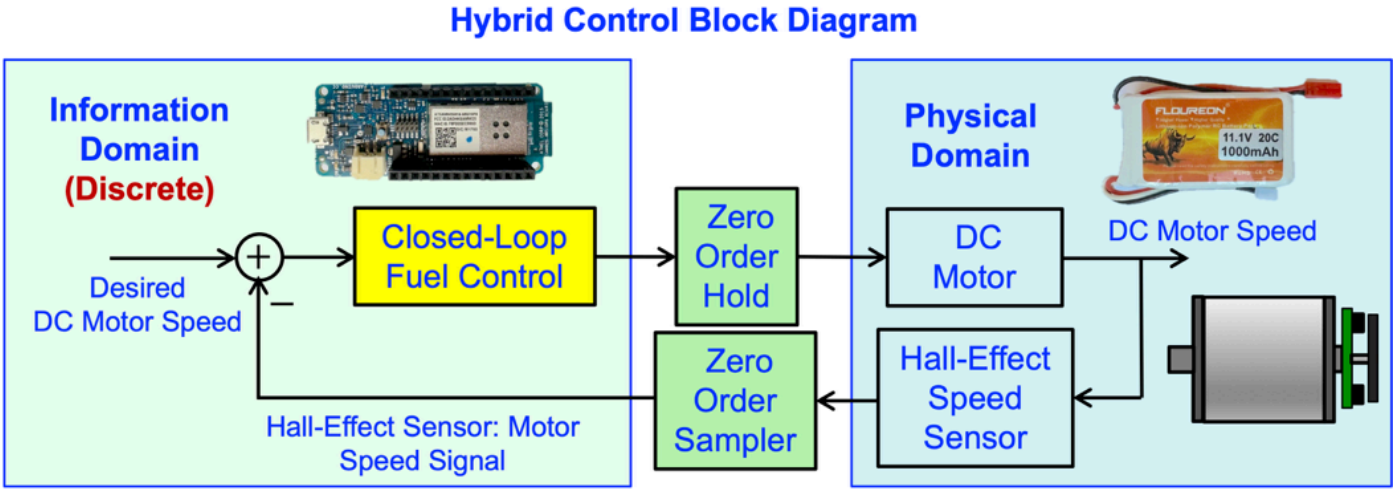


Figure 7.20: Hybrid system block diagram for discrete-time control design and analysis

Control theories are developed for synthesizing both continuous- and discrete-time systems. Most, practical physical systems are in the continuous-time domain. However, the system controllers are implemented into a microcontroller with a CPU or MPU. As a result, discrete-time controllers must be designed for implementation purposes.

There are two approaches for designing or synthesizing a discrete-time controller. The first approach uses the control system block diagram from Figure 7.19. Design the continuous-time controller, then discretize it to obtain a discrete-time controller based on the selected sample and hold such as zero-order sample and hold (see Section 4.2) [13]. The second approach is to design a discrete-time controller based directly on the control system diagram from Figure 7.20 [28]. This approach is preferred because discretizing a continuous-time controller could lead to additional error whereas the designed discrete-time controller can be directly implemented.

The next step is generating software for the mechatronic control system that can be implemented into a microcontroller. As an example, let's look at a DC-motor speed control system to demonstrate how to use Simulink for a program that can be implemented into the Arduino MKR 1000 Wi-Fi microcontroller. In order to control the DC-motor speed, the Wi-Fi board needs to be properly connected to the carrier board. We can start programming the board using Simulink when the mini-UBS wire is connected to the Wi-Fi board. To control the DC motor with speed feedback, we must install the Simulink Arduino MKR library. For MATLAB 2024b, click the drop list of "Add-Ons" and select "Get Hardware Support Packages." In the new window, select "Simulink Support Package for Arduino Hardware." Then install the Simulink library package that includes Arduino MKR 1000 Wi-Fi and its motor carrier board.

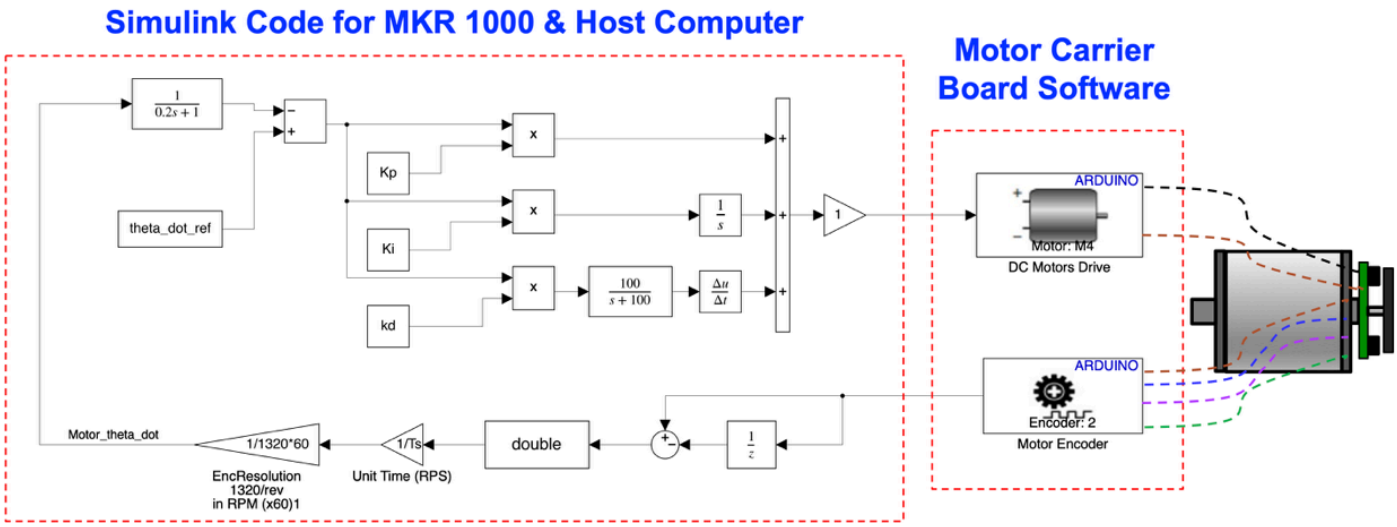


Figure 7.21: Simulink software for closed-loop DC-motor speed control.

When working with any DC motor, we need an external power source to feed the motor driver and power the motor. For the DC-motor speed control system, we can connect a LiPo battery to the MKR motor carrier board (see Figure 7.18). Adding a switch to turn the power on and off is recommended. Once power is connected, we can program in Simulink to regulate the DC-motor speed. Figure 7.21 shows the Simulink program as a block diagram. Now, let's discuss this Simulink program step-by-step from right to left.

Consider the program for the MKR motor carrier board on the right. A DC motor is connected to it. The armature voltage in the form of a PWM signal is used to drive the motor in two directions, and encoder signals (pulses A and B) are collected for motor speed feedback. To make the closed-loop control system work, we must program the carrier board. From the Motor Drive board in the Simulink library, find the Arduino DC Motor Drive block and the Arduino Motor Encoder block. Configure the carrier board to include the wire connections for both the DC motor and encoder.

For this example, the software configuration is based on the wire connection of the DC motor and encoder shown in Figure 7.18. The DC-motor armature winding is connected to the M3 motor pins, and the motor encoder is connected to the encoder #1 pins. As a result, both the DC-motor drive and motor encoder blocks must be calibrated to reflect these wire connections. The left configuration window in Figure 7.22 is for the DC-motor drive selection, where the motor drive port can be selected through the drop-down list. For the DC-motor wiring shown in Figure 7.18, the motor drive was selected to be M3. The right configuration window shown in Figure 7.22 is for the motor encoder selection, where the encoder port can be selected through the drop-down list. For the DC-motor wiring shown in Figure 7.18, the encoder number was selected to be #1.

The encoder output in Figure 7.21 shows the current digital count updated with a given sample period. Here, the controller update period, T_s, is input as the "sample time" shown in the right window of Figure 7.21. The feedback signal sample period matches the controller update period. In addition, the counter output increases as the motor shaft rotates clockwise and decreases as it rotates anticlockwise. The low-level software of the encoder block combines the two pulse trains (A and B) to generate one speed signal with a rational sign.

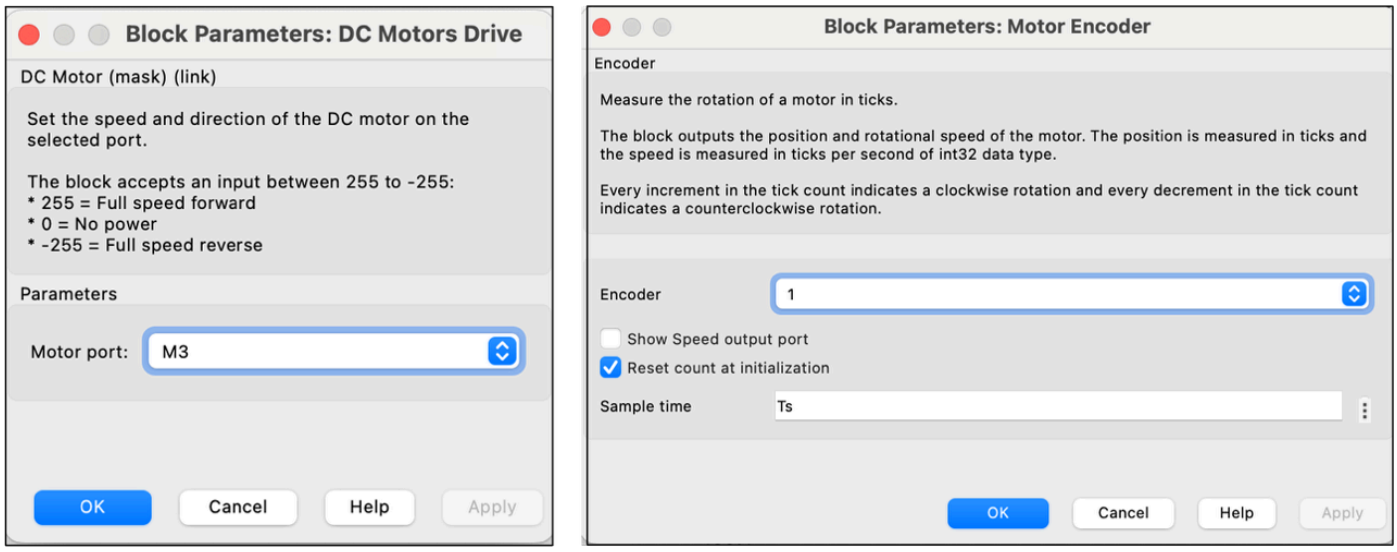


Figure 7.22: Simulink software interfaces for a motor carrier board.

The speed control Simulink software shown in the left Simulink block diagram in Figure 7.21 can be divided into three parts. The first is the DC-motor speed calculation. The counter difference signal is an integer (32-bits) and needs to be converted into double (64-bits floating number) to be used in Simulink. After the conversion, the difference is divided by the sample period, T_s (second), to obtain a signal with a unit of pulse per second. Based upon the discussion in Section 6.2, for the selected motor, each motor output axis rotation leads to 1320 pulses. As a result, we can obtain the signal in pulse per second by dividing the counts by 1320 and then multiplying by 60 to obtain a motor speed signal $\dot{\theta}_m$ in RPM (rotation per minute).

The second part of the block diagram is the reference speed signal $\dot{\theta}_{ref}$. Subtracting the low-pass filtered motor axis speed $\dot{\theta}_m$ from it generates the speed error signal for the PID controller. The low-pass filter (see Section 4.7) has a unit gain and a time constant of 0.2 second. In addition, we can develop certain GUI connections to change the reference signal $\dot{\theta}_{ref}$ (not shown here). This makes modifying the motor reference speed easy. (See the mini-Segway and three-wheel cart software in Chapters 8 and 9 for detail.)

The third part of the diagram is a continuous-time PID controller. Constant blocks (K_p, K_i, and K_d) are PID control gains (proportional, integral, and derivative). They can be tuned to achieve the desired performance of closed-loop controlled motor speed based upon the tuning method described in Section 3.4. The integral used in the Simulink model is continuous-time, and the same for the derivative operation. To improve the efficiency of integration and derivative operations, the discrete-time integral and derivative should be implemented. A low-pass filter is used to smooth the error speed signal in the Simulink model. This reduces the noise generated by the derivative operation (see Section 3.4). The DC-motor drive block uses an integer number (PWM duty-cycle number) to generate the armature PWM signal for motor control. The double format of the PID controller output is automatically converted into the integer for the motor drive.

Understand how the software is executed in these target processors is helpful in the long run. Note that the motor drive and encoder counting is executed in the Arduino MKR motor carrier board. The MKR 100 Wi-Fi microcontroller board is executed through the I2C serial communication. The rest of the software is executed in the host computer with the Simulink software running in real-time. As a result, the speed reference signal $\dot{\theta}_{ref}$ and PID gains (K_p, K_i, and K_d) can be adjusted in real-time by opening the associated Simulink constant block.

Section Summary

This section addresses the control system architecture for mechatronic systems. It uses an example of a DC-motor speed control, calling on the overall architecture for both hardware and software introduced in Section 7.1. The section presents the hardware of the DC-motor speed control system utilizing the Arduino MKR 1000 Wi-Fi microcontroller. To make this control system work, an additional DC-motor drive board, the Arduino MKR motor carrier board [12], is needed to drive the DC motor and collect the hall-effect speed sensor signals. Note that the Arduino MKR motor carrier board is open source. Next, the section introduces the Simulink motor control software. This software shows how the motor drive and hall-effect sensor input blocks in the Arduino Simulink library can be used to develop closed-loop control software utilizing the PID (proportional-integral- derivative) control [13].

Section Keywords

Arduino MKR motor carrier board: This add-on board is designed to control a variety of motors (servo, DC, and stepper) and to connect other sensors. Compatible with the Arduino MKR family, it provides connections for four servo motors, four DC motors (two high-performance and two standard), two encoder sensors, and four analog sensors.

Arduino Simulink library: This selection of Simulink blocks is specifically designed for interacting with Arduino hardware. These blocks enable configuration and access to Arduino sensors, actuators, and communication interfaces (e.g., I2C, SPI, serial).

Control system architecture: This structure comprises the physical hardware components and the software that organizes and controls them. The hardware components include sensors, microcontrollers, and actuators. The software architecture defines the overall structure.

PID DC-motor speed control: This control utilizes a feedback mechanism based on the difference between a desired motor speed and the actual speed. It continuously adjusts the armature voltage of a DC motor via a PID control signal.

Practice Problems

1. What is the number of high-performance DC-motor drives provided by the Arduino MKR motor carrier board? Select the correct answer.
    1. 2
    2. 4
    3. 6

1. The Arduino MKR 1000 Wi-Fi board sends the PWM motor control signal to the MKR motor carrier board via which of the following? Select the correct answer.
    1. I2C
    2. SPI
    3. Digital timer/counter channels

1. The DC-motor encoder signal was sent to the Arduino MKR 1000 Wi-Fi board by the MKR motor carrier board via which of the following? Select the correct answer.
    1. I2C
    2. SPI
    3. Digital timer/counter channels

1. List two approaches from this section used when a closed-loop mechatronic control software is to be implemented into an Arduino MKR microcontroller.

This page titled 7.4: Mechatronic Control System Architecture- Speed Control Example is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process

Learning Objectives

After completing this section, you will be able to:

1. Describe an iterative design process for mechatronic systems to meet desired system specifications.
2. Explain the difference between a simple (rough) mechatronic system model and its complex (detailed) version.
3. Use the iterative design process for mechatronic systems to achieve required system specifications and performances.

This section discusses the iterative design process for mechatronic systems shown in Figure 7.23. Called the V-design process, it is very important for the development of a model-based mechatronic system. This process starts with the required mechatronic system specifications. It ends with a designed mechatronic system, resulting in a life-cycle optimized product. The V-design process starts with a rough mathematical model. It then develops a detailed, complex mathematical model, also called a digital twin of the physical system. The process helps us evaluate closed-loop mechatronic system performances both by adding actuator and sensor dynamics and by considering the detailed physical system dynamics and nonlinearities.

Table 7.2 summarizes the three major steps from Figure 7.23 as well as the substeps in the process.

Step 1 in the design process shown in Figure 7.23 develop a conceptual design (Step 1.2) based upon the system specifications (Step 1.1), which are given. This step establishes the mechanical system concept in principle. Although the target microcontroller is not yet selected, the associated control system architecture is determined. This includes as actuator (control input) and a sensor measure (feedback signal).

Step 2 contains six blocks, going from the rough mathematical model to design optimization. It may need a few iterations to achieve the design optimization stage. We will discuss these blocks below.

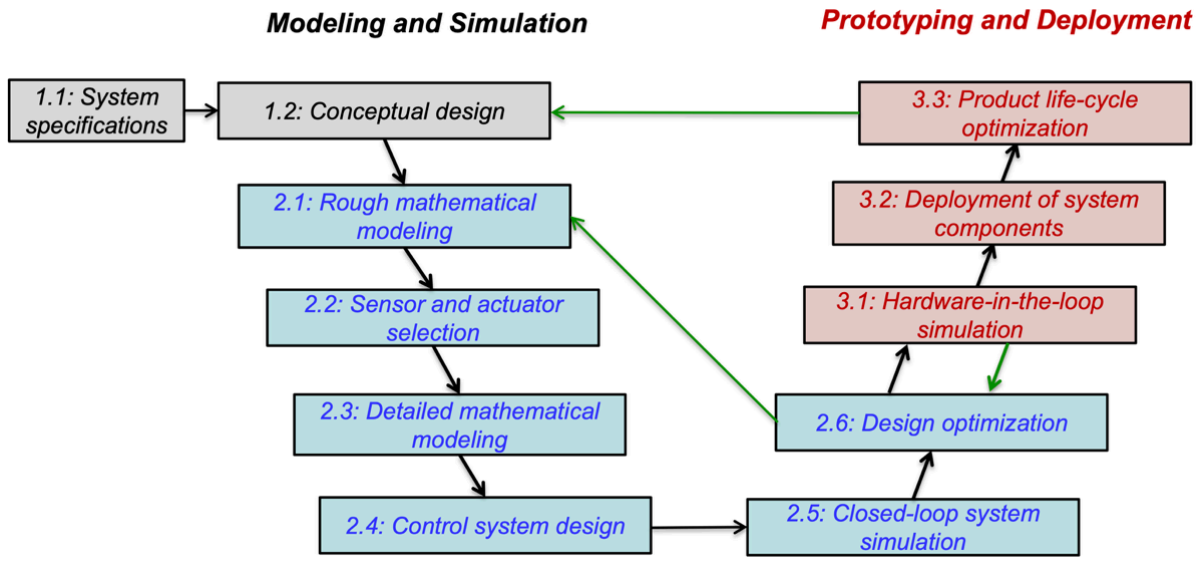


Figure 7.23: The V-design process for mechatronic systems.

Step 2.1: Rough mathematical model. In this step, we develop a general model for the physical system. We can use this to conduct a simulation study for the feasibility and performance of the conceptual design from Step 1 or the design optimization in Step 2.6. If the simulation results indicate a feasible design with satisfactory performance, we can proceed to the next step.

Step 2.2: Sensor and actuator selection. Next, we select the actuators and sensors based upon the system requirement.

Throughout Step 2, we must develop a detailed physical system model (a digital twin of the physical system) so that we can conduct a detailed simulation. When selecting each actuator, consider the output ranges (e.g., DC-motor peak torque and speed) and noise level (e.g., DC-motor torque ripple) needed to meet the system requirements. When selecting each sensor, consider its measurement range (e.g., speed range), accuracy (e.g., digital speed sensor resolution, and nonlinearity), and noise level (e.g., current sensor noise due to external magnetic field).

Step2.3: Detailed mathematical modeling. After selecting the actuators and sensors, we develop a detailed mathematical model. The physical system model should contain system dynamics as detailed as possible. These include system nonlinearity, thermal sensitivity, and environmental effects of the mechatronic system (e.g., temperature, humility, pressure). These actuator models should contain the actuator nonlinearity (e.g., saturations on torque and speed) and actuator output noise (e.g., torque ripple). The sensor models should contain the details of how these measurements are generated, with the associated noise levels, and how the sensor signals are converted into digital information.

Step 2.4: Control system design. After developing the detailed mathematical model of the physical system, our next step is to design an open- or closed-loop control strategy to achieve the designed system performance. This involves selecting a target microcontroller (see Section 7.2) with enough throughput to handle the control of the desired mechatronic system features with enough analog and digital input and output channels at the desired sample period. In addition, some sensors and actuators require serial communication with the microcontroller. Hence, the selected microcontroller should have the proper serial communication capabilities comparable with the selected sensors and actuators. At this state, control system design considers the use of model-based and model-free approaches. For model-based control, these controllers must be designed to meet mechatronic system performance requirements such as the settling time and percentage overshoot within the sensor and actuator operational ranges.

Step 2.5: Closed-loop system simulation. After designing the control system along with the detailed mathematical model, we next conduct a simulation study, also called software-in-the-loop (SiL) simulation. This study is based on a case study list generated from the system's requirements. Here, we use different control strategies and system sample periods to optimize mechatronic system performance. If the controller is designed in the continuous-time domain, we will study the effect of discretizing a continuous-time controller at different sample periods to find the optimal sample period—the minimal sample frequency with satisfactory discretization error. Other simulation studies introduce system noises, measurement noises, and system uncertainties (modeling errors) to review the control system's robustness under these effects. With recent improvements in rapid computer simulation capability, most design validation can be accomplished by SiL simulation studies. As a rule of thumb, we spend about 60% of development time to simulate validation and optimization.

Step 2.6: Design optimization. After simulating the current design, we determine whether the performance results are satisfactory. If not, we need a design iteration. We would modify the mechatronic system to improve and optimize it. This iterative process starts with Step 2.1, "Rough mathematical model." In some cases, even though the design of the mechatronic system meets system requirements, we still work harder to optimize the system in order to generate improved performance.

The design process now proceeds to Step 3 if the designed mechatronic system has satisfactory system performance and the decision is made to use the current design for mechatronic system prototyping. Step 3, shown in Figure 7.23, consists of the three red blocks, discussed below.

Step 3.1: Hardware-in-the-loop simulation. After completing the simulation study with satisfactory system performance, the next step is to conduct a hardware-in-the-loop (HiL) simulation study [29], [30], and [31]. In this study, the selected system actuators and sensors will be deployed and connected to the target microcontroller with proper actuator drives and sensor signal conditioning devices. We use an HiL real-time simulation computer to generate the actuator loads and all sensor signals based upon the detailed mathematical system model developed in Step 2. In certain cases, the detailed mathematical model needs to be simplified to make real-time simulation possible. The purpose of the HiL simulation study is two-fold. First, it verifies if the selected actuators and sensors work with the target microcontroller and its amplifiers (driving circuits). Second, it validates the implemented microcontroller software.

Step 3.2: Deployment of system components. At this stage, the physical mechatronic system is assembled, along with the selected microcontroller, actuator drives, and sensor signal conditioning electronics. We have validated the complete system. Now any deployment issue will be debugged and fixed. At this stage, we develop a detailed test plan for validating the designed mechatronic system. The output of this block is a working mechatronic system that is ready for life-cycle (such as durability) optimization.

Step 3.3: Product life-cycle optimization. Now that we have developed and validated a working prototype, in this step we will manage the developed product throughout its entire lifespan. We will adapt marketing and development strategies at each stage

to maximize its profitability and market presence. This step ensures the developed product remains relevant and competitive throughout its life cycle.

By the end of Step 3, if the final design and prototype have not met the system's requirements based upon the current design, we will generate a revised design to go back through Steps 1 to 3. Due to the availability of the first iterative design, the second iteration will be completed in relatively short time; most of the existing models and prototyping components are available. Example 7.3 illustrates this process.

Example 7.3

For this example, we will essentially go through the mechatronic system design process shown in Figure 7.23, concentrated on Step 2. (The prototyping process in Step 3 varies significantly, depending on the type of mechatronic system.) Consider designing a three-wheel cart driving system with two active wheels. There are three design targets. First, for a maximum total vehicle mass of 6.5 kg with a certain payload, the vehicle should reach its peak speed of 0.6 m/s with a 5% settling time of 1.5 second. Second, the vehicle should be able to turn with differential steering. Third, the vehicle should be powered by a 12-volt battery.

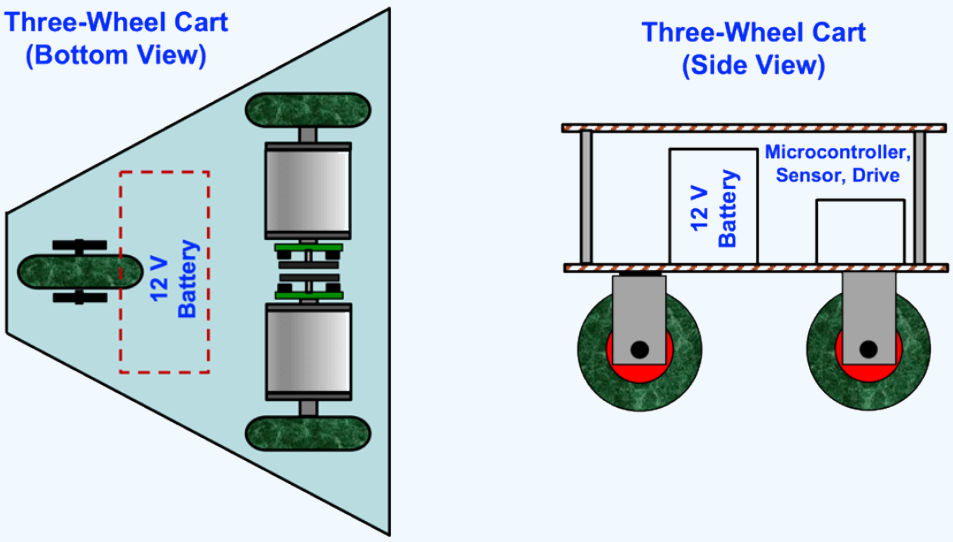


Figure 7.24: An initial design for a three-wheel cart.

For design Step 1, we need to generate a rough design for both the mechanical and electrical parts of this mechatronic system. Figure 7.24 shows a rough-designed drawing for the three-wheel cart. Due to the requirements of differential steering, both forward and backward driving, the three-wheel cart is designed with two front wheels powered by two geared DC motors. The differential steering can be realized by running the two front wheels in the opposite linear directions; operating both DC motors in the same linear direction leads to straight forward and backward movements. To increase the driving toque, a speed reduction gearbox is used in front of the DC-motor axis between the DC-motor axis and wheel axis. The 12-volt battery, microcontroller, certain sensors, and motor drive circuits are located inside the middle bay.

With Step 1 completed, we can now move to Step 2.1, developing a simplified system model so that certain system-related motor and sensor parameters can be determined through SiL simulations.

Assuming there is no wheel-slip on the road, the top plot of Figure 7.25 shows an equivalent vehicle model diagram. The two DC-motor circuits are merged into one. Assuming only forward or backward driving condition, the moment of inertia of the two driving wheels and motor rotors is lumped into a single moment of inertia J. Total vehicle mass m includes all cart components. All frictions, including rolling resistance, are lumped into damping coefficient b_m.

After this rough model is developed, we need to determine the model parameters so that a preliminary simulation can be conducted to validate the design feasibility. As noted in Section 2.4, the DC-motor inductance L_a can be assumed to be zero; it is normally much smaller than its resistance R_a, which is assumed to be about 5 Ohms. Before determining preliminary

motor coefficients K_τ and K_m, we must determine the equivalent moment of inertia $J + mR^2$. The wheel radius based on the rough design is about $R = 0.04\ m$. Assuming vehicle total mass to be $7\ Kg$, the equivalent moment of inertia is $mR^2 = 0.0112\ Kg \cdot m^2$. We will round that up to $J_{total} = 0.015\ Kg \cdot m^2$ to include the motor rotor and wheel moment of inertia. We will also assume that, at the target vehicle speed, the total friction torque will be around 6% of the motor peak torque.

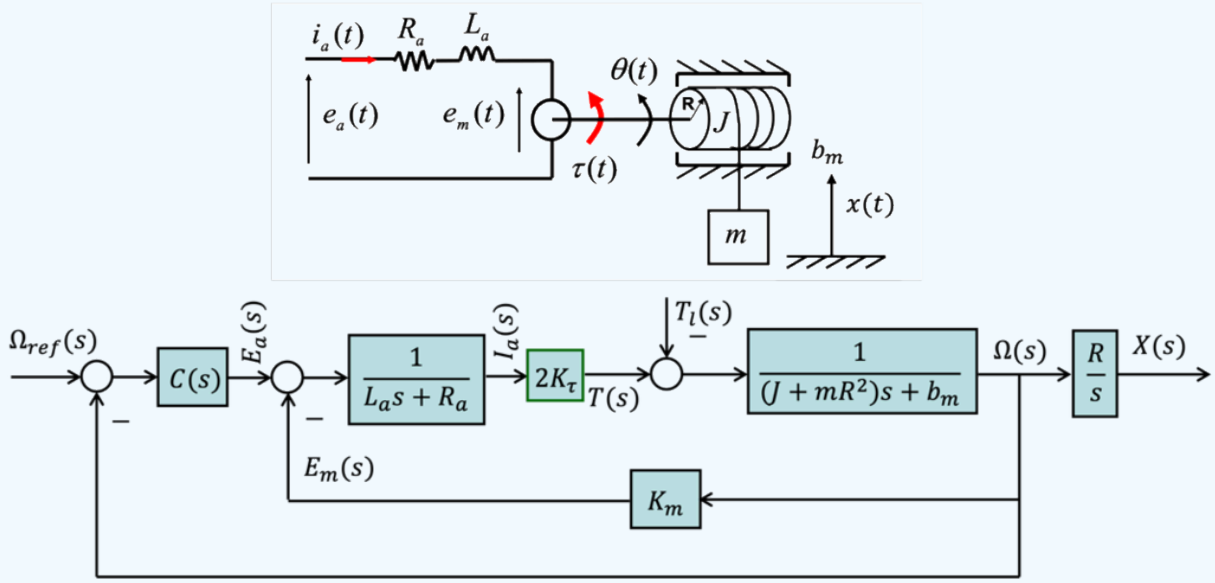


Figure 7.25: A simplified mechatronic system model (block diagram and Simulink implementation).

Now we need to find the minimum K_τ and maximum K_m that defines the minimum requirements for the DC motors. Based upon the DC-motor torque curve defined in Section 5.2, the DC-motor output torque is a function of its motor speed as below, assuming that the inductance is zero ($L_a = 0$).

$$\tau = \frac{2K_\tau}{R_a}(v_a - K_m\omega) \tag{7.5.1}$$

where the number 2 before the K_τ is due to the fact that there are two DC motors in the three-wheel cart. Assume that the target cart speed is reached at γ percent of the maximal motor speed $\omega_{no-load}$ with $v_{a\_max} = 12\ V$. As a result, we have

$$\omega_{no-load} = \frac{v_{v-max}}{K_{m-max}} = \frac{12\ (volt)}{K_{m-max}} = \frac{v_t}{\gamma R} = \frac{1.2\ (m/s)}{0.04\gamma\ (m)} = \frac{30}{\gamma}\ (rad/s) \tag{7.5.2}$$

where γ is a number between 0 and 1. Therefore,

$$K_{m-max} = \frac{12\ (volt)}{30\ (rad/s)}\gamma = 0.4\gamma\ (volt \times s/rad) \tag{7.5.3}$$

To find $K_{\tau-min}$ (minimal K_τ), we are going to assume that the friction coefficient equals zero ($b_m = 0$). The following dynamic equation can be obtained based upon the Newton's law.

$$J_{total}\dot{\omega} = \tau = \frac{2K_\tau}{R_a}(v_a - K_m\omega) \tag{7.5.4}$$

The following transfer function can be obtained after taking the Laplace transform of equation 7.5.4 and simplifying it.

\[\dfrac{Ω(s)}{V_a(s)}=\dfrac

ParseError: EOF expected (click for details)

```
Callstack:
    at (Bookshelves/Introductory_Engineering/Mechatronics:_Fundamentals_Design_Integration_a
```

=\dfrac{1}{K_m(Ts+1)}, T=\dfrac{R_aJ_{total}}{2K_mK_τ} \label{7.10}\]

With the design goal of 5% settling time of 1.5 second, we have the following.

$$3T = 1.5\ \ (second)\ \ \rightarrow\ \ K_{\tau-min} = \frac{3}{1.5}\frac{R_a J_{total}}{2K_{m-max}} = \frac{3\times 5\times 0.015}{1.5\times 2\times 0.4\times\gamma} = \frac{0.75}{4\gamma}\ (Nm/A) \tag{7.5.5}$$

Next, we develop a model for a simulation study to confirm these selected parameters. The transfer function from $E_a(s)$ to $\Omega(s)$ is shown below with K_{m-max} and $K_{\tau-min}$ substituted, assuming $L_a \approx 0$ and $b_m \approx 0$.

$$G(s) = \frac{\Omega(s)}{E_a(s)} = \frac{2K_\tau}{R_a(Js+b_m)+2K_\tau K_m} = \frac{1.5/\gamma}{0.075s+0.15} = \frac{10/\gamma}{0.5s+1} = \frac{K}{Ts+1} \tag{7.5.6}$$

As a result, the time constant is about 0.5 second and the 5% settling time t_s is about $3T = 1.5$ seconds with satisfactory performance.

In the next step (Step 2.2) we select the speed sensor and DC-motor actuator so that we can develop a complex mathematical (nonlinear) model. Assuming that there is no slip between tire and road, we need to select DC motors and associated motor speed sensors for these two motors. For the speed sensor, the hall-effect sensor is a good choice due to its high signal-to-noise ratio at high speed. However, the hall-effect sensor signal-to-noise ratio reduces as the axis speed goes down for a given sample rate. As the sample period reduces, the number of pulses available to be counted during that period also reduces. Therefore, selecting a hall effect sensor depends on the control system sample rate and minimal motor shaft speed. The control sample frequency of the three-wheel cart is about $f_h = 1/(2\pi T)\ Hz$. We can assume that the highest control loop sample frequency of 50 Hz. It is desirable to have at least 10 pulses at the vehicle speed of $0.1\ m/s$. We can calculate the corresponding wheel speed as follows.

$$\omega_{wheel}(0.05) = \frac{v_{vehicle}}{R_{wheel}} = \frac{0.1}{0.04} = 2.5\ (rad/s) \tag{7.5.7}$$

With a sample period of $T_s = 0.02$ second, the wheel rotation angle θ_{wheel} over each sample period is $\theta_{wheel} = 0.02\times 2.5 = 0.05$ (rad). Therefore, to have at least 10 pulses, we can calculate the number of pulses per wheel rotation can as below.

$$n_{p-req} = \frac{2\pi}{\theta_{wheel}}\times 10 = 1257 \tag{7.5.8}$$

The selected motor and encoder set is model B09MFT3VT7 (or JGB37-520-12V-333RPM) with DC-motor parameters $K_\tau = 0.00767\ (Nm/A)$ and $K_m = 0.81\ (V\times s/rad)$ that are not satisfactory for the required $K_{\tau-min} = 0.15\ (Nm/A)$ and $K_{m-max} = 0.1\ (Vs/rad)$ with any $0<\gamma<1$. As a result, we select a gearbox with a speed reduction ratio of $30:1$ at the motor driving shaft to move both parameters into the feasible ranges as below with $\gamma = 0.5$.

$$K_{\tau-g} = 30K_\tau = 0.23 > K_{\tau-min} = 0.2, K_{m-g} = \frac{K_m}{30} = 0.027 < k_{m-max} = 0.375 \tag{7.5.9}$$

<="" p=""> </k_{m-max}=0.375\>

For the encoder speed sensor, the selected available hall-effect sensor is an eleven-pulse per revolution one. In this case, one revolution of the gear-box output shaft leads to 330 (30×11) pulses, which is below the required 1257. A gearbox with ratio of 1:4 from motor to encoder shaft is employed to make

$$n_p = 11\times 4\times 30 = 1320\ >\ n_{p-req} = 1257\ (\ of\ pulse\ per\ revolution) \tag{7.5.10}$$

Now we develop a detailed mathematical model for simulation validation (Step 2.3). We will begin by developing the three-wheel cart equations of motion (EOMs). Figure 7.26 shows the cart free body diagram.

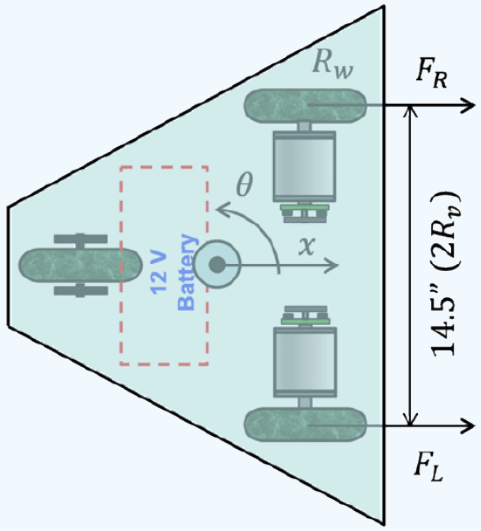


Figure 7.26: Free body diagram of the three-wheel cart.

Assuming that there is no slip between wheel and ground, the following EOMs can be developed based upon Newton's law, with $v = \dot{x}$ and $\omega = \dot{\theta}$.

$$m\dot{v} + b_v v = f_L + f_R, J_v\dot{\omega} + b_\theta\omega = (f_L - f_R)\, R_v \tag{7.5.11}$$

where $m = 6.5\ (Kg)$ and $J_v \approx 0.02\ (Kg{\times}m^2)$ are cart mass and moment of inertia around the center of mass, respectively; $b_v \approx 0.1\ (Ns/m)$ and $b_\theta \approx 0.01\ (Nms/rad)$ are the cart mass and moment of inertia around center of mass, respectively; and R_v, f_L, and f_R are the half distance between the two driving wheels, left and right wheel driving forces marked on Figure 7.26, respectively. As a result, the following transfer functions can be derived for the three-wheel cart.

$$V(s) = \frac{1}{ms + b_v}\left(F_L(s) + F_R(s)\right), \Omega(s) = \frac{1}{J_v s + b_\theta}\left(F_L(s) - F_R(s)\right) R_v \tag{7.5.12}$$

where $V(s)$ and $\Omega(s)$ are Laplace transforms of $v(t)$ and $\dot{\theta}(t)$, respectively, and $F_L(s)$ and $F_R(s)$ are Laplace transforms of f_L and f_R. After the three-wheel cart linear movement speed $v(t)$ and its rotational movement $\dot{\theta}(t)$ around the vertical axis are calculated, the three-wheel cart linear speed at each wheel can be calculated as below.

$$v_L(t) = v(t) + R_v\dot{\theta}(t), v_R(t) = v(t) - R_v\dot{\theta}(t) \tag{7.5.13}$$

where $R_v = 0.18415\ (m)$ is the half distance of the two front driving wheels (see Figure 7.26), and $v_L(t)$ and $v_R(t)$ are the three-wheel cart linear speeds at the left and right wheels. Now the corresponding wheel speed can be calculated as below.

$$\dot{\theta}_L(t) = \omega_L(t) = \frac{v_L(t)}{R_w}\dot{\theta}(t), \dot{\theta}_R(t) = \omega_R(t) = \frac{v_R(t)}{R_w} \tag{7.5.14}$$

where $R_w = 0.0381\ (m)$ is the radius for both driving wheels, $\dot{\theta}_L(t) = \omega_L(t)$ and $\dot{\theta}_R(t) = \omega_R(t)$ are the left and right driving wheel angular speeds, respectively.

Once all the mathematical equations have been derived for the detailed model, we can use Simulink to develop a detailed simulation model. See Figure 7.27, where the model includes both DC-motor dynamics (labeled "Motor dynamics"). Here, $R_a = 5.0\ (\Omega)$, $L_a = 2\ (mH)$, $K_\tau = 0.23\ (Nm/A)$, and $K_m = 0.027\ \ (Vs/rad)$. Note that the torque and back-EMF coefficients used are these equivalent ones for the geared output shaft.

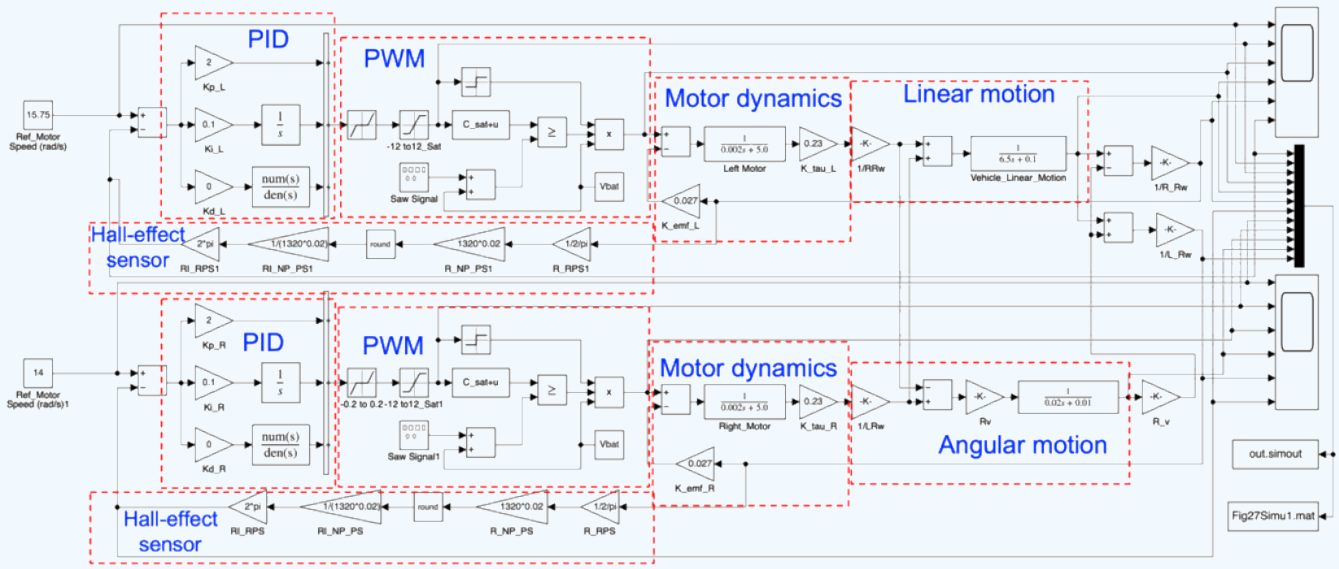


Figure 7.27: Detailed vehicle model for validation simulations.

For the three-wheel cart, the vehicle dynamics is modeled using equation 7.5.12 for both linear and rotational motions; see the red-dashed line blocks marked “Linear motion” and “Angular motion,” respectively. We can use equation 7.5.12 to calculate the overall three-wheel cart speed $v(t)$ and angular speed $\omega(t)$; equation 7.5.13 to calculate the linear cart speed signals $v_L\,(t)$ and $v_R\,(t)$ at both driving wheels; and equation 7.5.14 to calculate both wheel speed signals $\omega_L(t)$ and $\omega_R(t)$. See the right side of the “Linear motion” and “Angular motion” blocks in the Simulink model.

Both DC motors will be controlled by PWM signals (see Chapter 5). Therefore, we need to simulate the effect of PWM control signals in the detailed model, especially for the effect of the selected PWM frequency. Figure 7.27 shows two red-dashed blocks marked as “PWM” for both DC-motor models. Within each of these blocks, the PWM driving signal of the DC motor is generated using a 12-volt saw wave generated by Simulink signal generation block, where its peak is 12 volts and its bottom is –12 volts. The saw wave signal is then shifted by a 12-volt battery signal to make it a saw wave signal with its peak equal to 24 volts and bottom 0 volt. To generate a PWM signal with its duty-cycle proportional to the saturated control signal, we shift the saturated control signal with a range of $\pm C_{sat}$ by C_{sat}. The shifted control then has a range between 0 and $2C_{sat}$ volt. If $C_{sat} = 12$ volts, the shifted control signal has a range between 0 and 24 volts, the same as that of the shifted saw wave. The shifted control signal is compared with the shifted saw wave. If the control signal is greater than or equal to the saw wave signal, the logical output is 1. Otherwise, it’s zero. Finally, we can obtain the PWM signal output by multiplying the battery voltage by the logical output.

To simulate digital angular speed of the hall-effect sensor, we convert the continuous-time wheel speed signal to the number of pulses for a given sample period $T_s = 0.02\ (s)$. Divide by 2π, then multiply the number of pulses per motor revolution (1320) and sample period (0.02 s). Then round off the real number to the closest integer. Next, apply the reverse process, dividing $1320 \times T_s$ and multiplying by 2π. Through this process, we simulate the roundoff effect of the digital hall-effect sensor. Figure 7.27 has two red-dashed line blocks, named Hall-effect sensor, for roundoff operations for the two encoders mounted on the DC motors. Modeling the roundoff error of the hall-effect sensor shows the effect of the hall-effect sensors.

The last two red-dashed line blocks in Figure 7.27, named PID, are the two PID controllers. They take the DC-motor speed signals measured by hall-effect sensors as feedback signals; the error signal was calculated by subtracting the measured DC-motor speed signal from the given reference motor speed. Due to slow cart dynamics, the derivative gains of both PID controllers are set to zero; the proportional and integral gains are set to 2 and 0.1 respectively for both controllers.

The detailed mathematical model developed above can be further improved by adding sample and hold mechanism so that a digital controller can be simulated. With the detailed model now developed, we can move to Step 2.5 for simulation studies to evaluate the system’s performance. In practice, we should develop a test plan to validate the design. This example conducts two simulation studies: linear acceleration motion and angular acceleration motion. Figure 7.28 shows the linear forward motion simulation results. The reference wheel speed for both left and right are $15.75\ (rad/s)$. With wheel radius R_w= $0.0381\ (m)$, the target vehicle linear speed under these wheel speeds is

$$v = \frac{(\omega_R + \omega_L)R_w}{2} = \frac{(15.75 + 15.75) \times 0.0381}{2} = 0.6 \ (m/s) \tag{7.5.15}$$

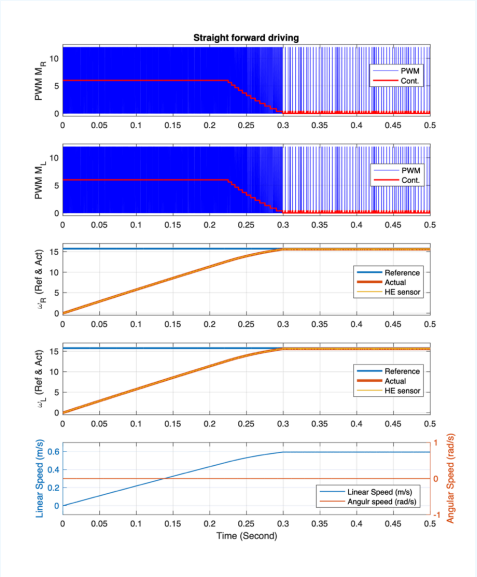


Figure 7.28: Simulation results for step linear vehicle speed.

Figure 7.28 has five plots. The top two plots (red lines) show the continuous-time saturated PID control outputs for both left and right motor speed control. The blue pulses are the PWM drive signals for both left and right motors. In this simulation, the continuous-time control signals are saturated at 6 volts to simulate a situation with low-voltage power supply. We can see that when the control voltage is saturated at 6 volts, the PWM drive signals are at 50% duty-cycle. As the magnitude of continuous-time control signal reduces, the corresponding PWM signal duty-cycle also reduces. After 0.3 second, the vehicle reaches its steady-state; the duty-cycle of the PWM signal reduces to a level that maintains the vehicle at its target speed of 0.6 (m/s). Also, the control signal is saturated at the beginning, leading to a nonlinear simulation.

The third and fourth plots show the reference and simulated left and right wheel speed signals. The blue lines are the reference wheel speed signals for the left and right wheels. The red lines represent the simulated speeds of the left and right wheels. The two thin orange lines are the simulated speeds of the left and right wheels measured by the hall-effect sensors. We can see that the speed signals measured by hall-effect sensors fluctuate around the simulated speed signals due to a finite number of pulses. It takes about 0.3 second for both wheels to reach their steady-state angular speeds of $15.75 \ rad/s$. Both wheels accelerate synchronously to reach the steady state speed. Recall that the simulation was conducted assuming no wheel slip; with slip, the acceleration will be slower.

The bottom graph of Figure 7.28 shows both the simulated linear cart speed and angular cart rotation rate (speed). The cart speed reaches $0.6 \ m/s$ within 0.3 second, which is less than the required 5% settling time of 1.5 seconds. As a result, both design goals of maximum cart speed of $0.6 \ m/s$ and 5% settling time of 1.5 second are satisfied. Because both motor speeds are synchronous, the angular cart speed is zero.

Figure 7.29 shows the simulation results with different reference wheel speeds and five subplots. (The left wheel is 14 rad/s and the right wheel is 15.75 rad/s). The top two subplots show the continuous-time saturated PID control outputs (red lines) and PWM drive signals (blue lines) for both left and right motors. In this simulation, the continuous-time control signals are also saturated at 6 volts. When the control voltage is saturated at 6 volts, the PWM drive signals are at 50% duty-cycle. As the magnitude of continuous-time control signal reduces, the corresponding PWM signal duty-cycle also reduces. After 0.3 second, the vehicle reaches its steady state, and the duty cycle of the PWM signal reduces. Different from Figure 7.28 due to different reference wheel speeds for the two wheels, the continuous-time control signal of the left motor decays earlier than the right one due to its reference speed being slower than the right one.

The third and fourth subplots of Figure 7.29 show the reference and simulated wheel speed signals, with blue lines for reference wheel speed signals and two red lines for simulated wheel speeds. Two thin orange lines present the simulated wheel speeds measured by the hall-effect (HE) sensors with digitization errors. Due to the control saturation, the two wheels

accelerate at the same rate. However, after about 0.2 second, the left wheel acceleration becomes smaller than that of the right one due to its lower reference speed. Similarly, it takes about 0.3 second for both wheels to reach their steady-state angular speeds of 15.75 and 14 rad/s.

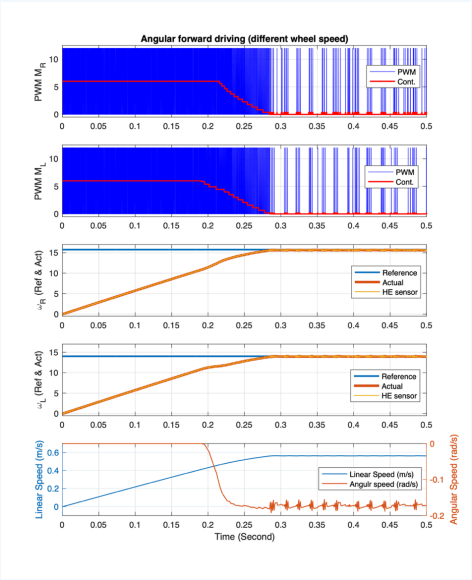


Figure 7.29: Simulation results for step angular vehicle motion.

The bottom subplot of Figure 7.29 also shows both simulated linear cart speed and angular cart rotation rate (speed). The cart speed reaches $0.5647\ m/s$ within 0.3 second. The angular speed of the cart starts at zero due to synchronous acceleration of both wheels until around 0.2 seconds. It then gradually reduces down to $-0.1667\ rad/s$ with certain oscillations due to the sparse PWM pulse at the steady state. This simulation results suggest using high PWM frequency.

In Step 2.6, we decide if the design needs to be optimized. If so, the process will go back to Step 2.1 (Rough mathematical model) to start a new design iteration. If the design is satisfactory, we move to Step 3.

In Step 3.1, we conduct an HiL simulation study. Our purpose is to select a proper microcontroller. We also want to select the associated drives (power amplifiers) for the selected actuators and the associated input channels for the selected sensors (Step 2.2). There are various levels of HiL simulation studies we can develop. They have different purposes, from validating the selected microcontroller and associated electronic systems to the entire mechatronic system. The HiL system is widely used due to its simplicity.

Figure 7.30 shows an HiL system developed to validate the microcontroller capability and associated electronic components. Here, the Arduino MKR 1000 Wi-Fi microcontroller and an Arduino MKR motor carrier board are used. The system also includes two DC motor and hall-effect sensor assemblies (see the right-side of Figure 7.30. Now with this HiL system in place, we can develop the associated two-motor speed control software in Simulink by doubling the motor speed controller shown in Figure 7.21. By running the HiL simulations, we can validate the feasibility of using the Arduino MKR 1000 Wi-Fi and its motor carrier board.

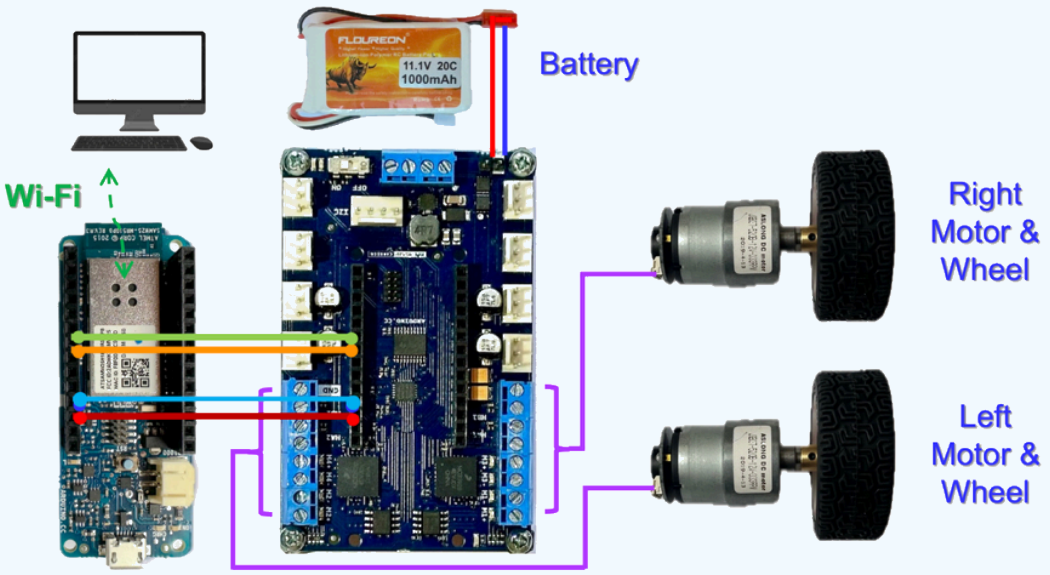


Figure 7.30: A hardware-in-the-loop (HiL) simulation system.

Next, in Step 3.2, we deploy all the selected hardware to the designed prototype three-wheel cart shown in Figure 7.24. Then we conduct validation tests on the prototype similar to these described in Step 2.5. If the prototype's performance is not satisfactory, we improve the mechatronic system until the performance is satisfactory. The actual deployment of the three-wheel cart deployment can be found in Chapter 9.

After meeting the prototype's performances, the next step (Step 3.3) is to optimize the prototype system for massive production. For example, the current three-wheel cart speed settling time is about 0.3 second with a requirement of 1.5 second. Therefore, to reduce system cost, we would look for a smaller motor with less torque output than the already-selected one. With any modification, we conduct a design iteration. Because the system model and simulation platform have been mostly developed, the next iteration is fairly fast; we can obtain the final prototype results.

Section Summary

This section presents a model-based development process for mechatronic systems, called a V-design process. This process begins with the required system specifications and ends with a validated system design, resulting in a life-cycle optimized product. Using an iterative design approach, the product moves from a rough mathematical model to a detailed, more complex model (also called a digital twin). This model helps evaluate closed-loop system performances by adding actuator and sensor dynamics and by considering the detailed physical system's dynamics and nonlinearities. The fully designed mechatronic system is then validated through SiL and HiL simulation studies, followed by prototyping the system.

Section Keywords

Detailed system model: Also called a digital twin, this dynamic, virtual replica of a mechatronic system is created and validated using real-time data from sensors and other devices.

Hardware-in-the-Loop (HiL) simulation: This technique validates mechatronic control systems by connecting real controller hardware to a virtual, real-time simulated environment. It accelerates development and improves safety by enabling testing of control algorithms without needing the full, physical system.

V-design process: This framework for mechatronic systems combines hardware and software development. Each development phase is paired with either simulation or experimental validation to ensure early identification of design frauds. An iterative process is also embedded in the V-design process to improve system performance.

Practice Problems

1. Select one correct response from the following list. In Step 2 of the mechatronic system development process, what two simulation studies are conducted?
    1. Simulation study using the rough mathematical model and software-in-the-loop simulation based upon the detailed mathematical model.
    2. Software-in-the-loop simulation based upon the detailed mathematical model and hardware-in-the-loop simulation.
    3. Simulation study using the rough mathematical model and hardware-in-the-loop simulation.

1. How many iteration-loop is in the mechatronics system development process? Select the correct answer.
    1. 1.
    2. 2.
    3. 3.

1. Which mathematical model is used for control design and closed-loop system performance simulation? Select the correct answer.
    1. Rough mathematical model.
    2. Detailed mathematical model.

This page titled 7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

7.6: Chapter References

References

1. C. Gamache, M. Van Nieuwstadt, J. Martz, and G. Zhu, “LQTI boost pressure and EGR rate control of a diesel air charge system with eBoost assistance,” *Int. J. of Engine Res.,* vol. 25, no. 9, pp. 1659–1670, Sep. 2024, doi: 10.1177/14680874241241364.
2. J. Chotai and K. Narwekar, “Modelling and position control of brushed DC motor,” in *Proc. 2017 Int. Conf. on Adv. Comput., Commun., and Control (ICAC3)*, Mumbai, India, Dec. 2017, pp. 1–5, doi: 10.1109/ICAC3.2017.8318792.
3. S. Qu, G. Zhu, W. Su, and S. S-M. Swei, “LPV model-based adaptive MPC of an eVTOL aircraft during tilt-transition subject to failure,” *Int. J. of Control, Autom., and Syst.,* vol. 21, no. 2, pp. 339–349, Jan. 2023, doi: 10.1007/s12555-021-0915-1.
4. C. Lin, E. Siampis, and E. Velenis, “Path-tracking control at the limits of handling of a prototype over-actuated autonomous vehicle,” *Int. J. of Veh. Syst. Dyn.*, vol. 63, no. 5, pp. 853–875, May 2024, doi: 10.1080/00423114.2024.2361726.
5. R. Oshana (editor), *DSP for Embedded and Real-Time Systems*, Boston, MA, USA: Newnes, 2012, ch. 3, ISBN-13: 978-0-1238-6535-9.
6. N. M. Morris and M. Noel, *Microelectronic and Microprocessor-based Systems*, London, United Kingdom: Macmillan Education UK, 1985, ISBN-13: 978-0-3333-6190-0.
7. Arduino MKR 1000 Wi-Fi microcontroller, Accessed Jan. 17, 2025. [Online] Available: https://store-usa.arduino.cc/product...3ifuAoM7XW9Tsy
8. Arduino IDE software, Accessed: Jan 17, 2025. [Online] Available: https://www.arduino.cc/en/software
9. D. Xue and F. Pan, *MATLAB and Simulink in action: programming, scientific computing and simulation,* 2024 ed., Singapore: Springer, 2024, ISBN-13: 978-9-8199-1175-2.
10. ARM 32-bit Cortex-M0+ MCU, Accessed: Jan 17, 2025. [Online] Available: https://www.st.com/content/st_com/en...x-m0-plus.html
11. Arduino UNO Q data sheet, Accessed: Dec. 17, 2025. [Online] Available: https://docs.arduino.cc/resources/da...gyMDA5NTA2MDg3
12. Arduino MKR Motor Shield (Carrier) board, Accessed: Jan 22, 2025. [Online] Available: https://store-usa.arduino.cc/product...f91tUDBsREKBez
13. R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 12th ed., Englewood Cliffs, NJ, USA: Prentice Hall, 2011, ISBN-13: 978-0-1360-2458-0.
14. H. Shi and T. Shen, “Design and experimental validation for nonlinear control of internal combustion engines with EGR and VVT,” *SICE J. of Control, Meas., and Syst. Integr.,* vol. 14, no. 1, pp. 51–58, Mar. 2021, doi: 10.1080/18824889.2021.1874680.
15. Z. Ren and G. Zhu, “Modeling and control of an electrical variable valve timing actuator,” *ASME J. of Dyn. Syst., Meas., and Control*, vol. 136, no. 2, p. 021015, Mar. 2014, doi: 10.1115/1.4025914.
16. Z. Ren and G. Zhu, “PRBS closed-loop system identification error with integration control,” *Proc. of Inst. of Mech. Eng., Part I, J. of Syst. Control Eng.*, vol. 223, no. 6, pp. 887–884, 2009, doi: 10.1243/09586518JSCE794.
17. J. von Neumann, “First draft of a report on the EDVAC, 1945,” Accessed: Dec. 30, 2025. [Online] Available: https://web.mit.edu/sts.035/www/PDFs/edvac.pdf
18. T. Noergaard, *Embedded systems architecture: a comprehensive guide for engineers and programmers,* Boston, MA, USA: Newnes, 2005, ISBN-13: 978-0-7506-7792-9.
19. D. Brash, “Extensions to the ARMv7-A architecture,” in *Proc. 2010 IEEE Hot Chips 22 Symp. (HCS),* Stanford University, CA, USA, Aug. 2010, pp. 1–21, doi: 10.1109/HOTCHIPS.2010.7480070.
20. A collection of application examples, Accessed: Jan. 19, 2025. [Online] Available: https://www.arduino.cc/en/Tutorial/BuiltInExamples
21. A project hub of various applications, Accessed: Jan. 19, 2025. [Online] Available: https://create.arduino.cc/projecthub
22. ATSAMW25 datasheet, Accessed: Jan. 19, 2025. [Online] Available: https://ww1.microchip.com/downloads/..._Datasheet.pdf
23. Eagle circuit files of the Arduino MKR Motor Carrier board (open source), Accessed: Jan. 23, 2025. [Online] Available: https://content.arduino.cc/assets/MK...erV3.0-sch.pdf.
24. Eagle PCB layout files of the Arduino MKR Motor Carrier board (open source), Accessed: Jan. 23, 2025. [Online] Available: https://content.arduino.cc/assets/MK...arrierV3.0.zip
25. Matlab/Simulink library for Arduino MKR Motor Carrier board, Accessed: Jan. 25, 2025. [Online] Available: https://docs.arduino.cc/libraries/ar...omotorcarrier/
26. Y. Wang and G. Zhu, “Performance improvement of an NMP mini Segway using sample and hold,” *MPDI J. of Appl. Sci.* (*Special Issue of Recent Adv. in Mech. Vib. and Control*), vol. 13, no. 2, p. 1070, Feb. 2023, doi: 10.3390/app13021070.
27. Y. Wang, G. Zhu, and R. Mukherjee, “Performance improvement demonstration of an NMP system using sample and hold inputs," *Int. J. of Dyn. and Control*, vol. 9, no. 1, pp. 109–120, Mar. 2021, doi: 10.1007/s40435-020-00643-w.

28. G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic System,* 2nd ed., New York, NY, USA: Addison-Wesley, 1990, ISBN-10: 0-20-151884-8.
29. T. Hwang, J. Rohl, K. Park, J. Hwang, K. H. Lee, K. Lee, S. J. Lee, and Y. J. Kim, "Development of HIL systems for active brake control systems," in *Proc. SICE-ICASE Int. Joint Conf.*, Busan, Korea, Oct. 18–21, 2006.
30. S. Raman, N. Sivashankar, W. Milam, W. Stuart, and S. Nabi, "Design and implementation of HIL simulators for powertrain control system software development" in *Proc. 1999 Amer. Control Conf.*, San Diego, CA, USA, Jun. 2–4,1999.
31. A. Cebi, L. Guvenc, M. Demirci, C. Karadeniz, K. Kanar, and E. Guraslan, "A low cost, portable engine electronic control unit hardware-in-the-loop test system" in *Proc. IEEE Int. Symp. on Ind. Electron.*, Dubrovnik, Croatia, Jun. 20–23, 2005.

Solutions to Practice Questions

1. The answer is b).

1.
2. The answer is a).

1. Step 1) Define system requirements.

   Step 2) Define system architecture and hardware requirements.

   Step 3) Define software requirements.

   Step 4) Define memory type and size.

   Step 5) Determine the microcontroller core requirements.

   Step 6) Define power, temperature, and reliability requirements.

   Step 7) Search for the right microcontroller based on developed requirements, availability, and longevity.

   Step 8) Study the microcontroller evaluation kits and design resource availability.

   Step 9) Evaluate the software tool chain.

   1. The answer is b).
   2. The answer is a).
   3. The answer is b).
   4. Using two constant blocks of 1 and 0 with an additional toggle switch block to form the following Simulink diagram, the user can turn the LED DL3 on and off by toggling the switch.
   5. The answer is b).
   6. The answer is a).
   1. The answer is a).
   2. These two approaches are a) Arduino IDE software and b) MATLAB and Simulink software.
   3. The answer is b).
   4. The answer is b).
   1. The answer is b).

This page titled 7.6: Chapter References is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8: MINI-SEGWAY SYSTEM — MECHATRONIC SYSTEM EXAMPLE I

Learning Objectives

1. Understand the mechatronic system modeling method through the mini-Segway modeling process and its implementation in Simulink.
2. Describe the dual-PID control architecture, along with its working principles, for the mini-Segway system, and the associated control tuning process.
3. Explain the conversion process from differential equations to a state-space model and model-based control design process for regulation tracking purpose.
4. Address how the sample and hold input method can be used to improve the closed-loop control system performance.
5. Introduce the simulation validation process for a dual-PID mini-Segway controller. Construct a simulation validation model in Simulink.
6. Discuss the simulation validation process for a model-based mini-Segway controller, especially the difference between utilizing linearized and original nonlinear models.
7. Describe the architecture of the mini-Segway system from both hardware and software points of view. Also describe the process of experimentally validating the dual-PID control strategy.
8. Provide a way to optimize the model-based controller experimentally.

This chapter provides an introduction to mini-Segway system modeling. It describes both model-free and model-based pendulum position regulation as well as speed tracking with pendulum position stabilization. The model-based control minimizes a quadratic cost function with added tracking error state in the state-space design model. The chapter also incorporates simulation and experimental validations.

8.0: Prelude

8.1: Mini-Segway System Modeling

8.2: Model-Free Dual-PID Mini-Segway Control System Architecture

8.3: Model-Based Mini-Segway Control

8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input

8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller

8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers

8.7: Experimental Study of the Mini-Segway Model-Free Control

8.8: Model-Based Control Experiments

8.9: Chapter References

This page titled 8: Mini-Segway System — Mechatronic System Example I is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8.0: Prelude

Chapters 2 through 7 addressed technologies used in mechatronic system design, control, and optimization. These include mechatronic system modelling and control, signal processing in both continuous- and discrete-time domains, mechatronic system actuators and sensors, and the control system architecture from both a hardware and software point of view. This chapter now introduces the first example of mechatronic systems—a mini-Segway system—from its modelling to control to simulation studies to experimental studies. Figure 8.1 shows how this chapter's eight sections are structured. Section 8.1 presents a mini-Segway mathematical model with both motor and sensor dynamics. Then Sections 8.2, 8.3, and 8.4 present three control methodologies for the mini-Segway. These are, respectively, model-free control, model-based control with zero-order sample and hold input, and model-based control with special sample and hold input for improved system performance. Next, these control methods are validated in software-in-the-loop (SiL) simulation studies (Sections 8.5 and 8.6). Finally, Sections 8.7 and 8.8 validate both model-free and model-based control methodologies experimentally. Model-based control validation focuses on special sample and hold input method discussed in Section 8.4.

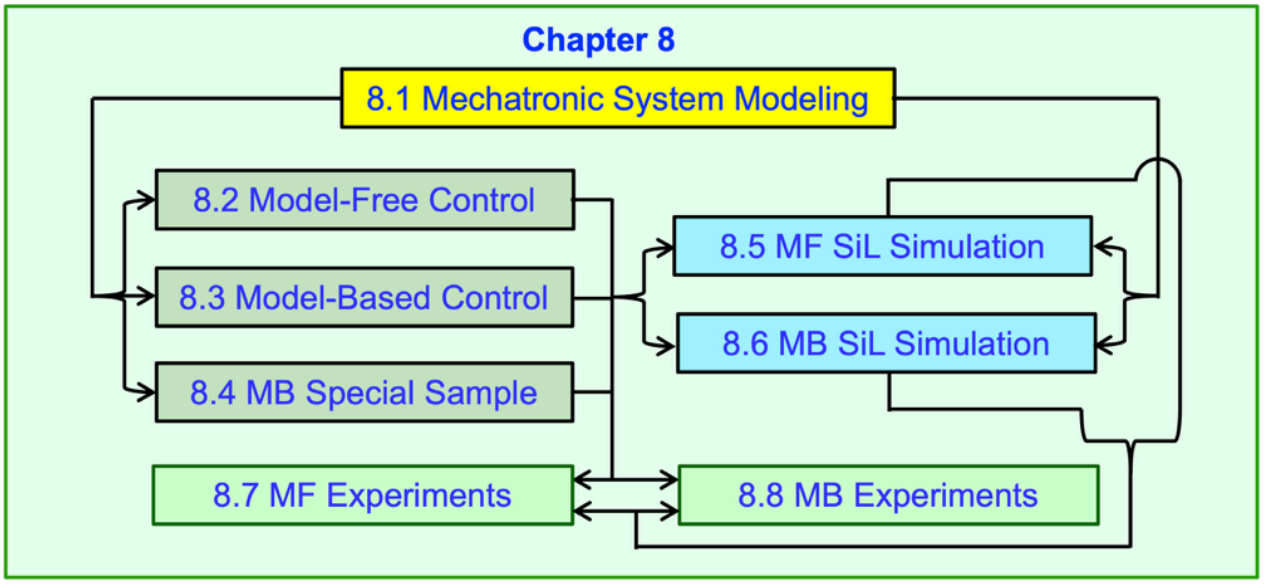


Figure 8.1: Organization of Chapter 8 sections.

Section 8.1 models a custom-built mini-Segway system, using the modeling techniques from Chapter 2. Mechanically, the mini-Segway is an inverted-pendulum on a cart [1], [2] with a relatively heavy pendulum. Therefore, in many ways, the mini-Segway can be modeled in similar fashion. This section models both motor electric characteristics and hall-effect sensor signals in detail. The mechanical dynamics of the mini-Segway system are based on Newton's law. The nonlinear mini-Segway model is implemented in MATLAB and Simulink with the Coulomb friction model. Closed-loop dual-PID (proportional-integral- derivative) [4] control of the mini-Segway, a model-free (MF) approach, is conducted in simulation studies in Section 8.5 utilizing the developed strategy, and in experimental studies in Section 8.7.

With the model developed, Section 8.2 proposes a control architecture, using two independent PID controllers [3], [4], [5]. One PID controller stabilizes the mini-Segway pendulum position, maintaining it at an upright position. The other controls the mini-Segway forward and backward motions in cart x direction. The coupling dynamics of pendulum angular displacement, and forward/backward and turning motions, are not considered (see Problems P8.2 and P8.4). To ensure that the integrators of the two PID controllers do not fight each other, which could lead to unbounded integration values, the integration gain of the PID controller for stabilizing the mini-Segway pendulum at the upright position is set to zero. The effect of this action is moderate due to the slow action of the integration operation.

Section 8.3 builds on Section 3.5 to address the model-based control design process for the mini-Segway, modeled in Section 8.1. It utilizes LQR (linear quadratic regulation) [6], [7], [8], [9] and LQTI (linear quadratic tracking with integral) [10], [11]. To apply model-based control design methodologies, we develop a linearized state-space mini-Segway model with respect to its normal operation condition. This model's parameters are then populated using the values in Table 8.1. Next, we design an LQR regulator, a model-based (MB) controller, for stabilizing the mini-Segway pendulum to keep the mini-Segway pendulum at its upright position.

Last, we add an error integral state of mini-Segway cart speed to the state-space model [10]. An LQTI controller is designed to track the mini-Segway cart speed with its pendulum position regulated at its upright position. The LQTI control strategy is a model-based one comparable with the dual-PID strategy discussed in Section 8.2, where the mini-Segway turning is controlled in an open loop.

Section 8.4 discusses control design for a discrete-time controller with a sample and hold input different from a conventional zero-order sample and hold. The square-pulse sample and hold input system is introduced, along with its state-space modeling methods. The mini-Segway system is non-minimal phase (NMP). At least one zero is located in the open right-half complex plan, which limits the achievable performance [12] by a closed-loop control strategy. By utilizing the special sample and hold input, a non-minimal phase continuous-time plan can be transferred into a minimal phase (MP) discrete-time one. This section demonstrates this phenomenon using our mini-Segway model [13]. It then proceeds to design a stabilizing controller.

Section 8.5 develops and validates the dual-PID control system model in Simulink. It uses the linearized mini-Segway six-state state-space model (see Section 8.3) with detailed PID gain selection to achieve the desired performance (pendulum stabilization and cart driving). First, the dual-PID control strategy is tuned and validated for pendulum stabilization with the linearized model. Next, both PID controllers are turned together and validated for simultaneous pendulum stabilizing and cart driving with the model developed in Section 8.3, with additional tuning of the dual-PID control gains for improved performance. Last, simulation studies are conducted for all three operational modes including pendulum stabilization, straight driving, and turning.

Section 8.6 conducts a model-based simulation utilizing the controller designed in Section 8.3. First, we investigate the LQR control strategy for state-feedback stabilization (factoring in the mini-Segway's cart position and speed, and pendulum position and speed). Next, we study state-feedback with the cart speed's integral tracking control. We factor in not only the mini-Segway's cart position and speed, and pendulum position and speed states, but also the tracking error state calculated internally in the LQTI controller. Then, we investigate the model-based state-feedback LQTI controller from Section 8.3 in simulation with the nonlinear mini-Segway model from Section 8.1, with the Coulomb friction model added. For all simulation studies, the feedback controller is sampled at 100 Hz.

Section 8.7 uses experiments to validate the dual-PID mini-Segway control strategy. It discusses the mini-Segway system hardware architecture, along with the sensors and actuators used to respectively generate feedback signals and drive the mini-Segway. Then we introduce the dual-PID control software in MATLAB/Simulink, along with the wheel speed signal processing and motor driving software. Recall that we calculated the pendulum angle through sensor fusion by utilizing both accelerometer and rate-gyro signals from MPU 6050. We also calculated the wheel speed based upon the hall-effect sensors mounted on the wheel axes through two gearboxes. Next, we introduce the overall software architecture in Simulink. The Simulink model runs within the host computer. The Arduino MKR 1000 Wi-Fi microcontroller collects the sensor signals; it executes the motor drive commands on the mini-Segway. Here, communication between the host computer and the Arduino microcontroller is through the Wi-Fi at a frequency of 100 Hz. Last, we conduct an experiment of the mini-Segway to validate the dual-PID control strategy for both pendulum stabilization and driving.

Section 8.8 addresses two experimental studies of the model-based dual-loop control strategy. One uses a standard inner-loop zero-order sample and hold input. The other uses an outer-loop square-pulse sample and hold input [13], [31]. The inner-loop model-based continuous-time LQR control stabilizes the pendulum by optimizing LQR input weight R_c (with fixed state weight Q_c) to achieve the best performance experimentally. The outer-loop discrete-time state-feedback LQR controller is designed for the mini-Segway system. Its discrete-time LQR controller is based upon the discrete-time model with square-pulse sample and hold input.

Orienting Questions

1. How do we model a mechatronic system step-by-step?
2. How can two independent PID controllers in the dual-PID control strategy be used to control coupled mini-Segway pendulum position and cart speed dynamics?
3. How do we design a model-based controller for a mechatronic system with tracking capability?
4. How does special sample and hold input work with model-based controllers?
5. How do we validate the model-based control strategy in an SiL environment?
6. How can we implement the dual-PID mini-Segway control strategy in real-time Simulink?

This page titled 8.0: Prelude is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8.1: Mini-Segway System Modeling

Learning Objectives

After completing this section, you will be able to:

1. Follow the step-by-step modeling process for the mini-Segway by utilizing Newton and Kirchhoff's laws.
2. Implement the developed differential equations in MATLAB and Simulink for future software-in-the-loop simulation (SiL) studies.

This section addresses the modeling process of the target mini-Segway. As shown in Figure 8.2, the battery is located on the top of the inverted pendulum. Mechatronic system electronics are located in the middle, including an MPU 6050 three-axis acceleration and rate gyro sensor board, the MSU interface board, an Arduino MKR 1000 Wi-Fi board, and an MKR motor carrier board. Two DC motors with their associated hall-effect speed sensors are mounted at the bottom. The mini-USB connector on the Wi-Fi board is extended to the top of the inverted pendulum for easy accessing and connection reliability. The mini-USB connector on the Wi-Fi board can be easily ripped off the board, With the full-size USB connector on the top of the inverted pendulum, the USB connector is secured, even with a large pulling force.



Figure 8.2: The mini-Segway mechatronic system.

The mini-Segway (MS) system shown in Figure 8.2 is a two-wheel self-balanced robot. It utilizes a low-cost microcontroller board (Arduino MKR 1000 Wi-Fi), two motor-wheel sets, and a cart body equipped with battery and mini-Segway electronics. Figure 8.3 shows a free body diagram of this system. The left and right motors provide the torques, τ_L and τ_R, respectively. They prevent the cart body (pendulum) from falling and maintain at the upright position, move the cart forward and backward, and turn to left and right. In addition, x_{cart}, y_{cart}, and z_{cart} are the cart displacements in its body coordinate with x_{cart}, y_{cart}, and z_{cart} marked on Figure 8.2, respectively; x_{abs}, y_{abs}, and y_{abs} are the cart absolute displacements in associated directions, respectively; θ_L and θ_R are the left and right wheel rotational angles, respectively; and θ_p and ϕ are the cart body pitch angle and cart yaw angle, respectively. In addition, m_p, l, and h are the total mass of the cart body (pendulum), distance from center of mass to the wheel axis, and the half distance between the two wheels, respectively. Last, x_p is the forward displacement of the cart body at the center of pendulum mass in the direction of x_{cart}.

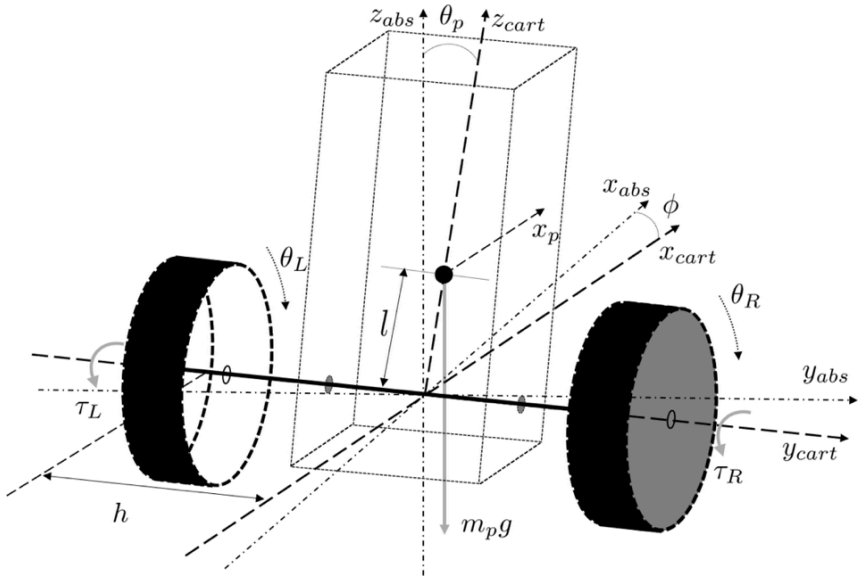


Figure 8.3: The overall mini-Segway diagram and its coordinate definitions [13].

With all parameters and variables in Figure 8.3 defined, let's proceed with a step-by-step modeling process.

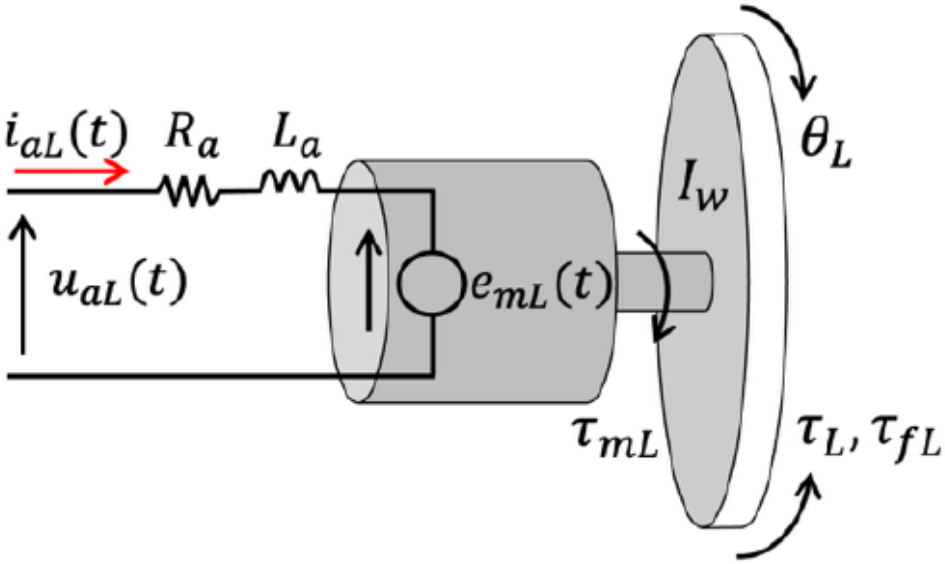


Figure 8.4: The left DC motor and wheel assembly.

Figure 8.4 shows the electric circuit of the left DC motor. It also shows the free-body diagram of the left DC motor's mechanical part with wheel attached. Here, u_{aL} and i_{aL} are the motor's armature voltage and current, respectively; and R_a and L_a are the motor's armature winding resistance and inductance, respectively. In addition, τ_{mL}, τ_L, and τ_{fL} are the motor's output torque, the load torque applied to the left wheel due to vehicle motion, and the left motor axis and bearing and gearbox friction torque; respectively. Also, I_w is the combined moment of inertia of the left wheel, wheel axis, gear box, and motor rotor; and θ_L and e_{mL} are the rotational angular displacement of the left wheel and the electromotive field voltage of the left motor, respectively.

When armature voltage e_{aL} is applied to the left DC motor, it generates armature current i_{aL}. The torque generated by the left motor and the associated electromotive field voltage are shown below.

$$\tau_{mL} = K_\tau i_{aL},\ e_{mL} = K_m \omega_L = K_m \dot{\theta}_L \tag{8.1.1}$$

where K_τ and K_m are the motor torque and electromotive field coefficients, respectively, defined in Section 5.2. Applying Kirchhoff's voltage law to the DC-motor electric circuit in Figure 8.4 results in the following equation.

$$u_{aL} - R_a i_{aL} - L_a \frac{di_{aL}}{dt} - e_{mL} = 0 \tag{8.1.2}$$

Assuming that $L_a = 0$ since normally $R_a \gg L_a$ leads to the following.

$$i_{aL} = \frac{u_{aL}}{R_a} - \frac{K_m}{R_a}\omega_L = \frac{u_{aL}}{R_a} - \frac{K_m}{R_a}\dot{\theta}_L \tag{8.1.3}$$

Now, let's consider the mechanical part of modeling. Applying Newton's law to the left DC-motor rotor and wheel assembly in Figure 8.4 yields the following equation.

$$I_w \frac{d\omega_L}{dt} = I_w \frac{d\dot{\theta}_L}{dt} = \tau_{mL} - b\omega_L - \tau_L \tag{8.1.4}$$

where $b\omega_L = \tau_{fL}$ is the axis friction torque and coefficient b is the rotational friction coefficient. Substituting equations 8.1.1 and 8.1.3 into 8.1.4 yields the following.

$$\frac{d\omega_L}{dt} = \frac{K_\tau}{I_\omega}\left(\frac{u_{aL}}{R_a} - \frac{K_m}{R_a}\omega_L\right) - \frac{b}{I_\omega}\omega_L - \frac{\tau_L}{I_\omega} \tag{8.1.5}$$

Combining equations 8.1.4 and 8.1.5 leads to the following motor torque function of input armature voltage u_{aL} and motor speed $\omega_L = \dot{\theta}_L$.

$$\tau_{mL} = \frac{-K_\tau K_m}{R_a}\omega_L + \frac{K_\tau}{R_a}u_{aL} = -\frac{K_\tau K_m}{R_a}\dot{\theta}_L + \frac{K_\tau}{R_a}u_{aL} \tag{8.1.6}$$

We are ready to model the right wheel assembly dynamics. Assume the right motor and wheel have the same parameters as the left. We have the following equations for the right motor and wheel assembly.

$$I_w \frac{d\omega_R}{dt} = I_w \frac{d\dot{\theta}_R}{dt} = \tau_{mR} - b\omega_R - \tau_R \tag{8.1.7}$$

$$\tau_{mR} = \frac{-K_\tau K_m}{R_a}\omega_R + \frac{K_\tau}{R_a}u_{aR} \quad = -\frac{K_\tau K_m}{R_a}\dot{\theta}_R + \frac{K_\tau}{R_a}u_{aR} \tag{8.1.8}$$

where the subscript R is used for the right motor model above. Consider the motion of the wheel moving forward. In this case, motor torque τ_{mL} (τ_{mR}) drives wheel and cart assembly, indicated by τ_L (τ_R), against its friction torque, $\tau_{fL} = b\dot{\theta}_L = b\omega_L$ (τ_{fR}), due to bearing and gear frictions. This yields the following equation.

$$\tau_L = \tau_{mL} - \tau_{fL} = \tau_{mL} - b\dot{\theta}_L = \frac{-K_\tau K_m}{R_a}\dot{\theta}_L + \frac{K_\tau}{R_a}u_{aL} - b\dot{\theta}_L \tag{8.1.9}$$

For the right motor wheel assembly, we have the following. Assume the motor and wheel parameters are the same.

$$\tau_R = \tau_{mR} - \tau_{fR} = \tau_{mR} - b\dot{\theta}_R = \frac{-K_\tau K_m}{R_a}\dot{\theta}_R + \frac{K_\tau}{R_a}u_{aR} - b\dot{\theta}_R \tag{8.1.10}$$

where u_{aR} and θ_R are the armature voltage and angular displacement of the right motor, respectively. Summing all the moments around the contacting point between the wheel and the ground for the left wheel (see the red dot in Figure 8.5) yields the following.

$$\tau_L - f_{axL} r_w = \tau_{mL} - b\dot{\theta}_L - f_{axL} r_w \tag{8.1.11}$$

where r_w is the effective wheel radius, and f_{axL} and f_{azL} are the x_{cart} and z_{abs} components of the reaction force generated by the mini-Segway pendulum.

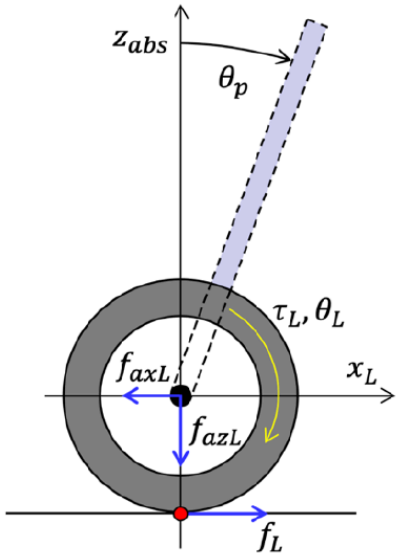


Figure 8.5: The left DC motor and wheel assembly

Combining equations 8.1.10 and 8.1.11 and applying Newton's law leads to the following equation.

$$\left(I_\omega + r_w^2 m_\omega\right)\ddot{\theta}_L = \frac{-K_\tau K_m}{R_a}\dot{\theta}_L + \frac{K_\tau}{R_a}u_{aL} - b\dot{\theta}_L - f_{axL}r_w = \tau_{mL} - b\dot{\theta}_L - f_{axL}r_w \tag{8.1.12}$$

where m_ω is the total wheel mass (rotational part only) including the DC-motor rotor mass. The following equation can be obtained using the same process for the right wheel, assuming all parameters are the same as the left ones.

$$\left(I_\omega + r_w^2 m_\omega\right)\ddot{\theta}_R = \frac{-K_\tau K_m}{R_a}\dot{\theta}_R + \frac{K_\tau}{R_a}u_{aR} - b\dot{\theta}_R - f_{axR}r_w = \tau_{mR} - b\dot{\theta}_R - f_{axR}r_w \tag{8.1.13}$$

Next, let's calculate f_{axL} and f_{axR} as functions of the motor armature voltages (u_{aL} and u_{aR}) and wheel angular speed and acceleration ($\dot{\theta}_L$, $\ddot{\theta}_L$, $\dot{\theta}_R$, and $\ddot{\theta}_R$), respectively. Based upon equations 8.1.12 and 8.1.13 we have the following equations.

$$f_{axL} = \frac{-K_\tau K_m}{R_a r_w}\dot{\theta}_L + \frac{K_\tau}{R_a r_w}u_{aL} - \frac{b\dot{\theta}_L}{r_w} - \frac{I_\omega \ddot{\theta}_L}{r_w} - m_\omega r_w \ddot{\theta}_L \tag{8.1.14}$$

$$f_{axR} = \frac{-K_\tau K_m}{R_a r_w}\dot{\theta}_R + \frac{K_\tau}{R_a r_w}u_{aR} - \frac{b\dot{\theta}_R}{r_w} - \frac{I_\omega \ddot{\theta}_R}{r_w} - m_\omega r_w \ddot{\theta}_R \tag{8.1.15}$$

Assume there is no slip between wheel and ground. We have $x_L = \theta_L r_w$ and $x_R = \theta_R r_w$, which yield the following.

$$\dot{x}_L = \dot{\theta}_L r_w, \ddot{x}_L = \ddot{\theta}_L r_w, \dot{x}_R = \dot{\theta}_R r_w, \ddot{x}_R = \ddot{\theta}_R r_w \tag{8.1.16}$$

Now, equations 8.1.14 and 8.1.15 can be rewritten as a function of left and right wheel axis linear displacements, x_L and x_R, as below.

$$f_{axL} = \frac{-K_\tau K_m}{R_a r_w^2}\dot{x}_L + \frac{K_\tau}{R_a r_w}u_{aL} - \frac{b\dot{x}_L}{r_w^2} - \frac{I_\omega \ddot{x}_L}{r_w^2} - m_\omega \ddot{x}_L \tag{8.1.17}$$

$$f_{axR} = \frac{-K_\tau K_m}{R_a r_w^2}\dot{x}_R + \frac{K_\tau}{R_a r_w}u_{aR} - \frac{b\dot{x}_R}{r_w^2} - \frac{I_\omega \ddot{x}_R}{r_w^2} - m_\omega \ddot{x}_R \tag{8.1.18}$$

We have now found both axis forces, f_{axL} and f_{axR}, in equations 8.1.17 and 8.1.18, and applied them to the mini-Segway axis (Figure 8.6). Our next step is to find the equations of motion for the mini-Segway pendulum with x_p and θ_p representing the displacement at the pendulum center of mass location and the anticlockwise rotational angle, respectively.

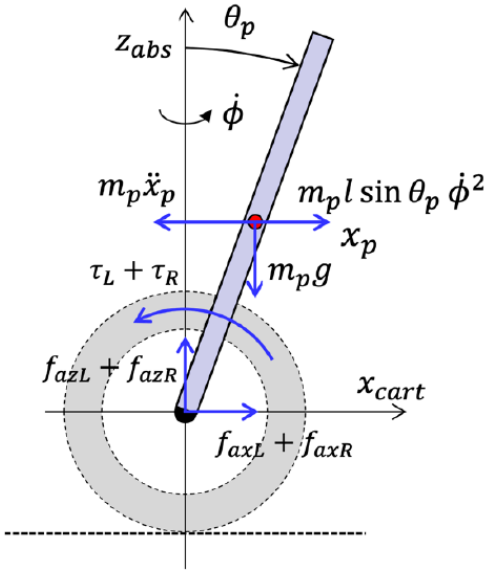


Figure 8.6: The mini-Segway pendulum free-body diagram

Assuming that the pendulum mass is evenly distributed to simplify the modeling process. We can now write the position of pendulum center of mass (rectangular bar in Figure 8.6) as a function of wheel axis displacement $x_{cart} = (x_L + x_R)/2$ at the center of the two wheels.

$$x_p = x_{cart} + l\sin\theta_p \tag{8.1.19}$$

and its first- and second-order time derivatives are shown below.

$$\begin{aligned} \dot{x}_p &= \dot{x}_{cart} + l\dot{\theta}_p\cos\theta_p \\ \ddot{x}_p &= \ddot{x}_{cart} - l\dot{\theta}_p^2\sin\theta_p + l\ddot{\theta}_p\cos\theta_p \end{aligned} \tag{8.1.20}$$

where l is the equivalent half length of the mini-Segway pendulum. By Newton's law, we can obtain the linear equation of motion for the pendulum.

$$m_p\ddot{x}_p = m_p\left(\ddot{x}_{cart} - l\dot{\theta}_p^2\sin\theta_p + l\ddot{\theta}_p\cos\theta_p\right) = f_{axL} + f_{axR} \; + m_p l\sin\theta_p\dot{\phi}^2 \tag{8.1.21}$$

Substituting equations 8.1.17, 8.1.18, 8.1.19, and 8.1.20 into equation 8.1.21 yields the first equation of motion in x_{cart} direction below.

$$m_p\left(\ddot{x}_{cart} - l(\dot{\theta}_p^2 + \dot{\phi}^2)\;\sin\theta_p + l\ddot{\theta}_p\cos\theta_p\right) = \frac{-2K_\tau K_m}{R_a r_w^2}\dot{x}_{cart} + \frac{K_\tau}{R_a r_w}(u_{aL} + u_{aR}) - \frac{2b\dot{x}_{cart}}{r_w^2} - 2\left(\frac{I_\omega}{r_w^2} + m_\omega\right)\ddot{x}_{cart} \tag{8.1.22}$$

We can obtain the angular motion θ_p of pendulum position (the second equation of motion) by applying Newton's rotational motion law at the joint of pendulum and wheel below (see the red dot in Figure 8.6).

$$\left(I_p + m_p l^2\right)\ddot{\theta}_p = -m_p\ddot{x}_p l\;\cos\theta_p + m_p l^2\sin\theta_p\cos\theta_p\dot{\phi}^2 + m_p g l\sin\theta_p - (\tau_L + \tau_R) \tag{8.1.23}$$

where τ_L and τ_R are defined in equations 8.1.8 and 8.1.9; term $-m_p\ddot{x}_p l\cos\theta_p$ is due to the linear acceleration at the pendulum center of mass; term $m_p l^2\sin\theta_p\cos\theta_p\dot{\phi}^2$ is the centrifugal force caused by the rotational motion of the mini-Segway in z-direction; and $\dot{\phi}$ is the rate of rotation defined in both Figure 8.6 and Figure 8.7. Substituting equations 8.1.8 and 8.1.9 into 8.1.23, and using the relations $x_p = x_{cart} + l\sin\theta_p$, $\ddot{x}_p = \ddot{x}_{cart} + \ddot{\theta}_p l\cos\theta_p - \dot{\theta}_p^2 l\sin\theta_p$, and $\left(\dot{\theta}_L + \dot{\theta}_R\right)R_w = 2\dot{x}_{cart}$; lead to the equation with state variables in terms of θ_p, x_{cart}, and ϕ below.

$$m_p l\cos\theta_p\ddot{x}_{cart} - 2\left(\frac{K_\tau K_m}{R_a r_w} + \frac{b}{r_w}\right)\dot{x}_{cart} + \left(I_p + m_p l^2\left(1 + \cos^2\theta_p\right)\right)\ddot{\theta}_p - m_p g l\sin\theta_p - m_p l^2\sin\theta_p\cos\theta_p\left(\dot{\phi}^2 + \dot{\theta}_p^2\right) = -\frac{K_\tau}{R_a}(u_{aL} \cdot$$

Now, we are ready to derive the third equation of motion (turning motion) based on Figure 8.7. Here, the main mass is contributed by the pendulum with two motors. Thus, the wheel mass m_w is ignored so that $f_L = f_{axL}$ and $f_R = f_{axR}$.

The mini-Segway turning motion can be decomposed to rotational motion and translational motion on the x_{cart} and y_{cart} axes (see Figure 8.3). The cart rotational motion of the mini-Segway induces rotational motion of pendulum around z_{abs} axis (Figure 8.3). Next, we will derive the rotational equation of motion about the z_{abs} axis. The rotation of the mini-Segway resulted from different driving forces on the left and right wheels, f_{axL} and f_{axR}. It can be modeled as below by ignoring the contact friction force between the wheel and road during turning.

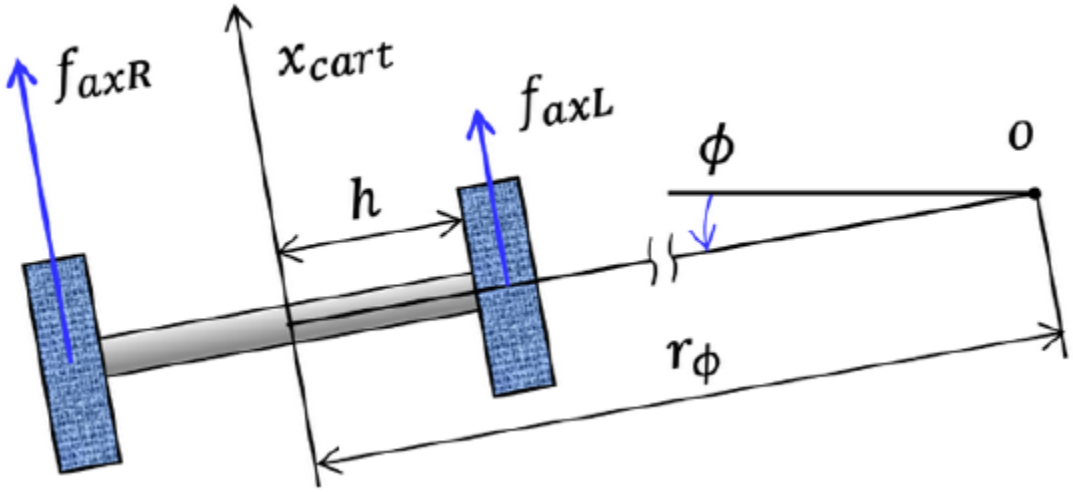


Figure 8.7: The mini-Segway turning free body diagram

$$\ddot{\phi} I_{sz} = (f_{axL} - f_{axR})\, h \tag{8.1.25}$$

where I_{sz} is the moment of inertia of the mini-Segway relative to z_{abs} axis. To calculate I_{sz}, we assume that the mass of pendulum is uniformly distributed along its length $2l$; wheel moment of inertia in z_{abs} direction is I_{wz} with an offset of $h_z = h$; and $I_z = m_p h_p^2/3$ (h_p is the effective width of pendulum) is the pendulum moment of inertia in z_{abs} direction when $\theta_p = 0$. Then I_{p_z} can be calculated below.

$$I_{pz} = 2\left(I_{wz} + m_\omega h_z^2\right) + I_z + \int_0^{2l} \frac{m_p}{l}(y \sin\theta_p)^2 dy = 2(I_{wz} + m_\omega h_z^2) + \frac{\left(h_p^2 + 8l^2 \sin^2\theta_p\right) m_p}{3} \tag{8.1.26}$$

where $I_{wz} + m_\omega h_z^2$ is the wheel moment of inertia in z_{abs} direction shifted by distance h_z. Substituting equations 8.1.17 and 8.1.18 into 8.1.25 yields the following.

$$\ddot{\phi} I_{p_z} = -\frac{(K_\tau K_m + R_a b)\, h}{R_a r_w^2}(\dot{x}_L - \dot{x}_R) - \left(\frac{I_\omega h}{r_w^2} + m_\omega h\right)(\ddot{x}_L - \ddot{x}_R) + \frac{K_\tau h}{R_a r_w}(u_{aL} - u_{aR}) \tag{8.1.27}$$

Turning the mini-Segway is realized by the angular velocity difference of left and right wheels (Figure 8.7). Assuming that there is no slip between the tire and road in the x_{cart} direction, the following relationships can be obtained.

$$\begin{cases} x_L = x + \phi h \\ x_R = x - \phi h \end{cases} ; \begin{cases} \dot{x}_L = \dot{x} + \dot{\phi} h \\ \dot{x}_R = \dot{x} - \dot{\phi} h \end{cases} ; \begin{cases} \ddot{x}_L = \ddot{x} + \ddot{\phi} h \\ \ddot{x}_R = \ddot{x} - \ddot{\phi} h \end{cases} \tag{8.1.28}$$

Substituting equation 8.1.28 into 8.1.27, we obtain the following equation of motion (the third EOM) describing rotation (yaw) of pendulum.

$$\ddot{\phi}\left(I_{sz} + 2\left(\frac{I_\omega}{r_w^2} + m_\omega\right) h^2\right) + 2\frac{(K_\tau K_m + R_a b)\, h^2}{R_a r_w^2}\dot{\phi} = \frac{K_\tau h}{R_a r_w}(u_{aL} - u_{aR}) \tag{8.1.29}$$

We've now developed three equations of motion 8.1.22, 8.1.23 and 8.1.29 for the mini-Segway mechatronic system. We can reorganize them in equation 8.1.30.

$$\left(m_p+2\left(\frac{I_\omega}{r_w^2}+m_\omega\right)\right)\ddot{x}_{cart}+2\left(\frac{K_\tau K_m}{R_a r_w^2}+\frac{b}{r_w^2}\right)\dot{x}_{cart}+m_p l\ddot{\theta}_p\cos\theta_p-m_{\ p}\ l(\dot{\theta}_p^2+\dot{\phi}^2)\sin\theta_p\ =\frac{K_\tau}{R_a r_w}(u_{aL}+u_{aR})$$

$$m_p l\cos\theta_p\ddot{x}_{cart}-2\left(\frac{K_\tau K_m}{R_a r_w}+\frac{b}{r_w}\right)\dot{x}_{cart}+\left(I_p+m_p l^2\left(1+\cos^2\theta_p\right)\right)\ddot{\theta}_p-m_p gl\sin\theta_p-m_p l^2\sin\theta_p\cos\theta_p\left(\dot{\phi}^2+\dot{\theta}_p^2\right)=-\frac{K_\tau}{R_a}(u_{aL}\cdot$$

$$\ddot{\phi}\left(I_{sz}+2\left(\frac{I_\omega}{r_w^2}+m_\omega\right)h^2\right)+2\frac{\left(K_\tau K_m+R_a b\right)h^2}{R_a r_w^2}\dot{\phi}=\frac{K_\tau h}{R_a r_w}\left(u_{aL}-u_{aR}\right)$$

With the nonlinear mini-Segway model developed in equation 8.1.30, our next step is to obtain a linearized model for the model-based control design that will be described in Sections 8.3 and 8.4. Consider the case of the nonlinear model 8.1.30 at the upright equilibrium position with the following equilibrium conditions.

$$\theta_p=0, x_{cart}=0, \dot{\phi}=0 \tag{8.1.31}$$

We can approximate these variables in equation 8.1.30 as follows or use the process of linearizing a nonlinear model described back in Section 3.6.

$$\cos\theta_p\approx 1, \sin\theta_p\approx\theta_p, \dot{\phi}^2+\dot{\theta}_p^2\approx 0 \tag{8.1.32}$$

Define these abbreviations as below.

$$c_1=m_p+\frac{2I_\omega}{r_w^2}+2m_\omega, c_2=I_p+2m_p l^2, c_3=I_{sz}+\frac{2I_\omega h^2}{r_w^2}+2m_\omega h^2 \tag{8.1.33}$$

Using these abbreviations leads to the following linearized model.

$$\begin{aligned}
\ddot{x}_{cart}&=-\frac{2}{c_1}\left(\frac{K_\tau K_m}{R_a r_w^2}+\frac{b}{r_w^2}\right)\dot{x}_{cart}-\frac{m_p l}{c_1}\ddot{\theta}_p+\frac{K_\tau}{c_1 R_a r_w}(u_{aL}+u_{aR})\\
\ddot{\theta}_p&=-\frac{m_p l}{c_2}\ddot{x}_{cart}+\frac{2}{c_2}\left(\frac{K_\tau K_m}{R_a r_w}+\frac{b}{r_w}\right)\dot{x}_{cart}+\frac{m_p gl}{c_2}\theta_p-\frac{K_\tau}{c_2 R_a}(u_{aL}+u_{aR})\\
\ddot{\phi}&=-2\frac{\left(K_\tau K_m+R_a b\right)h^2}{c_3 R_a r_w^2}\dot{\phi}+\frac{K_\tau h}{c_3 R_a r_w}\left(u_{aL}-u_{aR}\right)
\end{aligned} \tag{8.1.34}$$

Table 8.1: Mini-Segway parameter nomenclatures (names, values, and associated units).

Symbol	Description	Value (Unit)
τ_{mL} and τ_{mR}	Left and right DC-motor output torques.	(Nm)
i_{aL} and i_{aR}	Left and right DC-motor armature currents.	(A)
e_{mL} and e_{mR}	Left and right DC-motor EMF voltages.	(V)
K_m	DC-motor EMF constant (at wheel axis)	$0.027\ (V\cdot s/rad)$
K_τ	DC-motor torque constant (at wheel axis)	$0.23\ (Nm/A)$
u_{aL} and u_{aR}	Left and right DC-motor armature voltages.	(V)
R_a	DC-motor armature resistance	$4.8\ (\Omega)$
L_a	DC-motor armature inductance (assumed to be 0)	$\sim 0\ (H)$
b	DC-motor axis and gear-box friction coefficient	$\sim 0\ (Nm\cdot s/rad)$
τ_L and τ_R	Left and right applied wheel torques	(Nm)
$\dot{\theta}_L$ and $\dot{\theta}_R$	Left and right wheel speeds (ω_L and ω_R)	(rad/s)
θ_L and θ_R	Left and right wheel angular displacements	(rad)
f_L and f_R	Left and right wheel traction (driving) forces	(N)
I_w	Wheel axis moment of inertia	$4.09\times 10^{-5}(Kgm^2)$
I_p	Mini-Segway pendulum moment of inertia (θ_p)	$1.9\times 10^{-3}(Kgm^2)$

Symbol	Description	Value (Unit)
I_{sz} (I_{pz})	Mini-Segway pendulum moment of inertia (ϕ)	5×10^{-3} (Kgm^2)
l	Pendulum hinge to center of mass distance	$0.02(m)$
m_w	Wheel mass (including rotor/gear mass)	0.0492 (Kg)
m_p	Mini-Segway pendulum total mass	0.7197 (Kg)
f_{axL} and f_{axR}	Left and right horizonal hinge forces	(N)
r_w	Wheel effective radius	0.034 (m)
h	Half cart width $(h_z \approx h,\ h_p \approx 1.5h)$	0.086 (m)
x_{cart} (x)	Mini-Segway center of hinge displacement	(m)
θ_p	Mini-Segway pendulum tilt angle	(rad)
ϕ	Mini-Segway cart turning angle	(rad)

Note that the linearized model 8.1.34 has two control inputs, u_{aL} and u_{aR}, and six states, x_{cart}, $\dot{x}_{cart}$, θ_p, $\dot{\theta}_p$, ϕ, and $\dot{\phi}$. Table 8.1 summarizes parameter names, values, and their associated units. Figure 8.8 shows a Simulink implementation of the three-DOF nonlinear mini-Segway model described by equations 8.1.21, 8.1.23, and 8.1.25. The driving forces and torques are defined by equations 8.1.14 and 8.1.15, where the three double-integrators associated with equations 8.1.21, 8.1.23, and 8.1.25 represent the three degrees of freedom of the mini-Segway: x_{cart}, θ_p, and ϕ.

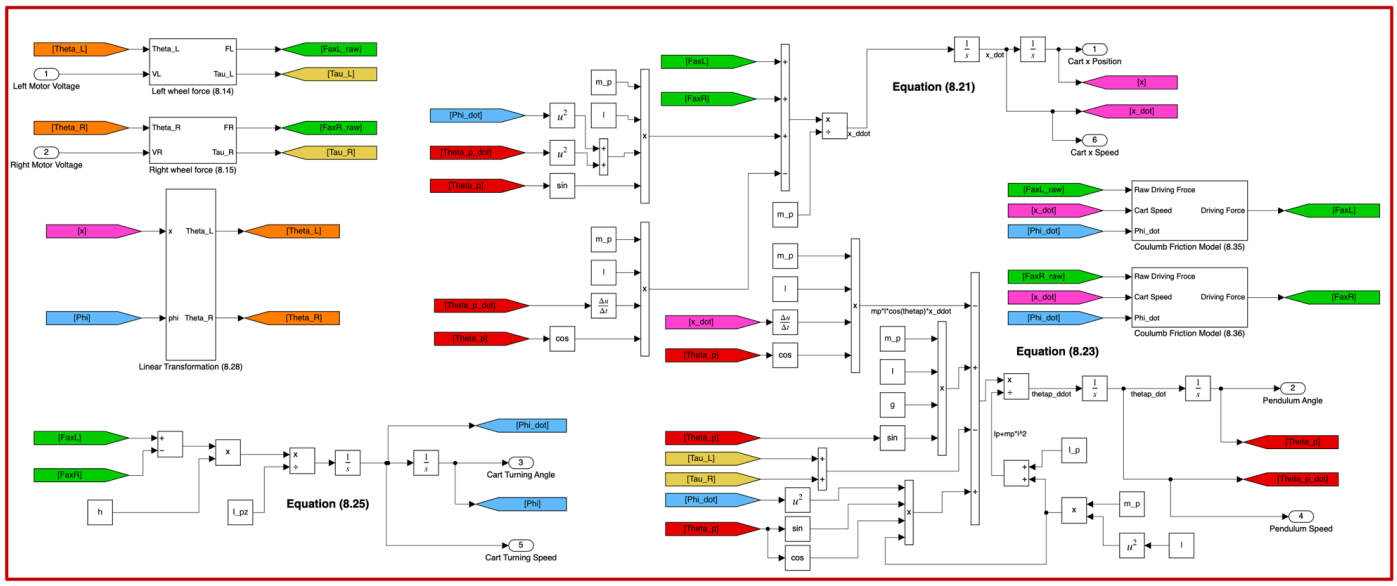


Figure 8.8: Simulink realization of developed mini-Segway model.

In order to simulate the nonlinear friction of the mini-Segway, a Coulomb friction model (see Figure 8.9) is developed to correct both the left and right wheel forces, f_L and f_R, simulating the minimal driving forces needed to move the mini-Segway.

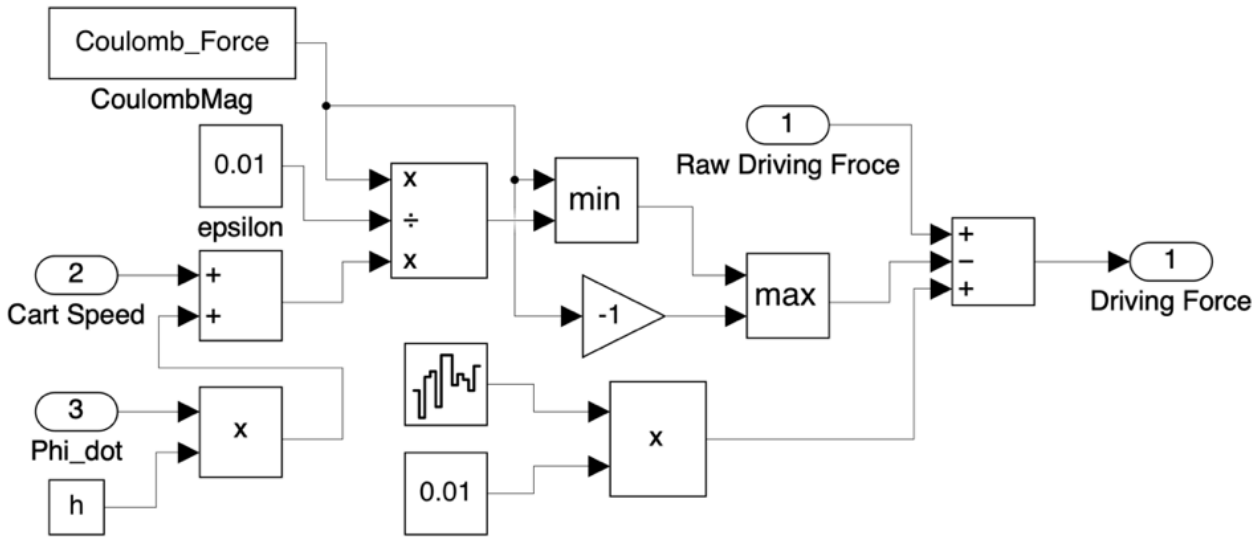


Figure 8.9: Simulink implementation of Coulomb friction on the left wheel.

The Coulomb friction is modeled using equation 8.1.35 for the left wheel force model.

$$f_{axL\_C} = \begin{cases} f_{axl} - sign\left(\dot{x}_L\right) f_{Coulomb} + \varepsilon_n, & if \ \ |\dot{x}_L f_{Coulomb}/\varepsilon| \geq f_{Coulomb} \\ f_{axl} - \dot{x}_L f_{Coulomb}/\varepsilon + \varepsilon_n, & else \end{cases} \tag{8.1.35}$$

where f_{axL_C} is the left wheel force f_{axL} with Coulomb friction; ε_n is the Coulomb friction variation noise enabled by the "Noise" block (see Figure 8.9) reduced by 100 times; and $\dot{x}_L = \dot{x}_{cart} + \dot{\phi} h$ is the left wheel linear speed.

Instead of using an exact sign function for modeling the Coulomb friction, we use a saturation function with magnitude of $f_{Coulomb}$. This reduces the system model's stiffness and makes numerical simulations feasible. For the right wheel, Coulomb force model follows the following equation.

$$f_{axR\_C} = \begin{cases} f_{axR} - sign\left(\dot{x}_R\right) f_{Coulomb} + \varepsilon_n, & if \ \ |\dot{x}_R f_{Coulomb}/\varepsilon| \geq f_{Coulomb} \\ f_{axR} - \dot{x}_R f_{Coulomb}/\varepsilon + \varepsilon_n, & else \end{cases} \tag{8.1.36}$$

where f_{axR_C} is the right wheel force f_{axR} with Coulomb friction and $\dot{x}_R = \dot{x}_R - \dot{\phi} h$ is the right wheel linear speed. The Simulink diagram of the right wheel Coulomb friction model is similar to the left one in Figure 8.9 and is omitted.

Section Summary

This section draws on techniques introduces in Chapter 2 to model a mechatronic system, in this case, a mini-Segway system. Mechanically, the mini-Segway can be considered as an inverted-pendulum on a cart [1], [2] with a relatively heavy pendulum. Therefore, it has a certain similarity to modeling an inverted pendulum on a cart except its turning motion. DC-motor electric characteristics are modeled in detail. The mechanical dynamics of the mini-Segway system are based upon Newton's law. Last, the nonlinear mini-Segway model is implemented in MATLAB and Simulink, with an added Coulomb friction model. The closed-loop dual-PID (proportional-integral- derivative) control of the inverted pendulum [4] is conducted in a simulation study in Section 8.5 and in an experimental study in Section 8.7.

Section Keywords

Practice Problems

1. What is the number of degrees of freedom (DOFs) for the mini-Segway? Select the correct answer.
   a. 2
   b. 3

   c. 4
2. From mini-Segway equation of motion 8.1.30, decide if the following statement is true or not. The mini-Segway turning motion ϕ does not affect the mini-Segway pendulum motion.
   a. True
   b. False
3. Is the following statement true or false? If both DC-motor armature voltages u_{aL} and u_{aR} in equation 8.1.30 are the same, mini-Segway turning motion ϕ will be damped out (going to zero).
   a. True
   b. False
4. Is the following statement true or false? Mini-Segway linear motion in x direction and angular pendulum motion θ_p are not coupled.
   a. True
   b. False

This page titled 8.1: Mini-Segway System Modeling is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8.2: Model-Free Dual-PID Mini-Segway Control System Architecture

Learning Objectives

After completing this section, you will be able to:

1. Describe the dual-PID control architecture of the mini-Segway.
2. Provide the advantage and disadvantage of the mini-Segway's dual-PID loop control.
3. Explain the reason for setting the integration gain of one of the dual PID controllers to zero.
4. Improve control performance by utilizing the anti-windup function for both continuous- and discrete-time PID controllers.

This section introduces the mini-Segway's dual-PID control. The detailed PID (proportional-integral-derivative) control technique and its control parameter tuning methods were addressed in Section 3.4 and reference [5]. Here, we will first review the PID controller in both continuous- and discrete-time forms. Let $e(t)$ be a continuous-time error signal (reference minus measured control target). Then, in the time domain, we can describe a continuous-time PID control law with $u_{PID}(t)$ as the PID control output.

$$u_{PID}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{8.2.1}$$

where K_p, K_i, and K_d are continuous-time proportional, integral, and derivative gains, respectively. Its Laplace transform is shown below.

$$U_{PID}(s) = \left(K_p + \frac{K_i}{s} + K_d s\right) E(s) = C_{PID}(s)\, E(s) \tag{8.2.2}$$

where $U_{PID}(s)$ and $E(s)$ are Laplace transforms of $u_{PID}(t)$ and $e(t)$, repectively; and $C_{PID}(s)$ is the PID controller transfer function. For details of the Laplace transform, see Section 2.1. A continuous-time PID controller cannot be efficiently implemented into a microcontroller because continuous-time integration and derivative calculations are inefficient. To implement the PID controller in discrete-time, we must use discrete-time integration and derivative operations to replace the continuous-time ones. The real-time Simulink software for the Arduino family microcontroller allows direct use of the continuous-time transfer function, where these transfer functions are converted into associated digital operations during the software compiling process. Practically, the derivative operation in the PID controller often amplifies the high-frequency components in the error signal normally from the measured feedback signal. This leads to noisy control output. Often, a low-pass filter in the form of $(T_d s + 1)^{-1}$is placed before the derivative operation shown below as a modified PID controller, where T_d is the time constant of the low-pass filter.

$$U_{PID}(s) = \left(K_p + \frac{K_i}{s} + \frac{K_d s}{T_d s + 1}\right) E(s) \tag{8.2.3}$$

The time constant T_d is selected so that the corner frequency of the low-pass filter, $\omega_c = T_d^{-1}$is at least five times the dynamic system bandwidth. Physical actuators have limitations such as maximum armature voltage for a DC motor corresponding to its maximum stalled torque and no-load speed (see Section 5.2). In this case, a saturation operation may be required to limit the PID control output to meet the limitation requirement. However, with the implementation of saturation on the PID control output, the integration windup phenomenon could occur, leading to slow PID control response [4], [14], [15]. The windup occurs when the PID controller output is saturated and the integrator continuously integrates the error, making the saturation worse. When the error signal sign is reversed, it takes a long time to get the integration output to reflect it. This causes a slow PID control response [4], [14], and [15]. To avoid integration windup of PID controllers, an anti-windup logic was developed in references [4] and [14]; it modified the PID controller described in equation 8.2.1 as follows, with l_{max} and l_{min} ($l_{max} > l_{min}$) as the saturation maximum and minimum limits, respectively.

$$u_{PID}(t) = K_p e(t) + g_i K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{8.2.4}$$

where g_i is a gain coefficient used to enable or disable the integration action. It is defined below.

$$g_i = \begin{cases} 0, & if\ u_{PID}(t) \geq l_{max}\ \&\ e(t) > 0\ or\ u_{PID}(t) \leq l_{min}\ \&\ e(t) < 0 \\ 1, & else \end{cases} \tag{8.2.5}$$

The numerical logic in equation 8.2.5 disables the integration operation (by setting $g_i = 0$) in two conditions. The first is when PID control output $u_{PID}(t)$ is greater than or equal to PID control output upper limit l_{max} with PID control input error $e(t)$ greater than zero (positive). The second is when PID control output $u_{PID}(t)$ is less than or equal to PID control output lower limit l_{min} with PID control input error $e(t)$ less than zero (negative). The intension is that if the PID control output $u_{PID}(t)$ is saturated from above, the integration action is enabled only when the PID input error $e(t)$ is negative. The control output $u_{PID}(t)$ can then be reduced. If the PID control output $u_{PID}(t)$ is saturated from below, the integration action is enabled only when the PID input error $e(t)$ is positive. The control output $u_{PID}(t)$ can then be increased. Finally, if the PID control output $u_{PID}(t)$ is between the lower saturation limit l_{min} and upper saturation limit l_{max}, the integration action is always enabled. Figure 8.10 shows a Simulink implementation of continuous-time PID control with anti-windup capability; all parameters are as defined before, assuming the integration gain K_i is positive.

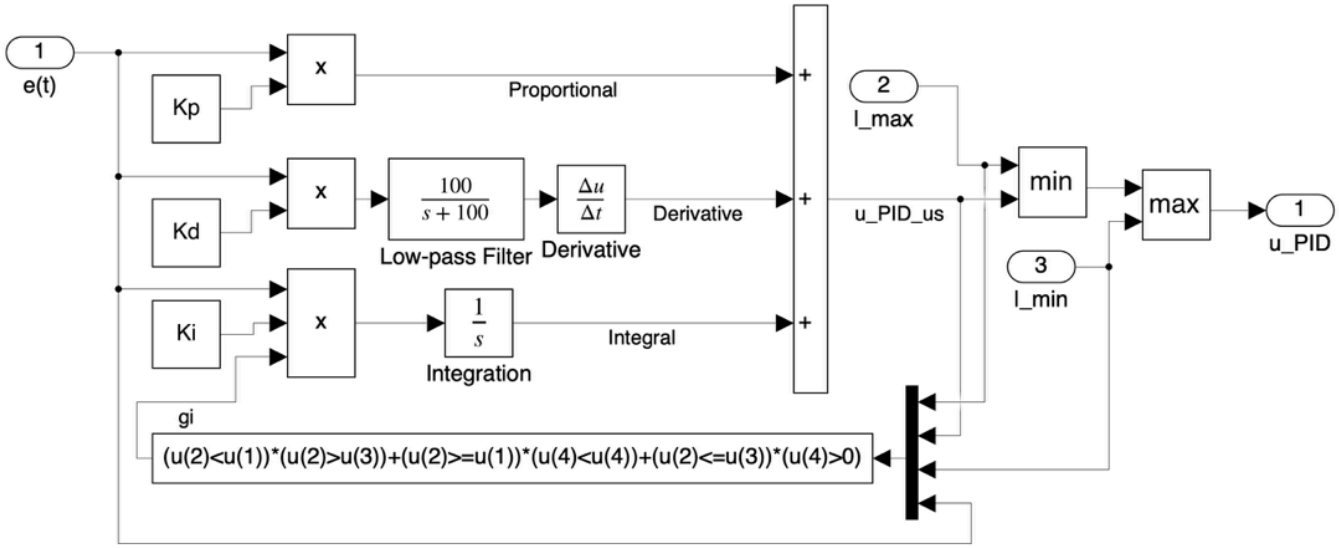


Figure 8.10: Continuous-time Simulink implementation of anti-windup PID controller.

Another form of the PID controller is defined in the discrete-time domain as below.

$$u_{PID}(k) = K_p e(k) + K_i T_s \sum_{n=0}^{k} e(k) + K_d \frac{e(k) - e(k-1)}{T_s} \tag{8.2.6}$$

where K_p, K_i, and K_d are discrete-time equivalent proportional, integral, and derivative gains, respectively; T_s is the sample period of the discrete-time PID controller; and $e(k)$ and $u_{PID}(k)$ are discrete-time error and PID control signals at sample step k, respectively. Then, in the discrete frequency domain (also called z-domain), we can describe a discrete-time PID control law 8.2.6 in the form of a z-transform, where $z = e^{sT_s}$ (see Section 4.4).

$$U_{PID}(z) = \left(K_p + \frac{K_i T_s}{1 - z^{-1}} + \frac{K_d}{T_s}(1 - z^{-1})\right) E(z) = C_{PID}(z)\, E(z) \tag{8.2.7}$$

where $U_{PID}(z)$ and $E(z)$ are z-transforms of $u_{PID}(k)$ and $e(k)$, respectively; $C_{PID}(z)$ is the discrete-time PID controller transfer function; $T_s(1 - z^{-1})^{-1}$ is the z-transform of the discrete-time integration; and $(1 - z^{-1})/T_s$ is the z-transform of the discrete-time time derivative (that is, the continuous-time derivative is replaced by the discrete-time difference operation). For the discrete-time transfer function, z^{-1} is a one-step delay operator; if inversed z-transform $\mathcal{Z}^{-1}[E(z)]$ equals to $e(k)$, then inversed z-transform of $\mathcal{Z}^{-1}[z^{-1}E(z)]$ is $e(k-1)$. Similarly, we can place a low-pass digital filter in the form of $(1 - e^{-T_s/T_d})(1 - e^{-T_s/T_d} z^{-1})^{-1}$ before the derivative operation shown below as a modified PID controller, where T_d is the time constant of the low-pass filter in a continuous-time domain.

$$U_{PID}(z) = \left(K_p + \frac{K_i T_s}{1 - z^{-1}} + \frac{K_d (1 - e^{-T_s/T_d})(1 - z^{-1})}{T_s (1 - e^{-T_s/T_d} z^{-1})} \right) E(z) \tag{8.2.8}$$

To avoid integration windup of PID controllers, an anti-windup logic similar to the continuous-time one can be employed, as below.

$$u_{PID}(t) = K_p e(k) + g_i K_i T_s \sum_{n=0}^{k} e(k) + K_d \frac{e(k) - e(k-1)}{T_s} \tag{8.2.9}$$

where g_i is a gain coefficient used to enable or disable the integration action and is defined below.

$$g_i = \begin{cases} 0, & if\ u_{PID}(k) \geq l_{max}\ \&\ e(k) > 0\ or\ u_{PID}(k) \leq l_{min}\ \&\ e(k) < 0 \\ 1, & else \end{cases} \tag{8.2.10}$$

Figure 8.11 shows a Simulink implementation of discrete-time PID control with anti-windup capability, with all parameters defined before except for $a = e^{-T_s/T_d}$, $d = (1 - a)/T_s$, and $K = 1$.

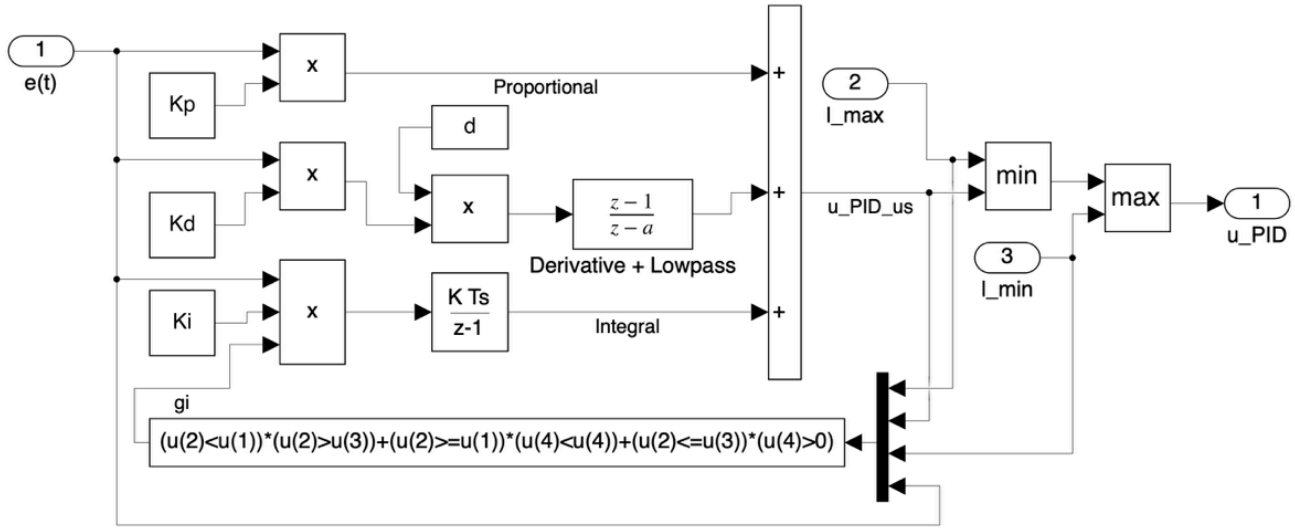


Figure 8.11: Discrete-time Simulink implementation of anti-windup PID controller.

Now, we are ready to discuss the dual-PID control architecture for the mini-Segway system. Two PID controllers are used to stabilize the mini-Segway pendulum and follow the given cart speed reference (forward and backward), respectively. The mini-Segway turning motion is controlled in an open-loop; see Figure 8.12 for its control system architecture. The mini-Segway turning motion can also be controlled in a closed-loop by using PID controllers to make two wheels (left and right) follow their reference speeds. In this case, the left and right wheel speeds are controlled independently. If the reference speeds for both wheels are the same, the mini-Segway will move forward or backward. If the left and right wheel reference speeds are different, the mini-Segway will turn. This strategy allows the mini-Segway turning while driving forward or backward. The three-wheel cart system discussed in Chapter 9 adopts this approach.

Figure 8.12 illustrates the entire architecture of the dual-PID control system. It shows all associated hardware including two DC motors, two wheel-speed encoders, and the MPU 6050 three-axis accelerometers and rate-gyros. The cart linear speed ($\dot{x}_{cart}$) is calculated by converting the hall-effect sensor pulse-trains of both left and right wheels into the numbers of counts for the left and right encoders over the given sample period T_s, then into two-wheel angular speed signals, and in turn into the cart speed signal. It multiplies the wheel effective radius by the averaged wheel speed, assuming that there is no wheel slip between wheel and road surface. (See Section 6.2 for the detailed speed calculation formula.) The pendulum angle (θ_p) is calculated based upon the sensor fusion technology discussed in Section 6.5; it utilizes both three-axis accelerometer and rate-gyro signals provided by the MPU 6050, a MEMS sensor. Both PID controller analog outputs are converted into PWM signals in the Arduino MKR motor carrier board. The two H-bridge DC motor drive circuits in the Arduino MKR motor carrier board are used to drive both left and right DC motors mounted on the mini-Segway.

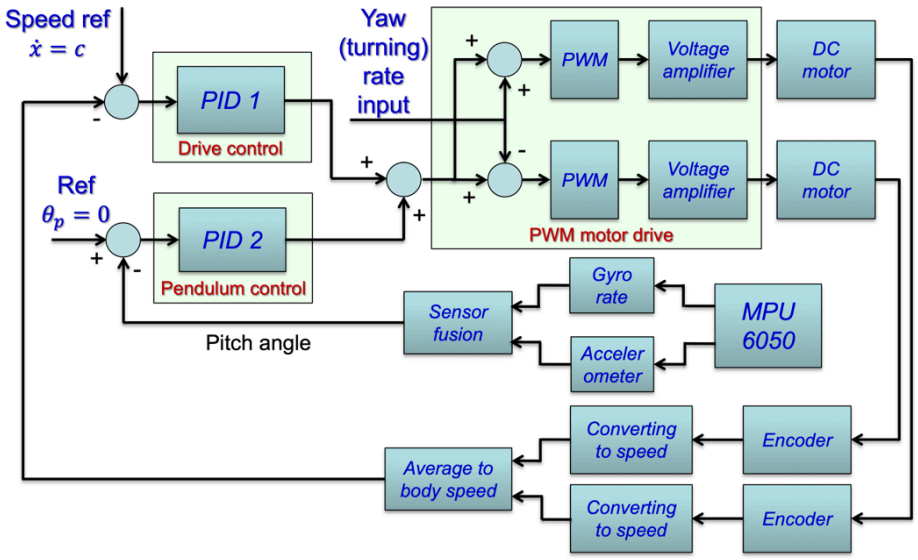


Figure 8.12: Dual-PID control architecture of a mini-Segway.

The mini-Segway pitch angle (θ_p) and cart speed ($\dot{x}_{cart}$) dynamics are NOT independent. Thus, the two PID controllers for these two coupled dynamic subsystems will influence each other, even though they are implemented in two independent control loops. Double integration operations in two PID controllers could lead to computational integration overflow, also called windup (Figure 8.12). To avoid this, one of the PID integration gains must be set to zero. Comparing the mini-Segway pendulum angle (θ_p) dynamics with the cart speed ($\dot{x}_{cart}$) dynamics, the pendulum dynamics are much faster; they require fast reaction to stabilize the inversed pendulum. As a result, the PID integration gain of the pendulum angle (θ_p) will not help much to improve the pendulum stability. In fact, in this case, the derivative gain is critical. Therefore, the PID integration gain for stabilizing pendulum angle (θ_p) is set to zero.

Control gains of two PID controllers can be tuned iteratively. Thus, we fix the first PID controller gains and tune the other one. Then we fix the tuned gains of the second PID controller and tune the first PID controller gains. This process can be repeated iteratively until we achieve satisfactory performance. The individual PID gain tuning method is introduced in Section 3.4 with the direct-tuning method, the Ziegler-Nichols step response and ultimate sensitivity methods. The detailed tuning method for Ziegler-Nichols methods can also be found in references [3] and [16]. PID gain tuning based upon the machine learning method [17] can also be used for gain tuning. If two independent PID controllers are used for controlling two individual motors (actuators), the PID control gain tuning will be even more challenge. This is one of the drawbacks mentioned earlier.

Section Summary

With the mini-Segway nonlinear model developed, this section proposes an architecture of using two independent PID controllers [3], [4], [5]. One PID controller is used to stabilize the mini-Segway pendulum position and maintain it in the upright position. The other controls the mini-Segway's forward and backward motions in cart x direction. The coupling dynamics (Problems P8.2 and P8.4) of pendulum angular displacement, forward/backward, and turning motions are not considered. To ensure that the two integrators of these two PID controllers do not fight each other, which could lead to unbounded integration values, we set the integration gain of the PID controller for stabilizing the mini-Segway pendulum at the upright position to zero. The effect of integration action for stabilizing the pendulum position is moderate because of the integration operation's slow action.

Section Keywords

Practice Problems

1. For PID control laws of continuous- and discrete-time, which method is preferred for implementation into a microcontroller for real-time control, continuous or discrete?
2. Assume you want to reduce the noise amplification of a derivative operation in a PID controller. For a physical system with its highest frequency component at f_h, what of the following is a proper low-pass filter corner frequency? Select the correct

answer.
   a. f_h
   b. $2f_h$
   c. $6f_h$
3. For the anti-windup logic used in the PID controller shown in Figure 8.10, select from the following list the three cases that the integration action is allowed.
   a. If the PID control output $u_{PID}(t)$ is saturated from above and the PID input error $e(t)$ is negative.
   b. If the PID control output $u_{PID}(t)$ is saturated from below and the PID input error $e(t)$ is positive.
   c. If the PID control output $u_{PID}(t)$ is saturated from above and the PID input error $e(t)$ is positive.
   d. If the PID control output $u_{PID}(t)$ is between the lower saturation limit l_{min} and upper saturation limit l_{max}.
4. For the dual-PID control architecture for the mini-Segway, select from the following list the correct control targets for the two PID controllers.
5. Mini-Segway pendulum angle and cart speed.
6. Mini-Segway pendulum angle and turning speed.
7. Mini-Segway cart and turning speeds.

This page titled 8.2: Model-Free Dual-PID Mini-Segway Control System Architecture is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8.3: Model-Based Mini-Segway Control

Learning Objectives

After completing this section, you will be able to:

1. Explain the process for obtaining a linearized mini-Segway model in the state-space form for designing model-based controllers.
2. Design an LQR (linear quadratic regulation) mini-Segway controller.
3. Describe the process of designing an LQTI (linear quadratic tracking with integral) mini-Segway controller for stabilization and driving.

Before starting a model-based control design, we must develop a linear state-space model of a mini-Segway based on the three linearized differential equations shown in equation ??? at $\theta_p = \dot{\theta}_p = 0$; $x_{cart} = \dot{x}_{cart} = 0$; and $\dot{\phi} = 0$. A state-space system model can be put into the following form.

$$\begin{aligned} \dot{x}_p &= A_p x_p + B_p u \\ y &= C_p x_p + D_p u \end{aligned} \tag{8.3.1}$$

where x_p, u, and y are the state vector, control input vector, and system output vector, respectively; and A_p, B_p, C_p, and D_p are the state-space model system, input, output, and feedforward output matrices, respectively. For a mini-Segway system, define the state, control input, and system output vectors as follows.

$$x_p = \left[x_{cart}, \dot{x}_{cart}, \theta_p,\ \dot{\theta}_p, \phi,\ \dot{\phi}\right]^T, u = \left[u_{aL},\ u_{aR}\right]^T, y = \left[\dot{x}_{cart}, \theta_p, \dot{\theta}_p, \dot{\phi}\right]^T \tag{8.3.2}$$

With these vectors defined above, the following system matrices of the linearized mini-Segway model are based upon the linearized mini-Segway dynamic equation ???.

$$\begin{aligned} A_p &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & c_4 & c_6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & c_5 & c_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & c_8 \end{bmatrix}, \\ B_p &= \begin{bmatrix} 0 & 0 \\ c_9 & c_9 \\ 0 & 0 \\ c_{10} & c_{10} \\ 0 & 0 \\ c_{11} & -c_{11} \end{bmatrix}, \\ C_p &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \\ D_p &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{8.3.3}$$

where coefficients c_1 to c_{11} are defined below.

$$
\begin{aligned}
c_1 &= m_p + \frac{2I_\omega}{r_w^2} + 2m_\omega, \\
c_2 &= I_p + 2m_p l^2, \\
c_3 &= I_{sz} + \frac{2I_\omega h^2}{r_w^2} + 2m_\omega h^2 \\
c_4 &= \frac{-2\left(K_\tau K_m + bR_a\right)\left(m_p l r_w + c_2\right)}{R_a r_w^2\left(c_1 c_2 - m_p^2 l^2\right)}, \\
c_5 &= \frac{2\left(K_\tau K_m + bR_a\right)\left(m_p l + r_w c_1\right)}{R_a r_w^2\left(c_1 c_2 - m_p^2 l^2\right)}, \\
c_6 &= \frac{-m_p^2 l^2 g}{c_1 c_2 - m_p^2 l^2}, \\
c_7 &= \frac{m_p l g c_1}{c_1 c_2 - m_p^2 l^2} \\
c_8 &= \frac{-2\left(K_\tau K_m h^2 + bhR_a\right)}{R_a r_w^2 c_3} \\
c_9 &= \frac{K_\tau\left(m_p l r_w + c_2\right)}{R_a r_w\left(c_1 c_2 - m_p^2 l^2\right)}, \\
c_{10} &= \frac{-K_\tau\left(m_p l + c_1 r_w\right)}{R_a r_w\left(c_1 c_2 - m_p^2 l^2\right)}, \\
c_{11} &= \frac{-K_\tau h}{R_a r_w c_3}
\end{aligned}
\tag{8.3.4}
$$

Utilizing the parameter values provided in Table 8.1, the numerical system matrices can be found below for matrices A_p and B_p with output and direct feedforward matrices C_p and D_p given in equation 8.3.3.

$$
A_p = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & -6.8776 & -1.6517 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 156.85 & 107.36 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & -26.450
\end{bmatrix}, B_p = \begin{bmatrix}
0 & 0 \\
2.4425 & 2.4425 \\
0 & 0 \\
-55.7028 & -55.7028 \\
0 & 0 \\
19.3887 & -19.3887
\end{bmatrix}
\tag{8.3.5}
$$

The mini-Segway system is open-loop unstable for the pendulum angle θ_p and cart displacement x_{cart} dynamics because one of its related system poles is located at $s = 9.5713 > 0$ from eigenvalues of A_p. First, we design a pendulum stabilizing controller with $u_{aL} = u_{aR}$. In this case, the system input matrix B_p can be merged into one column due to the fact that $u_{aL} = u_{aR}$. The mini-Segway turning speed dynamics related pole is stable ($-26.450 < 0$) and the turning dynamics are independent of the pendulum and cart linear motion dynamics for the linear model. As a result, we can simplify the stabilizing control design by considering the following system matrix pair with turning dynamic states removed.

$$
A_{cp} = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & -6.8776 & -1.6517 & 0 \\
0 & 0 & 0 & 1 \\
0 & 156.85 & 107.36 & 0
\end{bmatrix}, B_{cp} = \begin{bmatrix}
0 \\
4.885 \\
0 \\
-111.406
\end{bmatrix}
\tag{8.3.6}
$$

where the associated state vector is $x_{cp} = \left[x_{cart}, \dot{x}_{cart}, \theta_p, \dot{\theta}_p\right]^T$ and control input is $u_{cp} = u_{aL} = u_{aR}$, The associated state equation is shown below.

$$
\dot{x}_{cp} = A_{cp} x_{cp} + B_{cp} u_{cp} \tag{8.3.7}
$$

For the continuous-time stabilizing regulation, we will design an LQR controller with the following cost function.

$$J_{cp} = \frac{1}{2}\int_0^\infty \left(x_{cp}^T(t)\, Q_{cp} x_{cp}(t) + u_{cp}^T(t) R_{cp} u_{cp}(t)\right) dt \tag{8.3.8}$$

where $Q_{cp} \geq 0$ and $R_{cp} > 0$ are the weighting matrices to be selected. We will design a state-feedback controller because all four states can be measured by motor mounted encoders and accelerometers/rate-gyros from MPU 6050. As a result, control output u_{cp} is in the form of $u_{cp} = K_{cp} x_{cp}$, where $K_{cp} = -R_{cp}^{-1} B_{cp}^T P_{cp}$; and $P_{cp} \geq 0$ is the stabilizing solution of the following Riccati equation (See Section 3.5).

$$P_{cp} A_{cp} + A_{cp}^T P_{cp} - P_{cp} B_{cp} R_{cp}^{-1} B_{cp}^T P_{cp} + Q_{cp} = 0 \tag{8.3.9}$$

For the initial design, an initial set of LQR weighting matrices Q_{cp} and R_{cp} was chosen as $Q_{cp} = diag\ [100,\ 0.1,\ 100,\ 0.1]$ and $R_{cp} = 1$. As discussed in Section 3.5, the selection of both weighting matrices Q_{cp} and R_{cp} affects the resulting closed-loop system performance directly. Section 8.8 provides an example of experimentally tuning these weighting matrices to achieve desired performance. For the given weighting matrices, the resulting state feedback control gain is shown below.

$$K_{cp} = [-10.00,\ -10.358,\ -12.913,\ -0.966] \tag{8.3.10}$$

The associated closed-loop system pole locations are listed below. We can see that the closed-loop system is asymptotically stable because their real parts are negative.

$$s_{1,2} = -1.2754 \pm 1.1852i, s_{3,4} = -30.6237 \pm 13.6111i. \tag{8.3.11}$$

The control law described by equation 8.3.10, only stabilizes the mini-Segway and prevents the pendulum falling. This strategy cannot drive the mini-Segway in a way similar to the dual-PID controller discussed in Section 8.2. Now let's develop a model-based control strategy for both stabilization and driving with the same capability of the dual-PID strategy. In order to make the mini-Segway track the desired cart speed, we need to add a tracking error state $\dot{x}_{ce} = r - \dot{x}_{cart}$ of mini-Segway cart speed to the system state vector $x_{pi} = \left[x_{cart}, \dot{x}_{cart}, \theta_p,\ \dot{\theta}_p, x_{ce}, \phi, \dot{\phi}\right]^T$ (the fifth state, x_{ce}), where x_{ce} is an integration of the tracking error $r - \dot{x}_{cart}$. As a result, the updated state-space model is now in the following form. Ideally, the error state of x_{pi} (see Section 9.3) should be used. For this study, however, the original state vector is used.

$$\begin{aligned} \dot{x}_{pi} &= A_{pi} x_{pi} + B_{pi} u + B_i r \\ y &= C_{pi} x_{pi} + D_{pi} u \end{aligned} \tag{8.3.12}$$

with the associated system matrices defined below.

$$A_{pi} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_4 & c_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & c_5 & c_7 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_8 \end{bmatrix}, B_p = \begin{bmatrix} 0 & 0 \\ c_9 & c_9 \\ 0 & 0 \\ c_{10} & c_9 \\ 0 & 0 \\ 0 & 0 \\ c_{11} & -c_{11} \end{bmatrix}, B_i = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{8.3.13}$$

$$C_{pi} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, D_{pi} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

where parameters c_1 to c_{11} are defined in equation 8.3.4. We want to design a mini-Segway controller capable of stabilizing the pendulum position and tracking the desired cart speed ($\dot{x}_{cart}$). Note that the turning motion is controlled in open-loop. For the same reason stated before, we drop states ϕ and $\dot{\phi}$ in the state vector for state feedback tracking control design with the following system matrices.

$$A_{cpi}=\begin{bmatrix}0 & 1 & 0 & 0 & 0\\ 0 & -6.8776 & -1.6517 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\\ 0 & 156.85 & 107.36 & 0 & 0\\ 0 & -1 & 0 & 0 & 0\end{bmatrix}, B_{cpi}=\begin{bmatrix}0\\ 4.885\\ 0\\ -111.406\\ 0\end{bmatrix}, B_{ci}=\begin{bmatrix}0\\ 0\\ 0\\ 0\\ 1\end{bmatrix} \tag{8.3.14}$$

where the associated state vector is $x_{cpi}=\left[x_{cart},\dot{x}_{cart},\theta_p,\ \dot{\theta}_p,x_{ce}\right]^T$ and system state equation is $\dot{x}_{cpi}=A_{cpi}x_{cpi}+B_{cpi}u_{cp}+B_{ci}r$. For the continuous-time control design of tracking cart speed [10], [11], [18], we can design a state feedback LQR controller with the following cost function. All four physical states are directly or indirectly measurable (see earlier discussions). We can calculate the cart speed error state in real time within the microcontroller.

$$J_{cpi}=\frac{1}{2}\int_0^\infty\left(x_{cpi}^T(t)\,Q_{cpi}x_{cpi}(t)+u_{cp}^T(t)R_{cpi}u_{cp}(t)\right)dt \tag{8.3.15}$$

where $Q_{cpi}\geq 0$ and $R_{cpi}>0$ are the weighting matrices to be selected. As a result, control output u_{cp} is in the form of $u_{cp}=K_{cpi}x_{cpi}$, where $K_{cpi}=\ -R_{cpi}^{-1}B_{cpi}^TP_{cpi}$, and $P_{cpi}\geq 0$ is the stabilizing solution of the following Riccati equation.

$$P_{cpi}A_{cpi}+A_{cpi}^TP_{cpi}-P_{cpi}B_{cpi}R_{cpi}^{-1}B_{cpi}^TP_{cpi}+Q_{cpi}=0 \tag{8.3.16}$$

For the initial design, an initial set of LQR weighting matrices Q_{cpi} and R_{cpi} was chosen as $Q_{cpi}=diag\ \left[100,\ 0.1,\ 100,\ 0.1,\ 10^4\right]$ and $R_{cpi}=1$. Because the added integration mode (fifth state x_{ce}) is not controllable and it is not stable due to $A_{cpi}(5,5)=0)$, there is no positive definite solution for Riccati equation 8.3.16. We use an approximation of the integration for LQR control design by setting $A_{cpi}(5,5)=-0.0001=-\varepsilon$, making the integration model stabilizable. The resulting state feedback gain is shown below with control $u_{cpi}=K_{cpi}x_{cpi}$.

$$K_{cpi}=[-10.00,\ -38.16,\ -21.47,\ -2.2524,\ 90.495] \tag{8.3.17}$$

The control gains in equation 8.3.17 associated with cart position, cart speed, pendulum position, and speed (the first four states) are similar to these for state feedback case in equation 8.3.10; the associated weightings are the same. The closed-loop system pole locations are listed below. The closed-loop system is asymptotically stable. One pair of complex poles (pendulum mode) is located very close to those for the LQR design. The other pair (cart linear motion mode) is moved further to the left.

$$s_1=-9.9504\times 10^{-6}, s_{2,3}=-4.4374\pm 2.9257i, s_{4,5}=-30.1247\pm 15.321i. \tag{8.3.18}$$

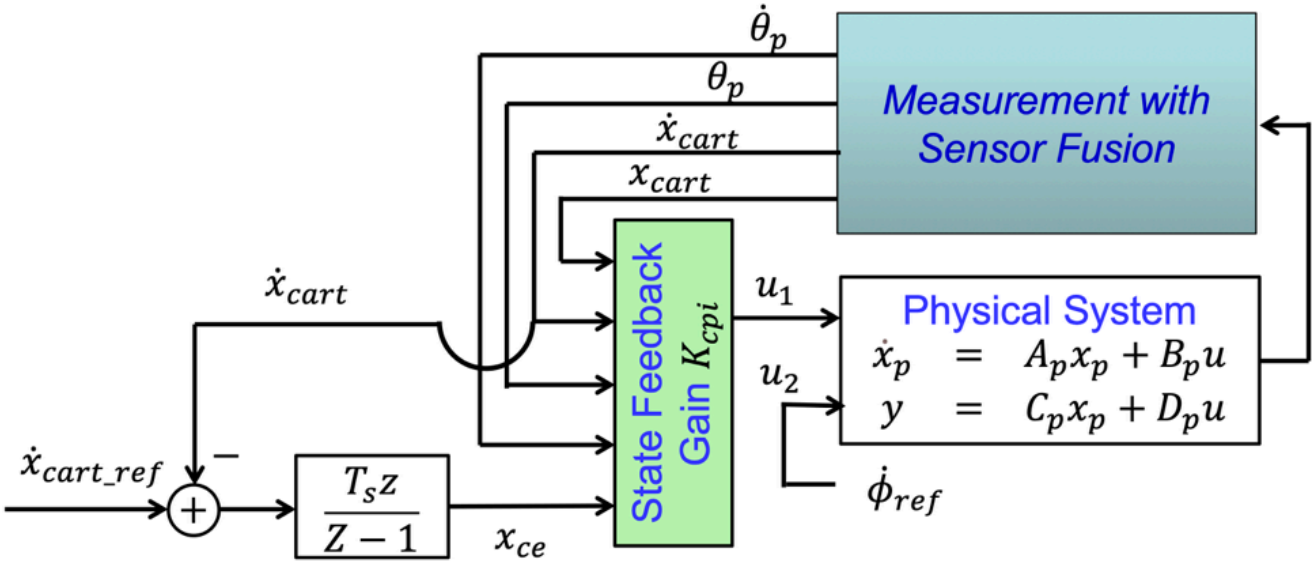


Figure 8.13: The LQTI control system architecture.

Figure 8.13 shows a complete architecture of the mini-Segway LQTI control system. The physical system (physical domain) is represented using the original linear state-space model shown in equation 8.3.1 with physical sensors (MPU 6050 and two wheel encoders) and actuators (two DC motors). The rest of the block diagram is located in the information domain. This control

system is comparable with the dual-PID control system discussed in Section 8.2. The turning motion is controlled in an open-loop because it is decoupled from the rest of the dynamics for the linearized model. The pendulum stabilization and cart driving motion are controlled by the designed LQTI state feedback controller. The cart position and speed (x_{cart} and $\dot{x}_{cart}$) are calculated from both wheel axis speed based upon the two encoders installed on these two motor shafts (see Section 7.3). We assume that there is no slip between wheel and road surface, the pendulum position is calculated based on the sensor fusion (see Section 7.5) using both accelerometer and rate-gyro signals from MPU 6050, and the pendulum speed is directly measured by the y-axis rate-gyro signal from MPU 6050. We complete the integration action using a digital integration of the error signal $\dot{x}_{ce}$.

Even though the controller is designed in the continuous-time domain, it can be directly used for the sampled sensor signals and held output signals. This assumes the sample period is small enough, which is validated in simulation studies in Section 8.6 with a sample frequency of 100 Hz. On the other hand, because the control design model is linearized around the pendulum position and cart speed at zero, the designed LQTI control is only good for low cart speed tracking with small pendulum perturbation.

This section provides two model-based stabilizing controllers (LQR and LQTI) for the mini-Segway. The LQTI controller can also be used to drive the mini-Segway. Both model-based controllers will be validated in simulation studies conducted in Section 8.6.

Section Summary

This section addresses the model-based control design process (see Section 3.5) for the mini-Segway, modeled in Section 8.1. It utilizes LQR (linear quadratic regulation) [6], [7], [8], [9] and LQTI (linear quadratic tracking with integral) [10], [11]. First, to apply model-based control design methodologies, we develop a state-space linearized mini-Segway model with respect to its normal operation condition, based on the nonlinear model developed in Section 8.1. We then, populate the model parameter using the values in Table 8.1. Then we design an LQR regulator for stabilizing the mini-Segway to keep the mini-Segway pendulum in its upright position. Last, we add an error integral state of mini-Segway cart speed to the state-space model [10]. We also design an LQTI controller to track the mini-Segway cart speed with its pendulum position regulated in its upright position. The LQTI control strategy is a model-based one comparable with the dual-PID strategy discussed in Section 8.2, where the turning is controlled in open-loop, too.

Section Keywords

Practice Problems

1. What is the state-space model order shown in equation 8.3.1? Select the correct answer.
   a. 4
   b. 5
   c. 6
2. True or false? In the state-space model shown in equation 8.3.1, the cart motion state x_{cart} and pendulum position state θ_p are independent from its turning state $\dot{\phi}$.
3. To make the mini-Segway track a desired cart speed in LQTI control, an additional integral state x_{ce} was created. Is this state (mode) controllable (Yes or No) and why?
4. To make it possible to design the LQTI controller, the integration pole location in the state-space design model was adjusted from zero to $-\varepsilon$ with $\varepsilon > 0$. Please discuss how the ε affects the control gain related to the error state x_{ce}.

This page titled 8.3: Model-Based Mini-Segway Control is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input

Learning Objectives

After completing this section, you will be able to:

1. Explain the concepts of minimal phase (MP) and non-minimal phase (NMP) dynamic systems.
2. Convert a continuous-time NMP system to a discrete-time MP system with the square-pulse sample and hold input (SPSHI).
3. Describe the dual-loop control architecture for mini-Segway pendulum stabilization using square-pulse sample and hold input.

This section discusses the control design for non-minimum-phase (NMP) systems and minimal-phase (MP) systems. An NMP system transfer function has at least one zero located in the open-right complex plane. An MP system has all its zeros in the open-left complex plane (inside the unit circle for discrete-time systems). Root Locus analysis [3] indicates that the closed-loop system poles move towards the open-loop zeros as the system proportional gain increases. This change results in unstable closed-loop dynamics at high control gain, preventing our using a high-gain controller to achieve the desired closed-loop performance. In addition, asymptotic tracking is difficult to achieve when the plant system is NMP; control methods can achieve only approximate output tracking for NMP systems [19]–[22].

In general, NMP systems are difficult to control, but discretization with zero-order-hold (ZOH) may alleviate the problem. References [23]–[29] show that an NMP continuous-time system can be mapped into an MP discrete-time system. This happens, for example, when a ZOH input is used with a sufficiently large sampling period [24]. Unfortunately, a precise mathematical formula for calculating the smallest feasible sampling period has not been found. For an NMP continuous-time system, a feasible sampling period is defined as one that results in an MP discrete-time system. Based on reference [24], the range of feasible sampling periods is determined by using a symbolic approach in [25] and genetic algorithms in [26]. These methods for finding feasible sampling periods with a ZOH input when the continuous-time system is minimum-phase (MP) can be found in [27]-[29]. For stable NMP systems, the Nyquist algorithm is used in [27]. These results were extended in [28] to unstable systems with no poles or zeros on the imaginary axis.

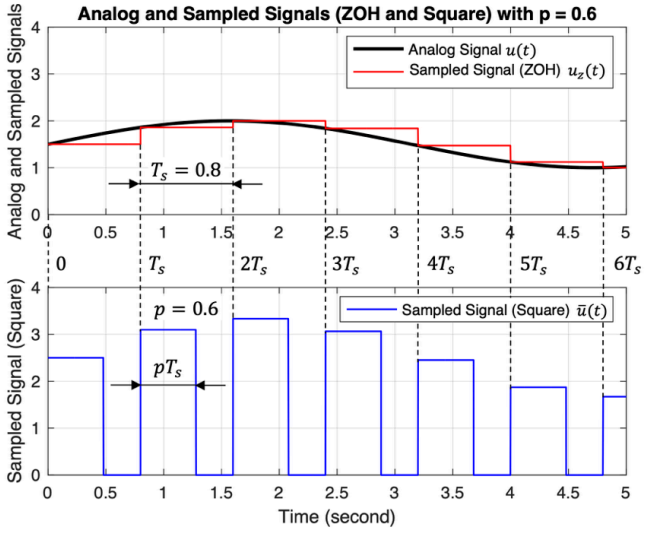


Figure 8.14: The square-pulse sample and hold input.

The mini-Segway system is NMP with at least one zero located in the open right-half complex plan, limiting the achievable performance. In reference [12], the special sample and hold input (SHI) functions, square-pulse, and forward and backward

triangle SHIs, indicate that by selecting the proper sampling period and duty cycle, the resulting discrete-time system can be minimum phase (MP). This is so even though the original continuous-time system is not MP with extended feasible region compared with the zero-order SHI. This section studies the case utilizing the square-pulse sample and hold input to result in an MP discrete-time system. Figure 8.14 compares the square-pulse sample and hold input to the conventional zero-order sample and hold input. The top plot shows an analog signal $u(t)$ (solid-line) sampled by the zero-order sample and hold input with a sample period of T_s (dotted-line). The middle plot shows the sampled (digital) signal $u_z(t)$ by the zero-order sample and hold input. The bottom plot shows an analog signal $\overline{u}(t)$ (solid-line) sampled by the square-pulse sample and hold input with the same sample period of T_s (solid-line). In order to ensure the sampled signal has magnitude comparable with the zero-order hold, we set the magnitude of $\overline{u}(t)$ in such a way that the integrations of zero-order and square-pulse sample and hold inputs over one sample period T_s are the same with the following mathematical expression.

\[\overline{u}(t)=\left\{ \begin{array}{cc} u_z\left(t\right), & t_k

where $p \in (0,\ 1]$ is the duty-cycle of the switched input by the square-pulse sample and hold input. See the bottom plot of Figure 8.14 for a graphic definition; and $u_z\,(t)$ is defined below.

\[u_z\left(t\right)=u\left(t_k\right)/p, if\ t_k

where $u(t_k)$ is sampled magnitude of continuous-time signal $u(t)$ at time t_k by the zero-order sample and hold input. When $p = 1$, the square-pulse sample and hold input becomes the zero-order one. As a result, the zero-order sample and hold input is a special case of the square-pulse one. Before introducing the derivation of the discretized system of equation ??? with $D_p = 0$, let's introduce the concept of discrete-equivalent (DE) system for the linear system defined below.

$$\begin{aligned} \dot{x}_p &= A_p x_p + B_p \overline{u}, x_p\,(t_0) = x_0 \\ y &= C_p x_p \end{aligned} \tag{8.4.1}$$

Definition 8.1 *[12]*

The time-invariant linear system below, with zero-order sample and hold input,

$$\begin{aligned} \dot{\overline{x}}_p &= \overline{A}_p \overline{x}_p + \overline{B}_p u_z,\ \overline{x}_p\,(t_0) = x_0 \\ \overline{y} &= \overline{C}_p \overline{x}_p \end{aligned} \tag{8.4.2}$$

is a discrete-equivalent (DE) system of 8.4.1 if the state variables of 8.4.1 and 8.4.2 match at the sampling times t_k for $k = 0,\ 1,\ 2,\ \cdots$.

The state variables in system 8.4.1 at time $t_{k+1} = t_k + T_s$ can be derived in two steps. First, we derive the state time response from time t_k to $t_k + pT_s$ as below.

$$\begin{aligned} x_p\,(t_k + pT_p) &= e^{A_p T_s p} x_p\,(t_k) + \int_0^{pT_s} e^{A_p (pT_s - \tau)} B_p \overline{u}\,(t_k + \tau)\, d\tau \\ &= e^{A_p T_s p} x_p\,(t_k) + A_p^{-1} \left(e^{A_p T_s p} - I\right) (B_p/p)\, u\,(t_k) \end{aligned} \tag{8.4.3}$$

and then from time t_k to t_{k+1} as below, based on equation 8.4.3.

$$\begin{aligned} x_p\,(t_{k+1}) &= e^{A_p T_s (1-p)} x_p\,(t_k + pT_p) \\ &= e^{A_p T_s} x_p\,(t_k) + A_p^{-1} e^{A_p T_s} \left(I - e^{-A_p T_s p}\right) (B_p/p)\, u\,(t_k) \end{aligned} \tag{8.4.4}$$

The state variables in system 8.4.2 from time t_k to t_{k+1} can be derived as below.

$$\begin{aligned} \overline{x}_p\,(t_{k+1}) &= e^{\overline{A}_p T_s} \overline{x}_p\,(t_k) + \int_0^{T_s} e^{\overline{A}_p (T_s - \tau)} \overline{B}_p u_z\,(t_k + \tau)\, d\tau \\ &= e^{\overline{A}_p T_s} \overline{x}_p\,(t_k) + \overline{A}_p^{-1} \left(e^{\overline{A}_p T_s} - I\right) \left(\overline{B}_p/p\right) u\,(t_k) \end{aligned} \tag{8.4.5}$$

Comparing system models 8.4.4 and 8.4.5, it is clear that system models 8.4.1 and 8.4.2 will be discrete-equivalent (DE) if the following conditions are met,

$$\overline{A}_p = A_p,\ \overline{B}_p = \left(e^{A_pT_s} - I\right)^{-1} e^{A_pT_s}\left(I - e^{-A_pT_s}\right) B_p \tag{8.4.6}$$

Now, we are ready to consider the following standard discrete-time (DT) system with the same initial conditions as the DE system in 8.4.2.

$$\begin{aligned} x_p(t_{k+1}) &= A_{pd}x_p(t_k) + B_{pd}u(t_k), x_p\left(t_0\right) = x_0 \\ y(t_k) &= C_{pd}x_p\left(t_k\right) \end{aligned} \tag{8.4.7}$$

The DT system 8.4.3 will have the same state and output variables as the DE system 8.4.2 with square-pulse sample and hold input, described by 8.4.2 and 8.4.6, at sampling time $t_k\ (k = 0,\ 1,\ 2,\ \cdots)$, if $A_{pd} = e^{A_pT_s}$ and $C_{pd} = C_p$ with the following condition satisfied.

$$B_{pd} = A_p^{-1} e^{A_pT_s}\left(I - e^{-A_pT_s}\right) B_p/p \tag{8.4.8}$$

The continuous-time system in 8.4.2 with the SPSHI, its DE system 8.4.6, and the corresponding DT system 8.4.7 have the same state and output variables at the sampling time $t_k\ (k = 0,\ 1,\ 2,\ \cdots)$.

Remark 8.1

The discrete-time (DT) system matrices A_{pd} and C_{pd} are the same for the zero-order and square-pulse sample and hold inputs. However, input matrix B_{pd} in equation 8.4.8 is different from the zero-order one. Because the system zeros are affected by A_{pd}, B_{pd}, and C_{pd}, with $p \in (0,\ 1)$, the DT system 8.4.7 could have different zeros comparing with the system with $p = 1$.

Remark 8.2

Equation 8.4.8 indicates that in order to make the square-pulse sample and hold input work for $p \in (0,\ 1)$, the continuous-time system matrix A_p cannot have any zero eigenvalue because the inverse of A_p is required in equation 8.4.8. If there is at least one eigenvalue at the origin of the complex plan, a stabilizing controller is needed before SPSHI can be applied. However, the solution for term $A_p^{-1}e^{A_pT_s}\left(I - e^{-A_pT_s}\right)$ exists even when A_p is not invertible because term $\left(I - e^{-A_pT_s}\right)$ has a factor of A_p, which can be used to cancel the term A_p^{-1}. In this case, calculating term $A_p^{-1}e^{A_pT_s}\left(I - e^{-A_pT_s}\right)$ requires Taylor series expansion of $e^{-A_pT_s}$ and summation of expanded series as below.

$$A_p^{-1}e^{A_pT_s}\left(I - e^{-A_pT_s}\right) = pT_s e^{A_pT_s}\left(\frac{I - A_pT_s}{2!} + \frac{\left(A_pT_s\right)^2}{3!} - \frac{\left(A_pT_s\right)^3}{4!} + \cdots\right) \tag{8.4.9}$$

Because there is no closed form solution of the series in equation 8.4.9, the existence of A_p^{-1} is required for simplified calculation.

Note that this study uses the mini-Segway model described by equations ??? in Section 8.3 with full state (six states), control, and output vectors defined below.

$$x_p = \left[x_{cart}, \dot{x}_{cart}, \theta_p,\ \dot{\theta}_p, \phi, \dot{\phi}\right]^T, u_p = \left[u_{aL},\ u_{aR}\right]^T, y = \left[\theta_p, \dot{\theta}_p\ \right]^T \tag{8.4.10}$$

The associated system matrix values for system ??? are provided below. For the rest of this section, these matrix values in equation 8.4.11 are used for control design.

$$
\begin{aligned}
A_p &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -6.8776 & -1.6517 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 156.85 & 107.36 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -26.450 \end{bmatrix}, \\
B_p &= \begin{bmatrix} 0 & 0 \\ 2.4425 & 2.4425 \\ 0 & 0 \\ -55.7028 & -55.7028 \\ 0 & 0 \\ 19.3887 & -19.3887 \end{bmatrix} \\
C_p &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \\
D_p &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}
\end{aligned}
\tag{8.4.11}
$$

The full-state system defined by the matrices in equation 8.4.11 has two poles located at the origin of the complex plan, making the system matrix A_p not invertible. As discussed in Remark 8.2, the square-pulse sample and hold and input technique cannot be applied directly to convert the continuous-time NMP system into a discrete-time MP system. Therefore, a continuous-time stabilizing LQR controller K_p (see Section 8.3) is designed with $Q_p = diag\left[100, 0.1, 300, 0.1, 10, 0.01\right]$ and $R_p = diag[1,\ 1]$ to minimize the following cost function.

$$
J_p = \frac{1}{2}\int_0^\infty \left(x_p^T\left(t\right) Q_p x_p\left(t\right) + u_p^T(t) R_p u_p(t)\right) dt \tag{8.4.12}
$$

The designed controller K_p stabilizes the mini-Segway system by moving all the closed-loop poles to the open left-half complex plan. The closed-loop system matrix $A_p + B_p K_p$ is then invertible with the following K_p.

$$
K_p = \begin{bmatrix} -7.07 & -9.49 & -14.64 & -0.91 & 2.24 & 0.08 \\ -7.07 & -9.49 & -14.64 & -0.91 & -2.24 & -0.08 \end{bmatrix} \tag{8.4.13}
$$

Figure 8.15 shows the proposed dual-loop architecture for the SPSHI application. The inner-loop LQR controller stabilizes the mini-Segway system so that the SPSHI technique can be applied. The outer-loop utilizes the SPSHI for discrete-time control output with a standard ZOSHI to sample the sensor signals. During the implementation period, the continuous-time controller K_p will be implemented as a discrete-time controller with ZOSHI at a sample rate much faster than the outer-loop SPSHI discrete-time controller.

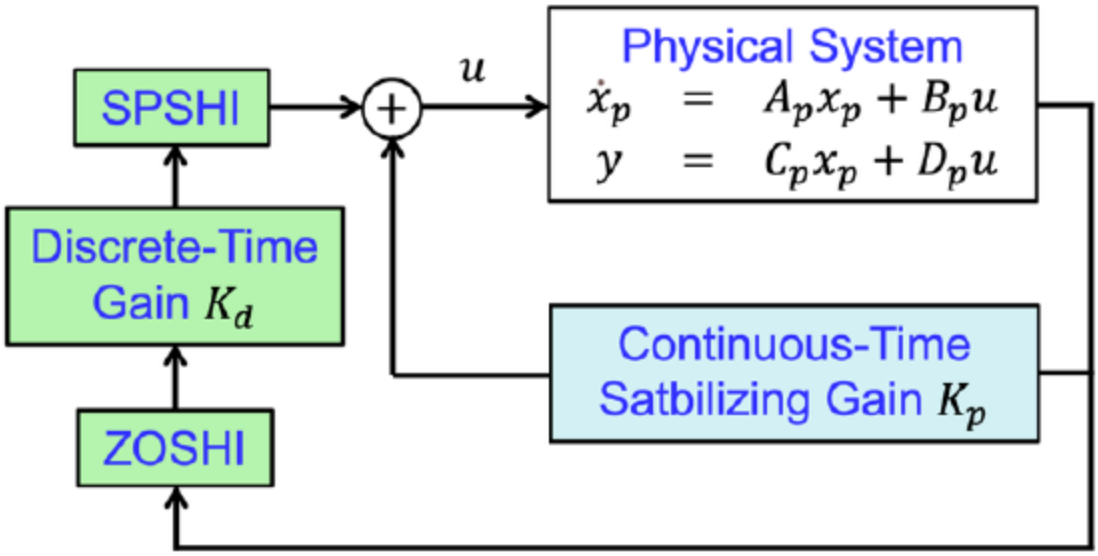


Figure 8.15: Dual-loop SPSHI stabilization system architecture.

Note that stabilized pole locations of the inner-loop closed-loop system are listed in the equation below.

$$\begin{aligned} s_{1,2} &= -0.9647 \pm 0.9081i, \\ s_3 &= -3.2859 \\ s_4 &= -26.3879 \\ s_{5,6} &= -30.1103 \pm 21.5604i \end{aligned} \tag{8.4.14}$$

For sample period $T_s \in (0, 1]$ and SPSHI duty-cycle $p \in (0,\ 1]$, we can calculate the zero locations of the K_{cp} stabilized mini-Segway system. The shaded area in Figure 8.16 shows the sample and hold input (SHI) feasible region. It is the common area formed by the sampling period T_s and duty-cycle p, resulting in a mini-Segway DT system for the closed-loop system with the inner-loop controller. We calculate the feasible region by an exhaustive search of sample period T_s and SPSHI duty-cycle p that results in a discrete-time MP system with respect to these outputs related to the pendulum angle states $(\theta_p, \dot{\theta}_p)$, where output matrix C_p is defined in equation 8.4.11.

The black thick horizontal line on the top of Figure 8.16 represents the feasible sampling periods of ZOSHI, a special case of SPSHI with $p = 1$ (100% duty-cycle). The feasible region corresponding to these sample period $T_s = \delta$ and square pulse duty-cycle p results in an MP discrete-time system. From Figure 8.16, the smallest feasible sampling period of ZOSHI (represented by the black-thick horizon line) is about as large as 0.61 second. This sampling period can be reduced to 0.1184 second using SPSHI with an extremely small duty-cycle p. For example, if we select a sampling period of 0.2 second (see the vertical dotted-line), we can choose a wide range of duty-cycles between almost zero and 0.6. A small sampling period is very important for improving closed-loop system performance of the mini-Segway using the outer-loop controller.

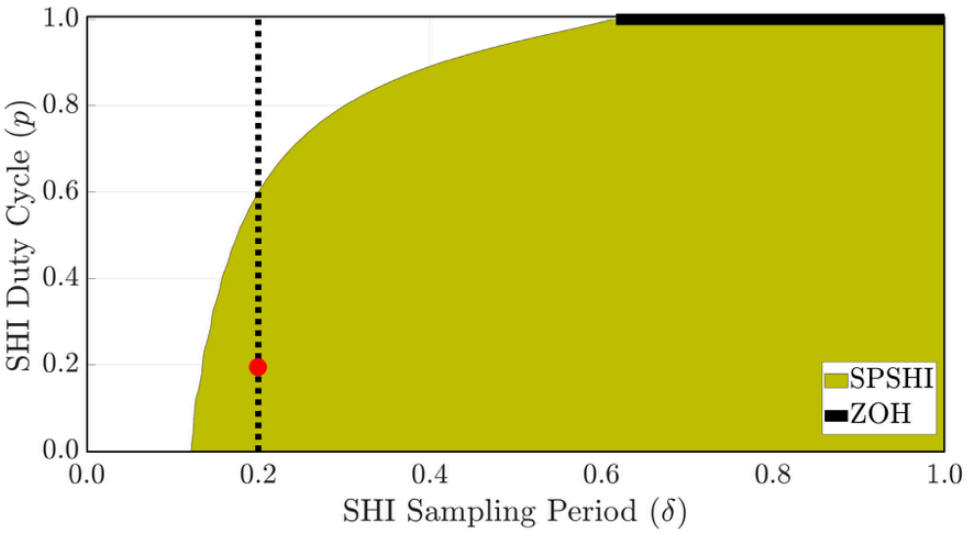


Figure 8.16: Feasible sample period of ZOSHI and SPSHI as a function of $T_s = \delta$ and p.

Equation 8.4.14 shows the pole locations for the inner-loop closed-loop system, with two oscillation frequencies at 1.74 (cart position) and $33.5\ rad/s$ (pendulum position). The stabilized system bandwidths with the inner-loop LQR controller are calculated in Hz. They are 0.277 and $5.33\ Hz$ for cart displacement and pendulum pitch angle, respectively. To ensure the outer-loop controller sample frequency is adequate to cover system dynamics for both modes, the required sampling frequency is at least two times faster than the system dynamics based upon the Nyquist-Shannon theory (see Section 4.5). However, at least eight times the highest frequency is required, that is, 42.65 Hz for the outer-loop discrete-time controller with SPSHI. From Figure 8.16, we can see that it is impossible to have the outer-loop discrete-time controller cover system dynamics for both modes. Consequently, for the same reason, only the mode associated with the cart displacement is considered in this case. It requires a controller with a lowest sample frequency of $1.816\ Hz$ (eight times its oscillation frequency) with a sample period of 0.55 second; see Section 4.5 and reference [30].

Therefore, a small sampling period within the feasible region shown in Figure 8.16 is preferred. With the SPSHI, the smallest feasible sample period is about 0.12 second. Due to the experimental validation result in reference [31], a small duty-cycle leads to improved performance over a larger one. This can also be interpreted as the fast action effect of the SPSHI. As the SPSHI duty-cycle reduces, its reaction speed reduces. As a result, the outer-loop sampling period is chosen to be $T_s = 0.2$ second (marked with the dashed line in Figure 8.16) with associated duty-cycle of square-pulse sample and hold input $p = 0.2$ (20% duty-cycle). The selected SPSHI parameters T_s and p are marked by the red dot in Figure 8.16.

Because the open-loop mini-Segway system is NMP, high gain control could destabilize the system [3]. With the SPSHI parameters selected in the previous paragraph, we can convert the continuous-time NMP mini-Segway system model into a discrete-time MP system. With the applied continuous-time controller K_p, the equivalent discrete-time (DT) system becomes the system below.

$$\begin{aligned} x_{pd}(k+1) &= A_{pd}x_{pd}(k) + B_{pd}u_d(k) \\ x_{pd}(t_0) &= x_0 \\ y(t_k) &= C_{pd}x_{pd}(k) \end{aligned} \tag{8.4.15}$$

where $A_{pd} = e^{(A_p - B_p K_p)T_s}$ and $C_{pd} = C_p$ with the B_{pd} defined below.

$$B_{pd} = A_p^{-1} e^{A_p T_s} \left(I - e^{-A_p p T_s}\right) B_p / p \tag{8.4.16}$$

Assume we want to further improve the stabilization performance of the mini-Segway system by reducing cart speed magnitude. To achieve this, we design the outer-loop controller—a discrete-time LQR (referred as DLQR)—with the SPSHI (see the outer-loop controller in Figure 8.15) based upon the system model 8.4.15. This provides additional control effort for further performance improvement, where the DLQR cost function is defined below.

$$J_d = \frac{1}{2}\sum_{k=0}^{\infty} x_{pd}^T(k)\, Q_d x_{pd}(k) + u_d^T(k) R_d u_d(k) \tag{8.4.17}$$

with weighting matrices $Q_d = Q_p$, which is defined in the paragraph before equation 8.4.12, and $R_d = diag[0.9,\ 0.9]$. The selection of DLQR weighting matrices (Q_d and R_d) is critical for this study. We must guarantee an adequate control effort while maintaining both actuators within their saturated input voltage of $\pm 12\ V$. Reference [13] optimizes the control weighting matrix R_d with fixed Q_d, and $R_d = diag[0.9,\ 0.9]$ is within the optimal range.

Using the MATLAB command $dlqr$ to design the discrete-time controller K_d results in the following control gain.

$$K_d = \begin{bmatrix} -4.6017 & -4.6526 & -0.8251 & -0.2143 & 0.6263 & 0.0236 \\ -4.6017 & -4.6526 & -0.8251 & -0.2143 & -0.6263 & -0.0236 \end{bmatrix} \tag{8.4.18}$$

From the discrete-time control gain shown in equation 8.4.18, we can see that the control gains associated with the mini-Segway cart displacement and speed (the first and second columns) are much larger than the gains for the pendulum angle and angular velocity (the third and fourth columns). Recall that the sample period selected for the discrete-time controller can compensate only the cart displacement dynamics because the pendulum one is at a much higher frequency. The last two columns of the control gain in equation 8.4.18 are associated with the cart turning angle. For the pendulum stabilization, these two feedback signals will cancel each other with respect to the pendulum and cart dynamics.

Section Summary

This section introduces control design for discrete-time controllers with a sample and hold input different from a conventional zero-order sample and hold input. First, we discuss the square pulse sample and hold input. The mini-Segway system is non-minimal phase (NMP) with at least one zero located in the open right-half complex plan; this limits the achievable performance [12] by closed-loop control strategy. We found that by utilizing the special sample and hold input, we can transfer a non-minimal phase continuous-time plan into a minimal phase (MP) discrete-time one. This section demonstrates this phenomenon using our mini-Segway model [13] as an example. It then proceeds to design a dual-loop stabilizing controller.

Section Keywords

Practice Problems

1. Select the correct answer from the following list to complete the sentence: The square-pulse sample and hold input
   a. can convert a continuous-time NMP system into a discrete-time MP system.
   b. cannot change the continuous-time NMP property for the discretized system.
   c. can convert a continuous-time MP system into a discrete-time NMP system.
2. Drawing on equation 8.4.8, discuss why inverse of A_p is required. Note that $A_p^{-1}\left(e^{A_p T_s} - I\right)$ can be calculated even when A_p is not invertible.
3. Why is an inner-loop stabilizing regulator needed before applying the square-pulse sample and hold input technique? Select the correct answer.
   a. The mini-Segway system is stable.
   b. The mini-Segway system has at least one pole at the origin of the complex plan.
   c. The mini-Segway system is unstable.
4. Why can't the selected sample period ($T_s = 0.2$ second) for the outer-loop discrete-time controller compensate the pendulum dynamics? Select the correct answer.
   a. The mode frequency of the pendulum dynamics is much higher than the sample frequency.
   b. The mode frequency of the pendulum dynamics is much lower than the sample frequency.

This page titled 8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller

Learning Objectives

After completing this section, you will be able to:

1. Explain the concept of software-in-the-loop (SiL) simulations.
2. Describe the process of implementing a discrete-time PID controller using zero-order sample and hold input in the SiL simulation environment.

We first need to implement the dual-PID control strategy (see Figure 8.12) into the MATLAB/Simulink environment so that we can conduct the simulation study, utilizing the linearized system model. In order to avoid PID controller integration windup, we will use the anti-windup PID controller structure (see Figure 8.11) for the dual-PID control strategy (see Figure 8.12). Figure 8.17 shows the simulation system architecture implemented in the MATLAB/Simulink diagram. The PID controller called Cart Speed PID helps regulate the mini-Segway speed. The PID controller called Pendulum Position PID helps maintain the mini-Segway pendulum position at its upright position with reference $\theta_p = 0$. Even though two independent PID controllers are used, the mini-Segway pendulum and cart motion dynamics are coupled. As a result, tuning of the pendulum and cart motion PID control gains needs to consider the coupling effect.

As Figure 8.17 indicates, we can use the linearized mini-Segway system model defined in equations ??? to ??? for the Simulink model with system matrices defined in equation ???. Because we are using the discrete-time PID controllers for simulation study, we use zero-order sample and holder inputs before control inputs of left and right motors and after the system outputs for feedback controls. The sample period selected for this study is $T_s = 0.01$ second. The DC-motor armature voltages of both left and right motors are limited by the battery voltage through an H-bridge control circuit (see Chapter 6). Thus, we use two Simulink saturation blocks to limit controlled armature voltages of both left and right motors to ± 12 volts. (See the two saturation blocks right before the control sample and holder inputs in the middle section of Figure 8.17.) For mini-Segway turning control, we use a difference armature voltage, called Turning_CMD, to control the left and right motors and to turn the mini-Segway to turn left or right, respectively.

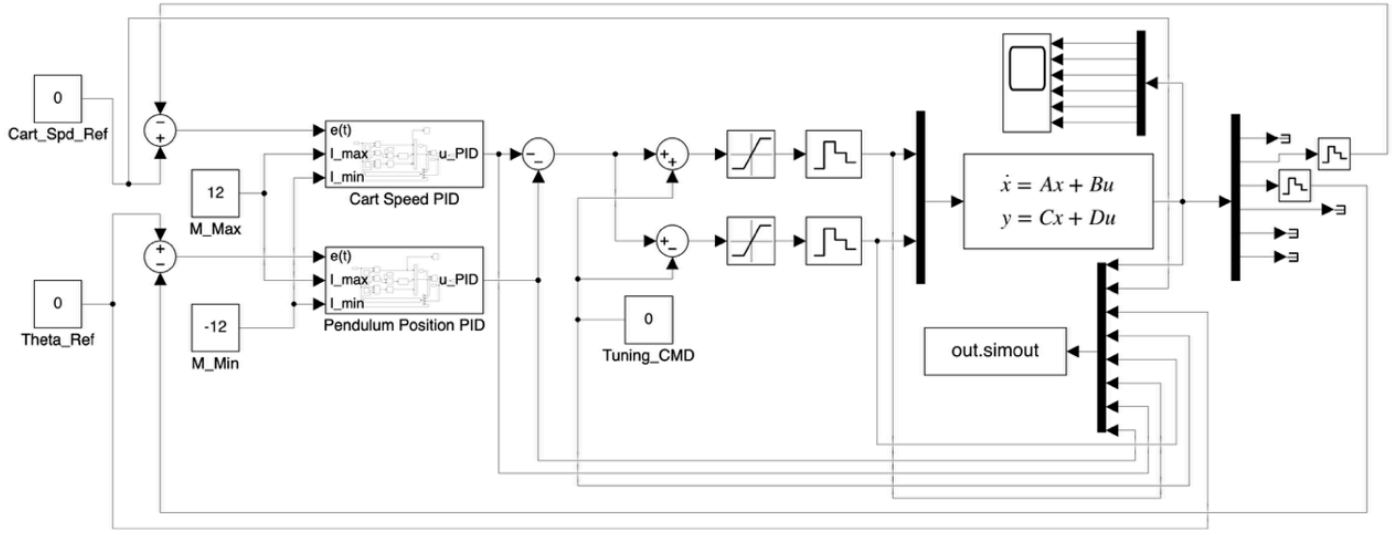


Figure 8.17: SiL simulation environment in MATLAB/Simulink.

As discussed earlier, we use two anti-windup PID controllers to generate stabilizing control for pendulum motion θ_p and cart linear motion control $\dot{x}_{cart}$, where the anti-windup upper and lower limits are selected as $+12$ and -12 volts (same as these for saturations), respectively. That is, when PID control output is greater than $+12$ volts, the integration action in the PID controller will be disabled when error is greater than zero. When PID control output is less than -12 volts, the integration action in the PID

controller will be disabled when error is less than zero. From the system control input matrix B_p defined in equation ???, we can see that the control gains related to pendulum stabilization (two entries on the fourth row of matrix B_p) are negative. In order to stabilize the pendulum motion with positive PID gains, signs of both PID control outputs are reversed. PID gains for the cart speed control are negative due to positive raw of input matrix B_p (associated with cart speed control). Finally, we generate both the mini-Segway pendulum position and cart speed error signals by subtracting the sampled values from the associated reference signals (Theta_Ref and Cart_Spd_Ref), respectively (see the left side of Figure 8.17).

For the dual-PID control strategy, we will conduct four simulation studies: pendulum stabilization; mini-Segway driving with its pendulum stabilized; mini-Segway turning with its pendulum stabilized; and combined simulation with stabilization, driving, and turning. The stabilizing simulation uses the Simulink model shown in Figure 8.17 with both reference signals and turning command (Theta_Ref, Cart_Spd_Ref, and Turning_CMD) equal to zero. That is, we want the mini-Segway pendulum to be regulated at its upright position with zero cart and turning speeds. Because the pendulum and cart motion dynamics are coupled, it is impossible to maintain cart speed at zero while regulating the pendulum position. After carefully tuning these two PID control gains, we use the gains listed in Table 8.2.

Table 8.2: PID gains for both pendulum and cart motion controller.

Proportional	3	Proportional	-3
Integral	0	Integral	-1
Derivative	0.3	Derivative	0.0 $(???)$

We conducted the simulation study with both PID controller gains presented in Table 8.2. Here, the derivative gain of the cart speed PID controller was set to zero, initial pendulum position equal to 0.2 (rad), and initial cart speed equal to zero. Again, the negative gains for cart speed control in Table 8.2 are due to the required opposite compensation for cart speed and pendulum stabilization control. Figure 8.18 shows the simulation results. We can see that both the mini-Segway pendulum position and cart speed converge to zero in about three seconds. Note that the control signal of the cart speed has about 150° phase lead, indicating these two control signals are fighting each other during the stabilization convergent process.

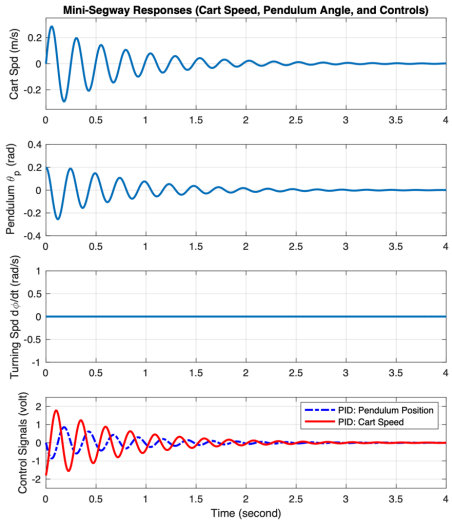


Figure 8.18: SiL simulation with dual-PID control (cart PID derivative gain equal to 0.0).

The second simulation (of pendulum stabilization) was conducted with both PID controller gains presented as in Table 8.2. However, the derivative gain of the cart speed PID controller was set to 0.1 with the same initial conditions as before. Figure 8.19 shows the simulation results. We can see that both the mini-Segway pendulum position and cart speed converge to zero in about 0.8 second. This is much faster than the case with the derivative gain of the cart speed PID equal to zero. Comparing the control signals in Figure 8.18 and Figure 8.19, we see that the magnitude of cart speed PID control is reduced due to the negative derivative gain. This reduces the correction magnitude of the cart speed, although the control signal of the cart speed still has a phase lead of about 150°.

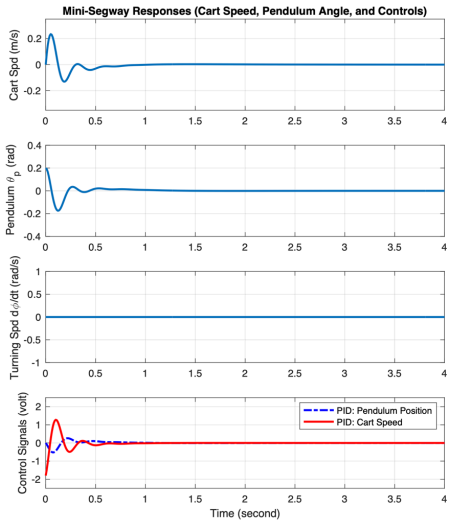


Figure 8.19: SiL simulation with dual-PID control (cart PID derivative gain equal to 0.1).

The third simulation study is for the mini-Segway driving with its pendulum stabilized. Both PID controller gains presented in Table 8.2 were used. However, the derivative gain of the cart speed PID controller was set to 0.1 with initial conditions of pendulum position θ_p equal to $0.2\ (rad)$ and cart speed equal to $0\ (m/s)$. The constant reference block on the right side of Figure 8.17 for the cart speed, named Cart_Spd_Ref, was changed from $0\ (m/s)$ to $0.1\ (m/s)$. Figure 8.20 shows the simulation results of cart driving while stabilizing the mini-Segway pendulum. We can see that the mini-Segway pendulum position converges to zero in about 1.5 seconds and the mini-Segway cart speed converges to its reference of $0.1\ (m/s)$ in about 3.5 seconds. This indicates that the cart dynamics are much slower than the pendulum ones.

Comparing the mini-Segway pendulum position and cart speed responses in Figure 8.19 and Figure 8.20 with the same PID control gains, we can see that when the cart speed reference is not zero, the response convergence time is significantly increased from 0.8 to 3.5 seconds. That is, the coupling cart dynamics between the pendulum position and cart linear motion make it difficult to track the mini-Segway cart speed refence of $0.1\ (m/s)$. Also, from Figure 8.20, we can see that the cart speed PID control converges to a constant value to drive the mini-Segway at a constant speed of $0.1\ (m/s)$. From the top plot of Figure 8.20, we can also observe that the cart speed slowly converges to its target speed of $0.1\ (m/s)$ with almost 100% overshoot with its peak at around one second. This is due mainly to the stabilization of the pendulum. Comparing with the case in Figure 8.19, with zero reference cart speed, the overshoot peak occurs at around 0.05 second.

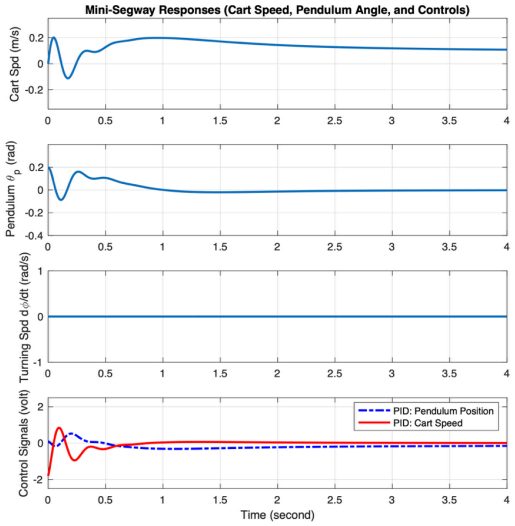


Figure 8.20: SiL simulation with dual-PID control (cart speed reference 0.2 m/s).

The fourth simulation study is for mini-Segway turning with its pendulum stabilized. Here, we used both PID controller gains presented in Table 8.2. However, the derivative gain of the cart speed PID controller was set to 0.1, with the initial conditions of pendulum position θ_p equal to $0.2\ (rad/s)$ and cart speed equal to $0\ (m/s)$. The constant reference block on the middle of Figure 8.17 for the turning speed, labeled Turning_CMD, was changed from 0 to 0.2 volt at time of 1.5 second and changed back to zero at time of 3.0 second. This resulted in turning motion of the mini-Segway between 1.5 and 3.0 seconds. Figure 8.21 shows the simulation results of cart turning while stabilizing the mini-Segway pendulum.

We can see that the mini-Segway pendulum position in Figure 8.21 converges to zero in about 0.8 second, similar to the case shown in Figure 8.18. The mini-Segway cart speed converges to its reference of $0.0\ (\mathrm{m/s})$ in about 0.8 seconds. Figure 8.21 indicates that the mini-Segway starts turning at 1.5 second point with its turning speed increasing gradually from 0 to about $0.3\ (\mathrm{rad/s})$ in about 0.1 second. This is due to acceleration and deceleration motions of the left and right wheels, respectively. The turning moment of inertia is caused by the differential turning input (Turning_CMD) with a magnitude of 0.2 volt. Figure 8.21 indicates that when the mini-Segway is turning, the turning dynamics are independent of the rest of the mini-Segway dynamics (pendulum and cart linear motion). However, for the actual nonlinear dynamics of the mini-Segway (see Section 8.1), the turning motion will affect the pendulum motion when the pendulum angular displacement is not zero.

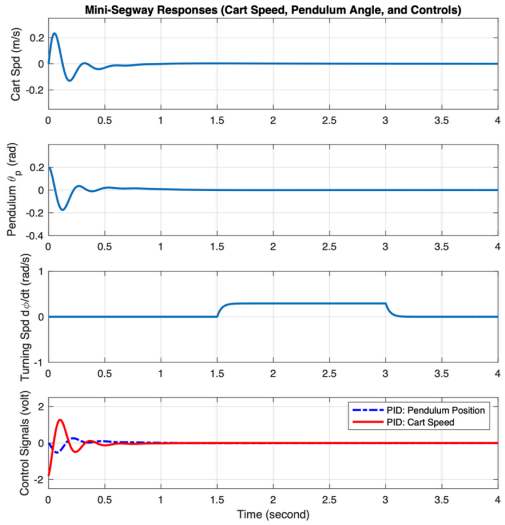


Figure 8.21: SiL simulation with dual-PID control (cart turning).

The fifth simulation study is for the combined simulation with stabilization, driving, and turning. Both PID controller gains presented in *Table 8.2* were used. However, the derivative gain of the cart speed PID controller was set to 0.1 and the initial conditions of pendulum position θ_p equal to $0.2\ (rad/s)$ and cart speed equal to $0\ (m/s)$. Similar to the fourth study, the constant reference block on the middle of *Figure 8.17* for the turning speed, labeled Turning_CMD, was changed from 0 to 0.2 volt at time of the second second and changed back to zero at time of the fourth second. This turns the mini-Segway between the second and fourth second. In addition, the constant reference block on the left of *Figure 8.17* for the reference cart speed, labeled Cart_Spd_Ref, was changed from 0 to $0.1\ (m/s)$ at the time of the third second and changed back to zero at the time of the sixth second. This moves the mini-Segway forward between the third and sixth second.

Figure 8.22 shows the simulation results of cart mixed motion (turning and driving) while stabilizing the mini-Segway pendulum. Once again, the mini-Segway pendulum position converges to zero in about 0.8 second, similar to the case shown in *Figure 8.18*. The mini-Segway cart speed also converges to $0.0\ (m/s)$ in about 0.8 seconds. From *Figure 8.22*, we can see that the mini-Segway starts turning at the second second with its turning speed increasing gradually from 0 to about $0.3\ (rad/s)$ in about 0.1 second. The mini-Segway then moves forward at about the third second because the reference cart speed increased from zero to $0.1\ (m/s)$. We can see that the actual cart speed overshoots a little bit, while the pendulum position leans forward. It then converges to its reference of $0.1\ (m/s)$ while the pendulum position converges within one second. At the sixth second, the cart speed has not converged yet. Recall from Figure 8.20 that it takes more than 3.5 second for the cart speed to converge. When the reference cart speed reduces to $0.0\ (m/s)$ at the sixth second, the cart speed overshoots a little, followed by an undershoot, and then converges to zero, while the pendulum leans backward and then converges back to upright. Again, we can observe that the turning dynamics is independent of the rest of the mini-Segway dynamics (pendulum and cart linear motion).

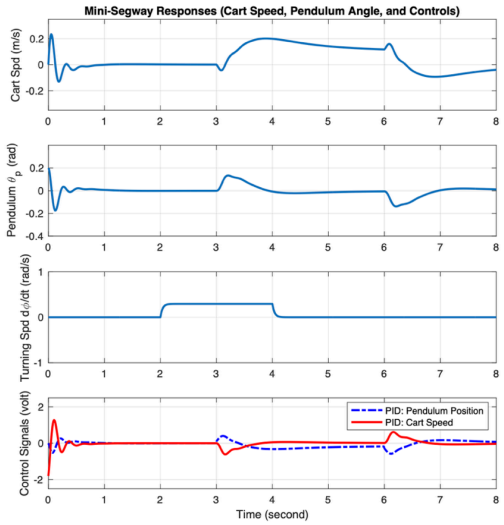


Figure 8.22: SiL simulation with dual-PID control (cart stabilizing, turning, and driving).

However, the cart linear driving motion is heavily coupled with the pendulum motion. This leads to very slow card speed response in order to stabilize the mini-Segway pendulum position. Note that after the reference change at the third and sixth second, control signals generated by both PID controllers are in the opposite direction.

Section Summary

This section develops and validates the dual-PID control system model in Simulink. It uses the linearized mini-Segway six-state state-space model (see Section 8.3) with detailed PID gain selection to achieve the desired performance (pendulum stabilization and cart driving). First, the dual-PID control strategy is tuned and validated for pendulum stabilization with the linearized model. Next, both PID controllers are turned together and validated for simultaneous pendulum stabilizing and cart driving, with additional tuning of the dual-PID control gains for improved performance. The last simulation study is conducted for all three operational modes including pendulum stabilization, straight driving, and turning. The simulation results show that the proposed dual-PID mini-Segway controller is capable of accomplishing all four simulation cases with reasonable response performances.

Section Keywords

Practice Problems

1. For the pendulum stabilization simulation shown in Figure 8.18, what is the pendulum stabilizing time? Select the correct answer.
   a. 0.8 second
   b. 1.2 second
   c. 3.0 second
2. In the simulation study for mini-Segway driving with its pendulum stabilized, select the correct statement from the list below:
   a. The coupled mini-Segway pendulum and cart speed dynamics reduces the response time of both pendulum position and cart speed.
   b. The coupled mini-Segway pendulum and cart speed dynamics keeps the response time of both pendulum position and cart speed unchanged.
   c. The coupled mini-Segway pendulum and cart speed dynamics increases the response time of both pendulum position and cart speed.
3. The simulation study for the mini-Segway turning with its pendulum stabilized indicated that the cart turning and pendulum dynamics are independent. How are the physical system pendulum and the cart turning dynamics related? Select the correct answer.
   a. Always coupled.

   b. Coupled when the pendulum angle is not zero.
   c. Not coupled at any time.
4. Select true or false for the following statement. Due to coupled cart linear motion and pendulum dynamics, a positive derivation gain of the cart speed PID controller with negative proportional and integral gains is able to improve both cart speed and pendulum response significantly.

This page titled 8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers

Learning Objectives

After completing this section, you will be able to:

1. Describe the architecture of the MATLAB and Simulink simulation setup for model-based, state-feedback stabilizing control, with linearized model.
2. Set up the simulation environment in MATLAB and Simulink for model-based tracking control, utilizing linearized model.
3. Develop a simulation environment for model-based tracking control, using the nonlinear mini-Segway system model.

This section conducts three simulation studies. They are a model-based, state-feedback regulation with a linearized model; a model-based, cart speed tracking control (LQR) with a linearized model; and a model-based, state-feedback cart speed tracking (LQTI) simulation with the nonlinear mini-Segway model.

First, we will study case of a model-based, state-feedback regulation with a linearized model, as shown in equations ???, ???, and ???. The simulation study is conducted in MATLAB/Simulink with the Simulink model shown in Figure 8.23. All the components in the Simulink model were introduced in the last section. The state feedback gain $K_p = K_{cp}$ was provided in equation ???.

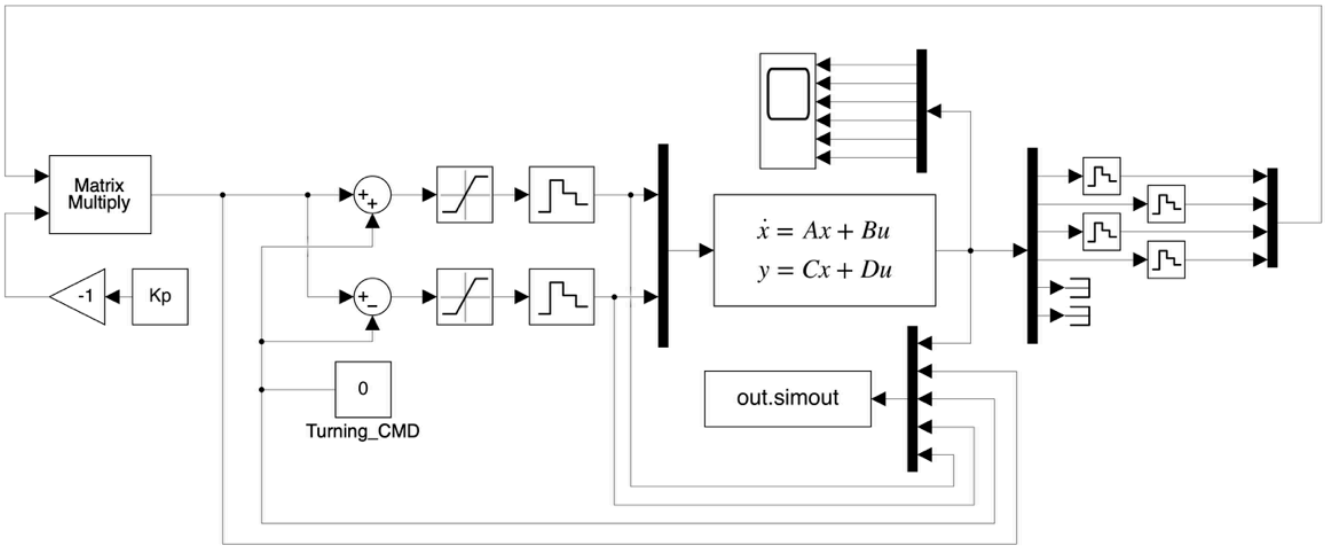


Figure 8.23: Model-based, state-feedback simulation model in Simulink.

Figure 8.24 shows the simulation results with the state feedback gain designed in Section 8.3 and initial conditions of pendulum angle $\theta_p(0) = 0.2\ rad/s$ and cart speed $\dot{x}_{cart}(0) = 0$. We can see that the mini-Segway responses are much faster than these with the dual-PID control, with a response time about 0.2 second. Recall that the dual-PID case is about 0.8 second (see Figure 8.19). This is mainly because the model-based control utilizes all four-state information with consideration of the coupled pendulum and cart speed dynamics. Meanwhile, the dual-PID control uses only the cart speed and pendulum position feedback signal (two states) without considering the mini-Segway coupling dynamics. In Figure 8.19, negative derivative gain is used for the cart speed PID controller to improve coupling compensation. However, the dual-PID control strategy is still much slower than the model-based one due to coordinated control of model-based one.

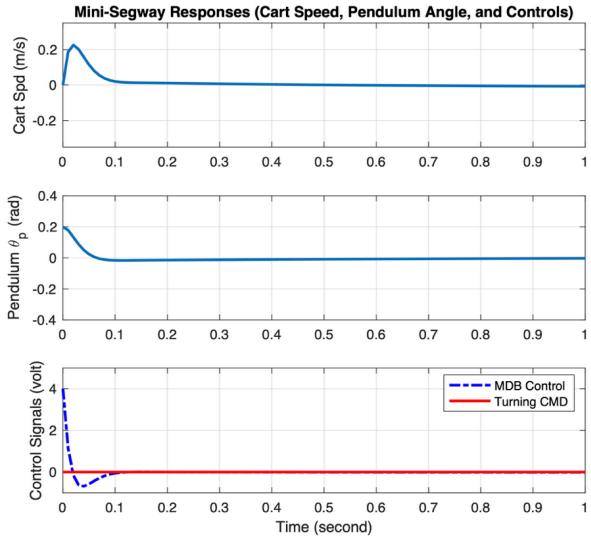


Figure 8.24: Model-based state-feedback regulation with initial pendulum angle $\theta_p = 0.2\ (rad)$.

Next, we will conduct the study with nonzero initial pendulum position $\theta_p(0) = 0.2\ rad/s$ and cart speed $\dot{x}_{cart}\,(0) = 0.2\ m/s$, where the turning command Turning_CMD is also set from zero to 0.2 volt at the second second with model-based state-feedback regulation. The simulation study is conducted with the Simulink model shown in Figure 8.23; all the model components were introduced in the last section.

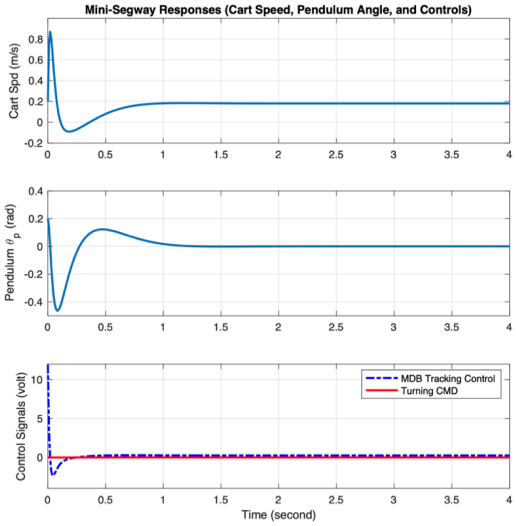


Figure 8.25: Model-based regulation with initial $\theta_p = 0.2\ (rad)$ and $\dot{x}_{cart} = 0.2\ (m/s)$.

We can see the simulation results shown in Figure 8.25; it has state feedback gain from equation ??? in Section 8.3 and initial conditions of pendulum angle $\theta_p = 0.2\ (\text{rad})$ and cart speed $\dot{x}_{cart} = 0.2\ (\text{m/s})$. The mini-Segway responses in this case are much slower than the case with only one nonzero initial condition of pendulum angle $\theta_p = 0.2\ (\text{rad})$, which takes about 1 second for all states to go to zero. The responses in Figure 8.24 take only about 0.2 second to converge. This is mainly due to the coupled pendulum and cart speed dynamics, which slows down convergent speed.

Now we will use simulation to investigate a model-based, cart speed tracking controller with a linearized model. We will use the controller from equation ??? designed in Section 8.3. The state feedback regulation controller from equation ??? only stabilizes the mini-Segway. but it cannot be used to drive it. However, the controller designed in equation ??? with cart speed error integration can. In fact, controller ???, which utilizes the control system architecture in Figure 8.13, has the same functionalities

as these of dual-PID one. Figure 8.26 shows the Simulink model that implements the integral tracking control architecture from Figure 8.13. Here, the digital error integration is assumed to be calculated within the target microcontroller as a virtual state with coefficient $K = 0$. Control gain K_{cpi} in equation ??? is used in Figure 8.26 as K_{pi}.

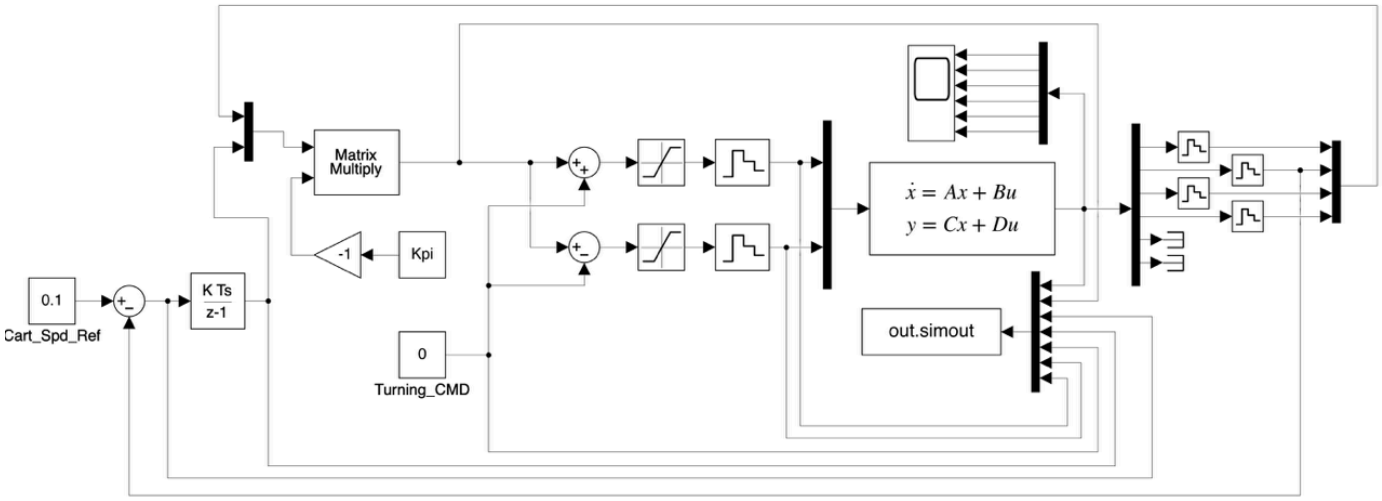


Figure 8.26: Model-based cart speed tracking in Simulink model.

We conduct our first simulation for this controller with a cart reference speed of $0.2\ (\mathrm{m/s})$, initial pendulum position of $\theta_p = 0.2\ (\mathrm{rad})$ and cart speed of $\dot{x}_{cart} = 0.2\ (\mathrm{m/s})$. Figure 8.27 shows the simulation results.

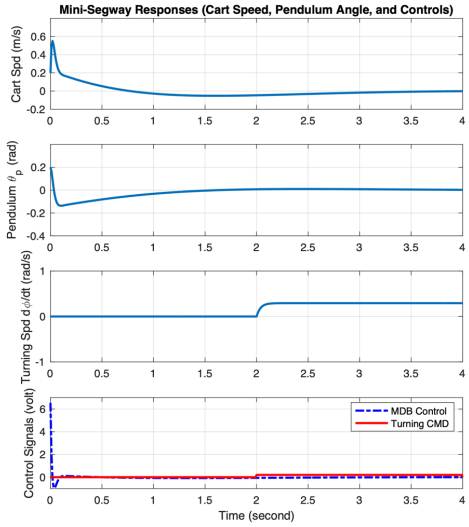


Figure 8.27: Model-based tracking control with initial $\theta_p = 0.2\ (rad)$ and $\dot{x}_{cart} = 0.2\ (m/s)$.

Compare the responses in Figure 8.27 with those in Figure 8.25. We can see that with a nonzero cart speed and pendulum position references (Cart_Spd_Ref $= 0.2\ m/s$), system responses of the mini-Segway cart speed and pendulum position converged to zero with a little bit over one second from its initial condition of cart speed $\dot{x}_{cart}(0) = 0.2\ (m/s)$ and pendulum position $\theta_p(0) = 0.2\ (rad)$. This is very close to the regulation controller response shown in Figure 8.25. This response can be attributed to the same weight coefficients for cart position and speed, and pendulum position and speed used for designing the LQR controller. The only difference between the LQR and LQTI controller designs is the added cart speed error integration state, x_{ce}, with the corresponding design weighting $q_{55} = 10^4$. As a result, we can conclude that the LQTI design did not change the response convergence performance with additional tracking capability. At the second second, the turning command becomes 0.2 volt, leading to mini-Segway turning at a speed close to $0.3\ (rad/s)$. From Figure 8.27, we can observe that the mini-Segway turning dynamics are independent of its pendulum and cart motion dynamics with the linearized evaluation model. As a result,

state feedback uses only these pendulum and cart motion states for feedback control.

Now, we are ready to conduct a composite simulation with nonzero state initial conditions ($\theta_p(0) = 0.2\ (rad)$ and $\dot{x}_{cart}(0) = 0.2\ (m/s)$), zero reference signal for pendulum angle.

During the simulation, the mini-Segway cart speed reference Cart_Spd_Ref was changed from zero to $0.1\ (m/s)$ at the second second, and then back to zero at the fourth second, where it stayed for the rest of the simulation. The turning command Turning_CMD was changed from zero to 0.4 volt at the third second and remained at 0.4 volt until to the end of simulation. From the simulation results shown in Figure 8.28, we can see that with zero mini-Segway cart reference speed, it takes about one second for both the pendulum angle and cart speed to converge back to zero, where control voltage is used initially to get cart speed to close to zero while the pendulum position has a large negative oscillation before converging to zero.

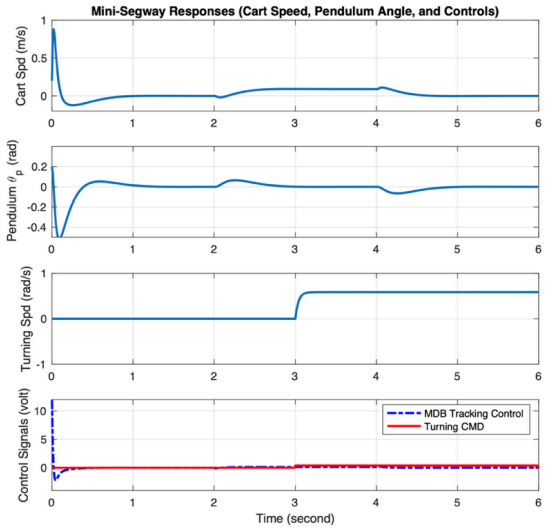


Figure 8.28: Model-based tracking control with turning, initial $\theta_p = 0.2\ (rad)\ and\ \dot{x}_{cart} = 0.2\ (m/s)$.

With the step cart-speed reference at the second second, it takes about 0.8 second for the pendulum position to converge to zero and for the cart speed to track its reference one at 0.1 (m/s). Last, at the third second, the turning command of 0.4 volt was applied; the mini-Segway started turning and converged to a turning speed of about $0.6\ (rad/s)$ with a response time of about 0.1 second. We can see that the turning motion does not affect the pendulum and cart speed motion. This is the main reason that both LQR and LQTI control designs in Section 8.4 can use the reduced-order model with the turning motion dynamics deleted. It simplifies the control design and results in a reduced dimension state feedback controller.

Next, use simulation to investigate the nonlinear Simulink plant model developed in Section 8.1 (see Figure 8.9). Here, we will use the cart speed tracking (LQTI) controller from equation ???, designed in Section 8.3. We will conduct two simulation studies. One studies the difference between the simulation results of linear and nonlinear models without the effect of Coulomb friction on both wheels. The other investigates the effect of the Coulomb friction with a nonlinear friction model (see Figure 8.8).

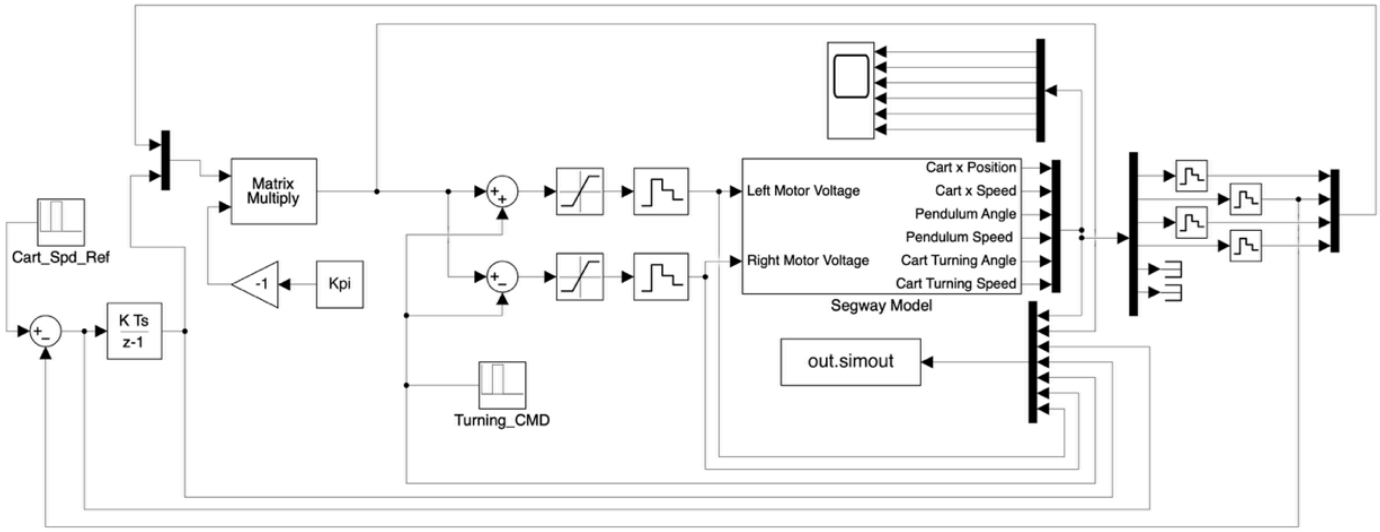


Figure 8.29: Model-based cart speed tracking with a nonlinear Simulink model (Figure 8.8).

We can see in Figure 8.29 that the simulation environment with a nonlinear Simulink mini-Segway model is similar to the one with the linear Simulink model shown in Figure 8.26. The only difference is that the linearized model was replaced by the nonlinear Simulink model shown in Figure 8.8. As a result, no further discussion on the simulation environment will be provided. For the first simulation, we can compare the nonlinear simulation results (see Figure 8.30) with the linear results (see Figure 8.28). The simulation was conducted with the initial mini-Segway position of $\theta_p(0) = 0.2\ (rad)$ and cart speed of $\dot{x}_{cart}(0) = 0.2\ (m/s)$; reference cart speed varying from zero to $0.1\ (m/s)$ at the second second and from $0.1\ (m/s)$ back to zero at the fourth second; and cart turning changing from zero to $0.1\ (volt)$ at the third second and from $0.1\ (m/s)$ back to zero at the sixth second. The simulation environment is exactly the same as the results shown in Figure 8.28, but with the linear model replaced by the nonlinear one.

The nonlinear mini-Segway simulation results are shown in Figure 8.30 with cart speed, pendulum position, turning speed, and control inputs. Comparing these nonlinear results with the linear results from Figure 8.28, we can see that, with the same initial condition, the results are similar, with both pendulum angle and cart speed converging to zero within one second. At the second second, the cart reference speed changed from zero to $0.1\ (m/s)$ and the cart speed converged to the reference value in about 0.6 second, similar to the linear case. However, the cart speed response of the nonlinear simulation is not as smooth. The main difference between linear and nonlinear simulation results occurs when the turning command of $0.1\ (volt)$ is applied at the third second; the convergence rate is much slower than the linear case, but with larger turning speed.

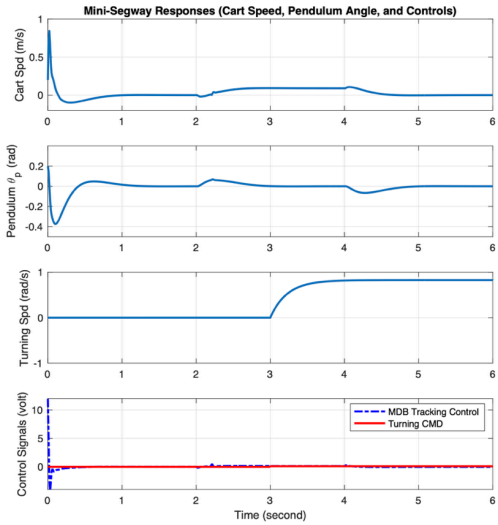


Figure 8.30: Model-based tracking control under the same conditions as Figure 8.28 with nonlinear model.

Figure 8.31 shows the nonlinear simulation results, with the Coulomb friction block activated [32] (see Figure 8.8). In this simulation study the cart speed reference and turning command are the same as those in the previous study, along with the initial condition. With initial condition of the mini-Segway position of $\theta_p(0) = 0.2\ (rad)$ and cart speed of $\dot{x}_{cart}(0) = 0.2\ (m/s)$, both the cart speed and pendulum position converge to zero in one second. This is similar to both linear and previous nonlinear cases.

When the cart reference speed changed to $0.1\ (m/s)$ from zero at the second second, the cart speed converged to the reference value in about 0.6 second smoothly. This is similar to the linear and nonlinear without Coulomb friction cases. However, the cart speed response of the nonlinear simulation with Coulomb friction is much smoother than the previous case without Coulomb friction. When the turning command of $0.1\ (volt)$ is applied at the third second, we can see that the convergence rate of the turning speed is similar to the previous case without Coulomb friction. However, the coupling effect of cart speed and turning at the fourth second occurred with a slight dip of turning speed when the reference cart speed is set to zero from $0.1\ (m/s)$. This is caused mainly by the added Coulomb friction. In fact, this effect can also be seen in the experiments.

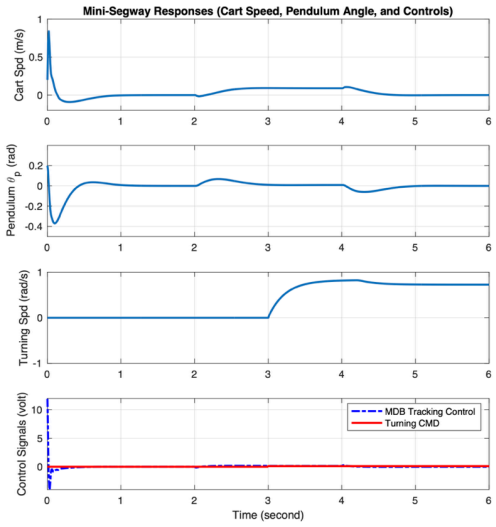


Figure 8.31: Model-based tracking control under the same conditions as Figure 8.30 with Coulomb friction.

Figure 8.32 compares the raw left wheel driving force with and without Coulomb frication modification. There is a certain difference between the unmodified and modified (with Coulomb friction activated) forces. Because the left and right wheel driving forces could be modified in slightly different ways due to the turning action, the mini-Segway turning motion could be affected by the cart speed variation.

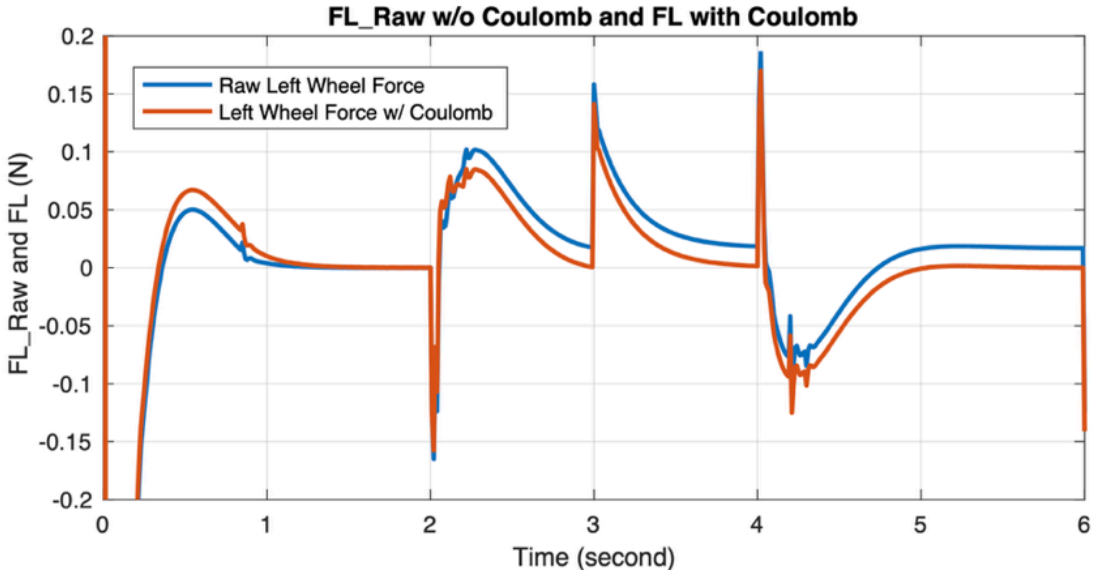


Figure 8.32: Left wheel forces with and without Coulomb friction.

Section Summary

This section conducts model-based simulations utilizing the model-based controller designed in Section 8.3. First, it investigates the LQR control strategy for state-feedback stabilization, utilizing mini-Segway cart position and speed, and pendulum position and speed. Then it studies state-feedback with integral cart speed tracking control, incorporating feedback of mini-Segway cart position and speed, pendulum position and speed states, and the tracking error state calculated internally in the LQTI controller. Last, the section investigates the model-based state-feedback LQTI controller designed in Section 8.3 in simulation with the nonlinear mini-Segway model developed in Section 8.1, with the added Coulomb friction model. For all the simulations, the state feedback controller is sampled at 100 Hz.

Section Keywords

Practice Problems

1. Select one correct statement from the following list given the simulation initial condition of pendulum position of $\theta_p(0) = 0.2\ (rad)$ and cart speed of $\dot{x}_{cart} = 0.2\ (m/s)$.
   a. The model-based simulation results in Figure 8.28 indicated that the cart speed and pendulum position states converge to zero in about one second for both dual-PID and model-based LQTI controllers.
   b. The dual-PID control simulation results in Figure 8.21 indicated that the cart speed and pendulum position states converge to zero in about one second for both dual-PID and model-based LQTI controllers.
2. Discuss why two reference signals (cart speed and pendulum position) are needed for the dual-PID control while only one reference signal (cart speed) is required for the model-based LQTI control.
3. Compare the simulation results in Figure 8.28 and Figure 8.31 by utilizing the LQTI controller in equation ??? for linear and nonlinear mini-Segway models, respectively. Select one correct statement from the following list.
   a. The linear simulation results in Figure 8.28 indicated that the mini-Segway turning motion can be affected by its driving motion.
   b. The nonlinear simulation results in Figure 8.31 indicated that the mini-Segway turning motion can be affected by its driving motion.
   c. The nonlinear simulation results in Figure 8.31 indicated that the mini-Segway turning motion cannot be affected by its driving motion.
4. Design an LQTI controller with $Q_{cpi} = diag\ \left[0.1,\ 10,\ 100,\ 0.1,\ 10^4\right]$ and $R_{cpi} = 1$ utilizing the system model matrices defined in ???. Conduct a closed-loop simulation using the Simulink diagram in Figure 8.29. Then compare the simulation results with that from the LQTI controller designed using $Q_{cpi} = diag\ \left[1,\ 1000,\ 100,\ 0.1,\ 10^4\right]$ and $R_{cpi} = 1$.

This page titled 8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8.7: Experimental Study of the Mini-Segway Model-Free Control

Learning Objectives

After completing this section, you will be able to:

1. Explain the hardware architecture of the mini-Segway system.
2. Describe the Simulink-based software architecture of the mini-Segway system, along with the signal processing of pendulum angle and cart speed.
3. Conduct the experimental validation of the dual-PID control strategy.

This section focuses on experimental studies of the dual-PID control strategy. First, however, we discuss the mini-Segway system hardware architecture along with its sensors and actuators used to generate feedback control signals and to drive the mini-Segway, respectively. Next, we introduce the dual-PID real-time control software in MATLAB/Simulink, along with the wheel speed signal processing and motor driving software. Last, we introduce the overall software architecture to clarify that the Simulink model of dual-PID control strategy is running within the host computer. Also, the Arduino MKR 1000 Wi-Fi microcontroller collects the sensor signals and executes the motor drive commands. Now, let's discuss the hardware architecture of the mini-Segway.

Figure 8.33 shows a mini-Segway picture and its hardware structural diagram. The mini-Segway has two main sensor sources. These include hall-effect encodes mounted on both Segway wheels through gear boxes (see Section 6.2) and MPU 6050 three-axis accelerometer and rate-gyro MEMS sensors mounted on the mini-Segway pendulum. Two DC motors (see Section 5.2) drive the mini-Segway. The two motors spin in the opposite direction when the mini-Segway drives in a straight line. An MKR motor carrier board (see Section 7.4) drives both DC motors utilizing the on-board H-bridge PWM drive circuits (see Section 5.3) and to collect pulse counts for both hall-effect encoders (see Section 6.2) mounted on the motor/wheel axes. The Arduino MKR 1000 Wi-Fi board can be inserted onto the MKR motor shield board directly without manual connection required.

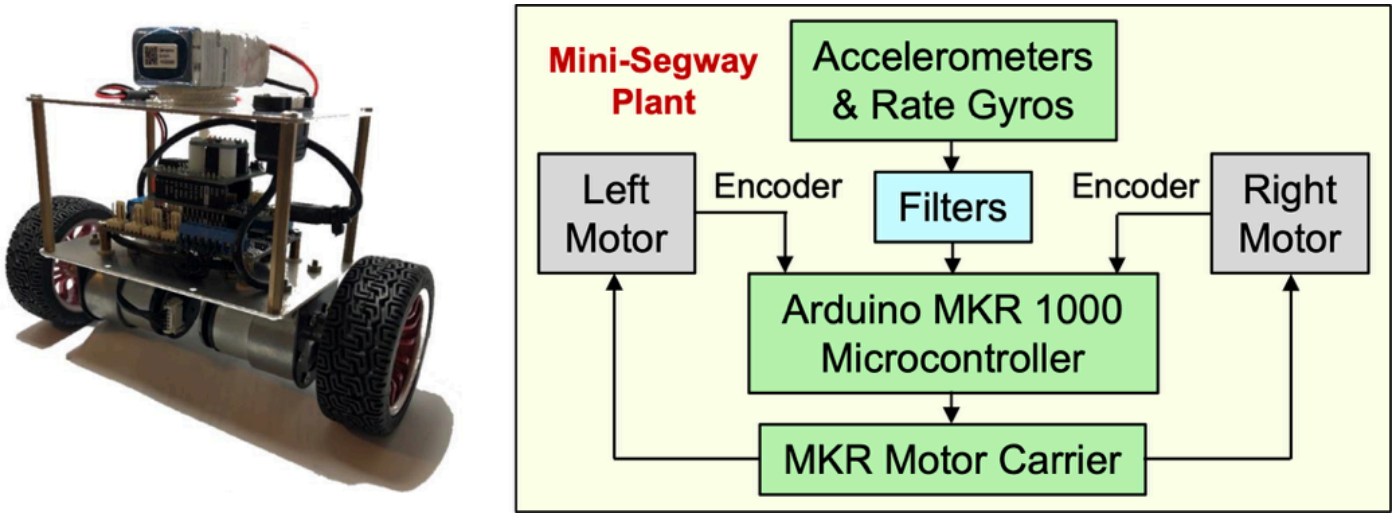


Figure 8.33: Mini-Segway hardware architecture.

The control hardware of the mini-Segway consists of the Arduino MKR 1000 Wi-Fi microcontroller board. It reads MPU 6050 three-axis acceleration and rotational speed signals though the I2C serial bus and also the wheel rotational speed signals from the Arduino MKR motor carrier board through I2C. Two motor control commands in PWM duty-cycle are also sent through the I2C link. We will discuss later how the Arduino MKR 1000 Wi-Fi microcontroller constantly communicates with it host computer.

Figure 8.34 shows the electronic hardware architecture with key connections. The connections from the MKR motor carrier board to both DC motors and to the Arduino MKR 100 Wi-Fi board are the same as those in Figure 7.30. The MPU 6050 three-axis accelerometer and rate-gyro are the additional hardware in Figure 8.34. In order to connect this sensor board to the Wi-Fi board,

a custom interface board (see the left-most picture) was made to slide the interface board into the Wi-Fi board; the MPU 6050 can be plugged into the interface board. The interface and MPU 6050 boards are powered by the MKR motor carrier board and communicate with the Wi-Fi board through a I2C bus.

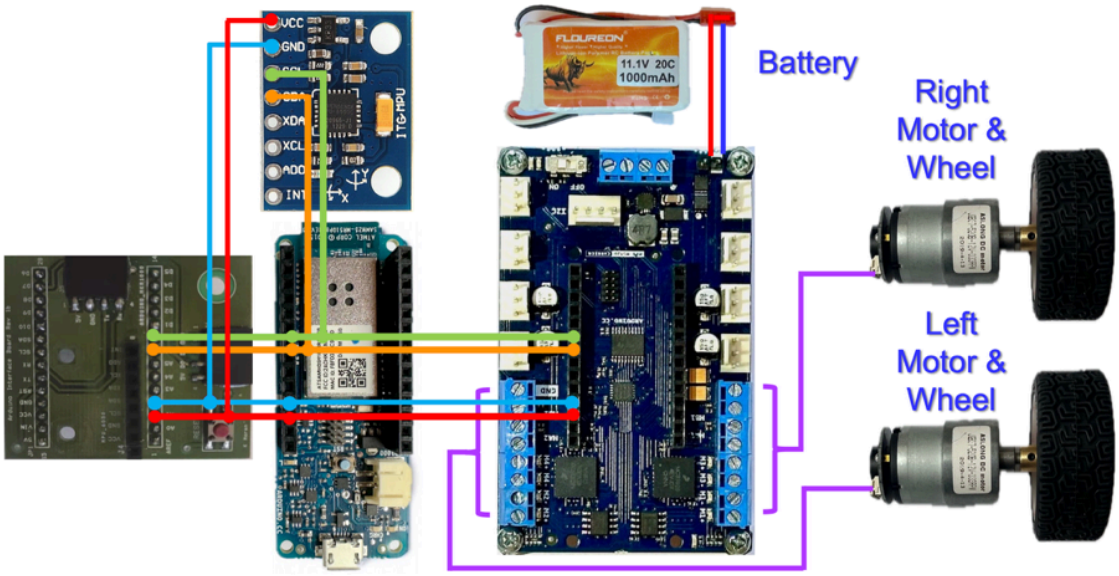


Figure 8.34: Mini-Segway electronic hardware architecture.

Figure 8.35 shows a top layer Simulink implementation of the mini-Segway real-time control strategy utilizing the dual-PID control architecture. This figure shows six blocks of the control strategies, including a) PID controller for stabilization (pendulum position); b) PID controller for driving; c) Segway protection logics, turning, and motor drives; d) cart speed and displacement calculation; e) pendulum angular position calculation; and f) driving pushing bottoms.

Figure 8.36 shows the details of a Simulink model for a PID controller used to regulate the pendulum position regulation, also part a) in Figure 8.35. Instead of using an anti-windup PID controller, it uses an enabled PID controller to prevent the integration windup and to ensure the integration value is zero at the startup. The PID controller Simulink block is shown in the left graph and the enabled PID controller is shown in the right graph. From the top of the left graph, we can see that the pendulum PID controller is enabled only when the pendulum angle is less than 45°. This prevents the wheel spinning when the mini-Segway is laid down. In addition, the pendulum position control output will be set to zero when the pendulum angle is greater than 45°.

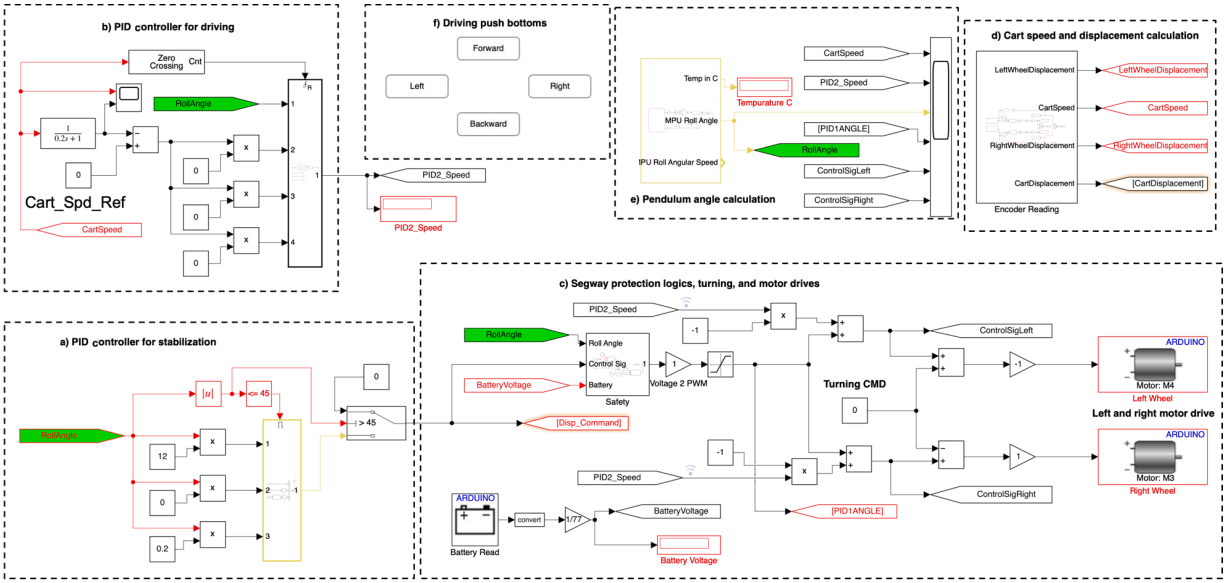


Figure 8.35: Top layer Simulink realization of dual-PID real-time control strategy.

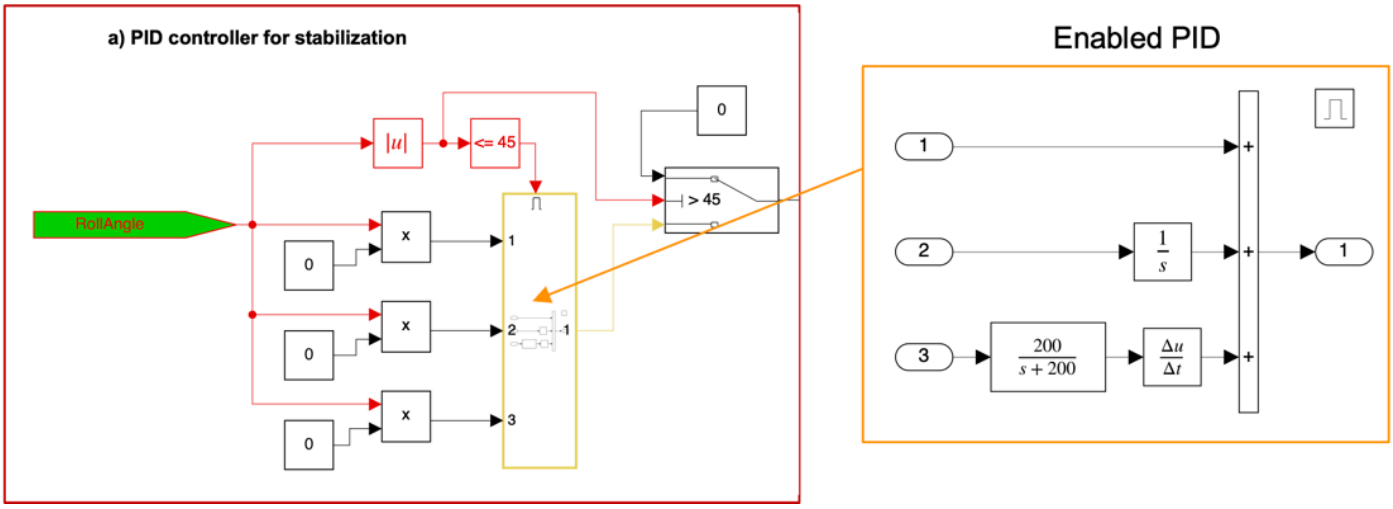


Figure 8.36: PID controller for mini-Segway stabilization (pendulum position).

Figure 8.37 shows the details of the Simulink model of a PID controller for cart driving. Instead of using an anti-windup PID controller, it uses an enabled PID controller (yellow block on the right graph) to prevent the integration windup and to ensure the integration value is zero at the startup. In addition, when the pendulum angle is greater than 45°, the cart driving controller output is set to zero.

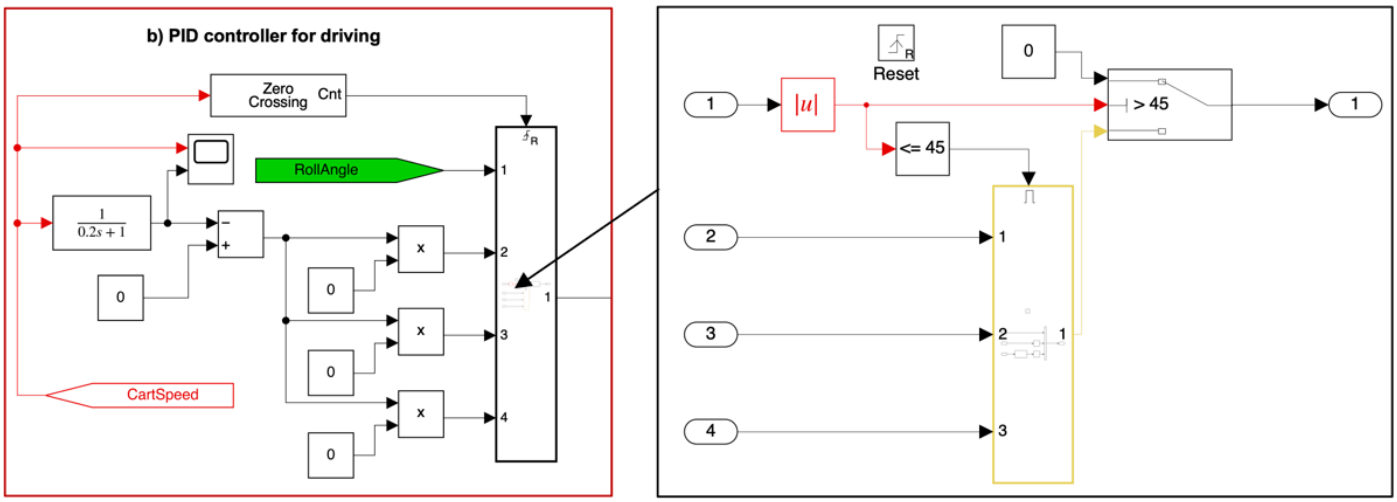


Figure 8.37: Mini-Segway cart driving PID control.

The PID controller Simulink block diagram is shown on the left graph as a trigged PID controller in Figure 8.37. From the top of the left graph, we can see that the cart driving PID controller is trigged by the zero-crossing detection so that the PID integration will be reset when a zero-crossing is detected. In addition, a low-pass filter with time constant of 0.2 second (0.6 second 5% settling time) is used to filter the cart speed feedback signal. Both PID controller gains shown are zeros. We will address the gain turning later during the experimental study.

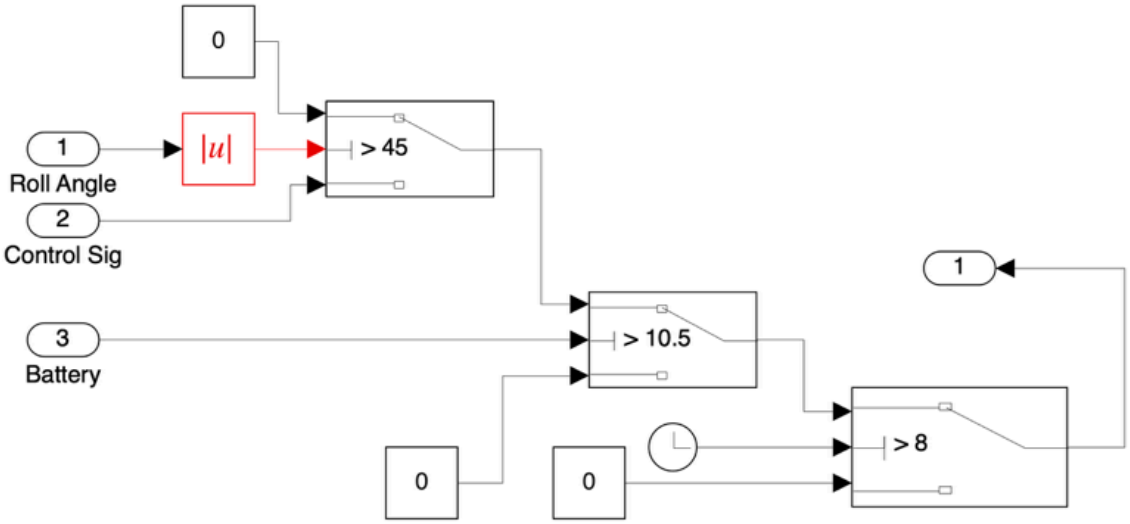


Figure 8.38: Mini-Segway Simulink safety logics.

Next, we will look at part c), the Simulink mini-Segway control diagram for Segway protection logics, turning, and motor drives (the lower-right diagram in Figure 8.35). The pendulum angle stabilization is activated only when the pendulum angle is less than 45° and the battery voltage is greater than 10.5 volts. In addition, it will take eight seconds to activate the pendulum stabilization (see Figure 8.38). These features are mainly for protecting the mini-Segway from activation motor control logics.

From Figure 8.35, we can see that the control signals of the pendulum angle and cart driving PID control outputs are in an opposite direction with the pendulum angle control signal multiplied by negative one, which can be interpreted directly. When the cart is driving in the forward direction with both motors running in the forward direction, the pendulum will lean backward. As a result, the pendulum stabilization requires both motors to run in the backward direction. The turning command is added after both pendulum stabilization and cart driving signals are combined. The positive and negative gains of one are used to ensure both mini-Segway wheels rotate in the opposite direction when the turning command is zero. Both left and right motor blocks are associated with the Simulink Arduino software for motor drives utilizing the Arduino MKR 100 Wi-Fi board and the MKR motor carrier board. The associated software will be executed in the Arduino MKR 1000 Wi-Fi board while the rest of software is running within the host computer.

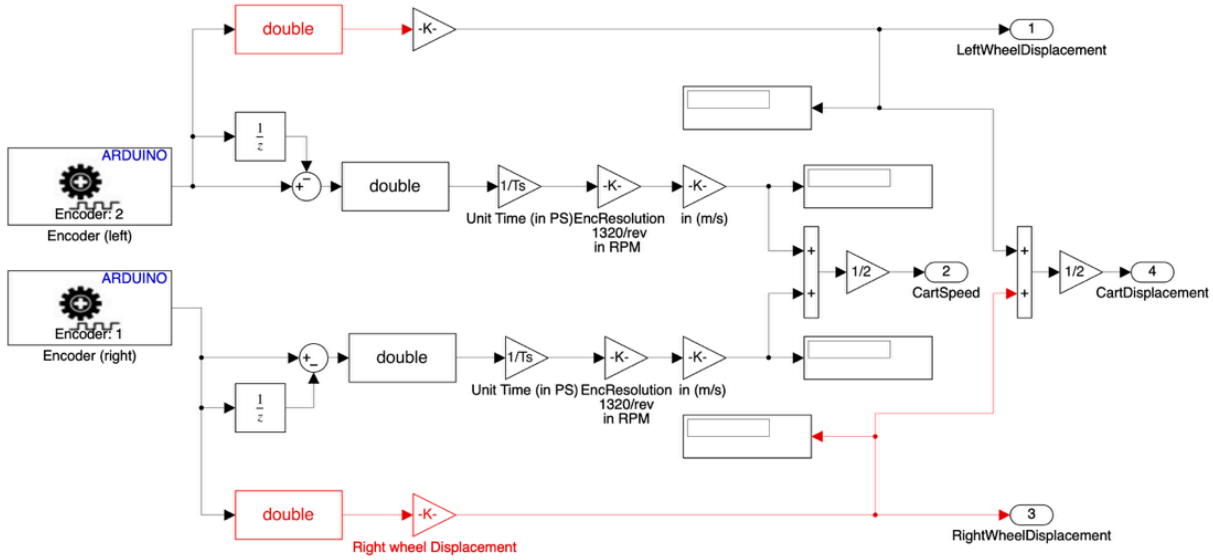


Figure 8.39: Mini-Segway cart speed and displacement calculation block.

Figure 8.39 shows part d), the mini-Segway for cart speed and displacement calculations with details. The Simulink block is located in the upper-right corner of Figure 8.35. We can see that the two most left blocks are the software that reads the left and

right motor encoders; their outputs are the pulse counts of the left and right wheel hall-effect sensors. The top and bottom block rows convert the pulse counts to the associated wheel linear displacement with a gain $k = 2\pi \times 0.034/1320$. The cart displacement is the average of the two-wheel linear displacement signals. Note that $0.034\ (m)$ is the radius of the mini-Segway wheel and 1320 is the number of pulses per wheel revolution. The middle two rows in Figure 8.39 calculate the individual wheel linear speed signals. The cart speed is the average of the two-wheel linear speed signals. The two gains in the middle two rows are defined in Section 6.2, with the sample period of $T_s = 0.01$. second. The conversion gain from pulse per second (PS) to revolution per minute (RPM) is $k = 60/1320$, The conversion gain from revolution per minute to linear wheel speed in (m/s) is $k = 2\pi \times 0.034/60$. Similar to the case of the motor drive blocks in Figure 8.38, the hall-effect sensor pulse counting software in Figure 8.39 is executed within the motor carrier board while the rest of software is running within the host computer.

Figure 8.40 shows the calculation for mini-Segway pendulum displacement (angle), where the control block for part e) of the Simulink simulation is located in the upper-middle part of Figure 8.35. We can see that the left-most block represents the software associated with the MEMS sensor MPU 6050, which provides the three-axis acceleration and rotational angle signals for the mini-Segway pendulum. This part of the software is executed in MPU 6050 and the Arduino MKR 1000 Wi-Fi board. The MPU 6050 block has three output groups: three-axis acceleration signals, three-axis rational angular rates, and the MPU sensor temperature for compensation. The pendulum angular position is calculated based upon the sensor fusion method introduced in Section 6.5 (see the top two rows in Figure 8.40). Pendulum angular speed and MPU 6050 temperature are the direct outputs of the MPU 6050.

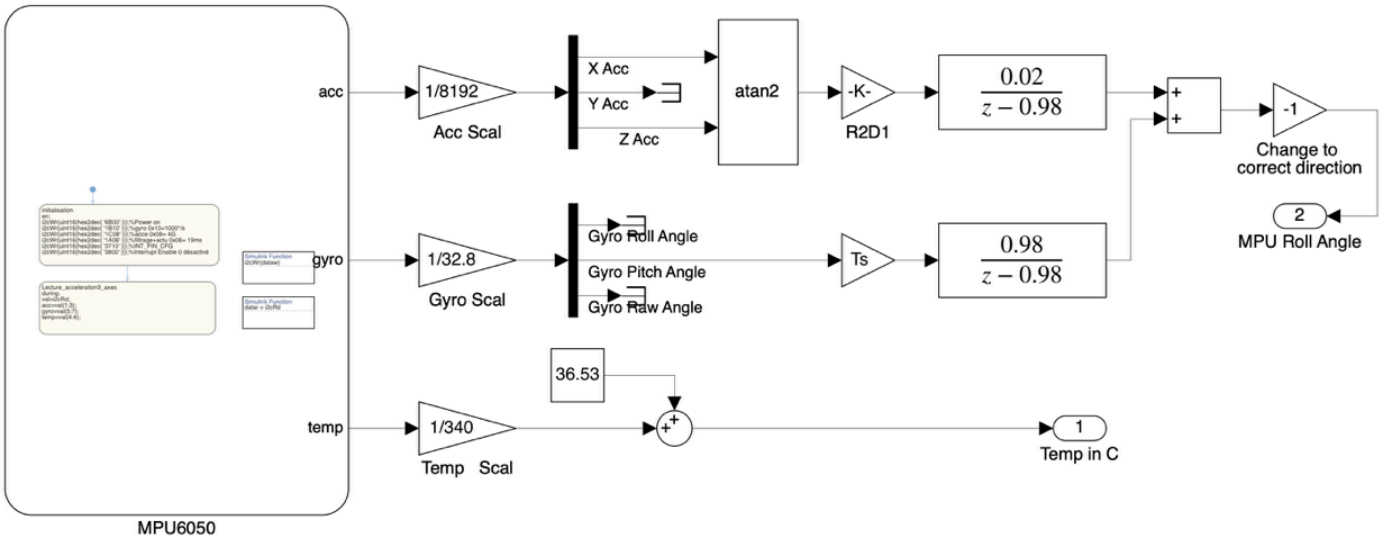


Figure 8.40: Mini-Segway pendulum angle (position) calculation.

Part f) of the mini-Segway software is for the control buttons shown on Figure 8.35. The forward and backward bottoms change the Cart_Sped_Ref value from zero to a calibrated value of $0.1\ (m/s)$ for forward driving and $-0.1\ (m/s)$ for backward driving. The left and right bottoms change the Turning_CMD value from zero to a calibrated value of $25\ (count)$ for left turn and $-25\ (count)$ for right turn. The maximum count number for motor PWM input is 255, and $25\ (count)$ is equivalent to about 1.1 (volts), assuming a battery voltage of 11.5 volts.

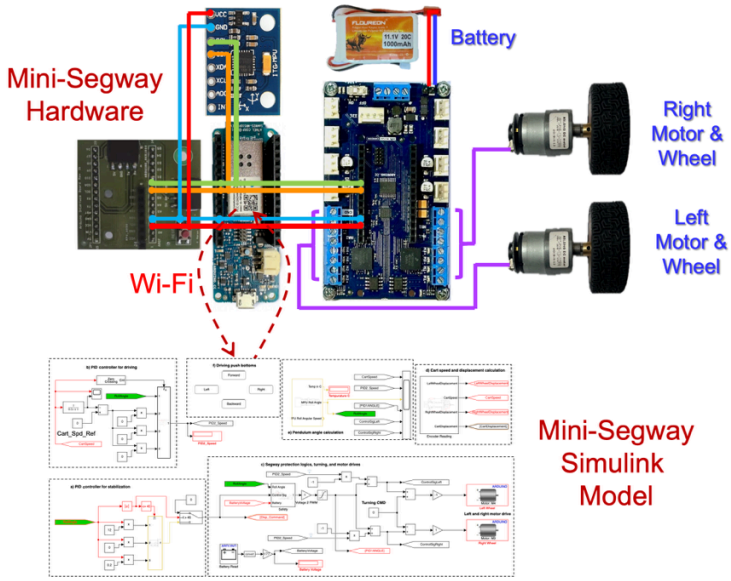


Figure 8.41: Mini-Segway software architecture.

Now, let's discuss the software structure of the mini-Segway operated under the operational mode of XCP on Wi-Fi. Here, most of the Simulink blocks in the real-time control model shown on Figure 8.35 are executed by the real-time Simulink software in the host computer. Only these blocks associated with left and right motors, left and right encoders, and MPU 6050 are running on the mini-Segway hardware. The motor control commands are sent from the real-time Simulink software to the Arduino MKR 100 board through Wi-Fi communication. They are then sent to the MKR motor carrier board through the I2C serial link within the mini-Segway. Meanwhile, the sampled left and right encoder counts and the MPU 6050 signals are sent to the Simulink software through the Wi-Fi communication from Arduino MKR 1000 Wi-Fi to the host computer. The mini-Segway cart speed and pendulum position are processed for feedback control in the host computer. Figure 8.41 demonstrates these relationships.

The mini-Segway control software is implemented in MATLAB/Simulink. See Figure 8.35 for Simulink implementation details, along with the detailed subsystem diagrams discussed above. We will now conduct two experimental studies: pendulum stabilization and cart driving.

The first study is for pendulum stabilization. After carefully tuning the stabilization PID control gains; see the Simulink control diagram shown in Figure 8.36 for detail. Table 8.3 lists the tuned gains. The integral gain is intentionally set to zero to avoid integration windup. Only one of the PID integral gains for pendulum stabilization and cart driving PID controllers can be nonzero; see Section 8.5 for details.

Table 8.3: Tuned pendulum stabilization PID gains.

Proportional gain	Integral gain	Derivative gain
12	0	0.8

These PID proportional and derivative control gains are much larger than these used in the simulation; see Table 8.2. Note that the control input to the Simulink DC-motor block has a range between -255 and 255 (PWM duty-cycle count with ± 255 corresponding to $\pm 100\%$ duty-cycle) whereas in the simulation, the input to the DC motor in Section 8.5 is armature voltage of ± 12 volts. Assuming that the battery voltage is 11.5 volts, there is a conversion ratio of $k_{cv} = 255/11.5 = 22.17$. In addition, the angular displacement unit of implemented experimental software is in degree and radius for simulations. There is a conversion ratio of $k_{dr} = \pi/180 = 0.01745$. With a combined conversion ratio of $k_{cv}k_{dr} = 0.387$, the tuned proportional gain [GrindEQ__12_] is close to converted simulation gain of $3/0.387 = 7.75$, which is about one and half times larger than the simulation one. The derivative gain is about seven times smaller, which could be due to the unconsidered nonlinearity (such as Coulomb friction) and sensor noises in the actual system, which prevents using high derivative gain for the pendulum stabilization.

Figure 8.42 shows the stabilization control responses. The left column of Figure 8.42 shows the mini-Segway cart speed, speed control count, and pendulum position (angle) from top to bottom. The right column shows the mini-Segway pendulum position control count, and left and right motor total control counts from top to bottom. This is a stabilization experiment, so the reference cart speed is always zero. From the bottom-left pendulum position plot, the mini-Segway pendulum was lifted to the vertical

stabilization slightly before the 13th second. When the pendulum angle θ_p is reduced to 45°, the Segway control is activated. It takes about two seconds (from 12.8 to 14.8 second) for the mini-Segway to stabilize and remain in the upright position with θ_p close to zero. However, after the pendulum angle is stabilized, the cart remains moving for an additional two seconds until the 17th second. This can be observed from the cart speed plot in the top-left of Figure 8.42. After the 17th second, both the cart speed and pendulum position remain close to zero. The associated motor control for both left and right motor control counts are close to zero (see the middle and bottom plots on the right of Figure 8.42).

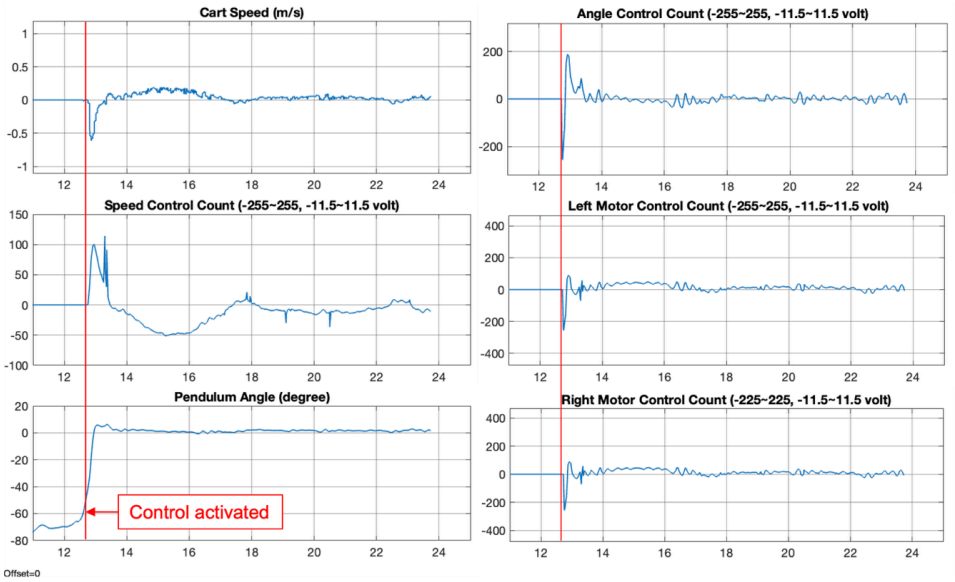


Figure 8.42: Mini-Segway stabilization responses.

The second study is for simultaneous cart driving and pendulum stabilization. After careful tuning, the cart driving PID control gains with the same stabilization gains; see the Simulink control diagram in Figure 8.37. The tuned cart driving gains are listed in Table 8.4. Normally an iterative tuning process is required due to coupled dynamics. In addition, the derivative gain has an opposite sign to proportional and integral gains for improved performance; see the result of dual-PID simulation results in Section 8.5.

Table 8.4: Tuned cart driving PID gains.

Proportional gain	Integral gain	Derivative gain
320	10	-2

The PID control gains seen in *Table 8.4* are much larger than those used in the simulation study (see *Table 8.2*) in Section 8.5, where the equivalent proportional gain is $300/22.17 = 13.53$. After considering the conversion ratio k_{cv}, the tuned proportional and integral gains are about four and half times greater than the simulation ones; and derivative gain is about the same as the simulation one. Similarly, this could be due to the unconsidered nonlinearity (such as Coulomb friction), modeling error, and PWM conversion error (especially at low duty-cycle level) in the actual system.

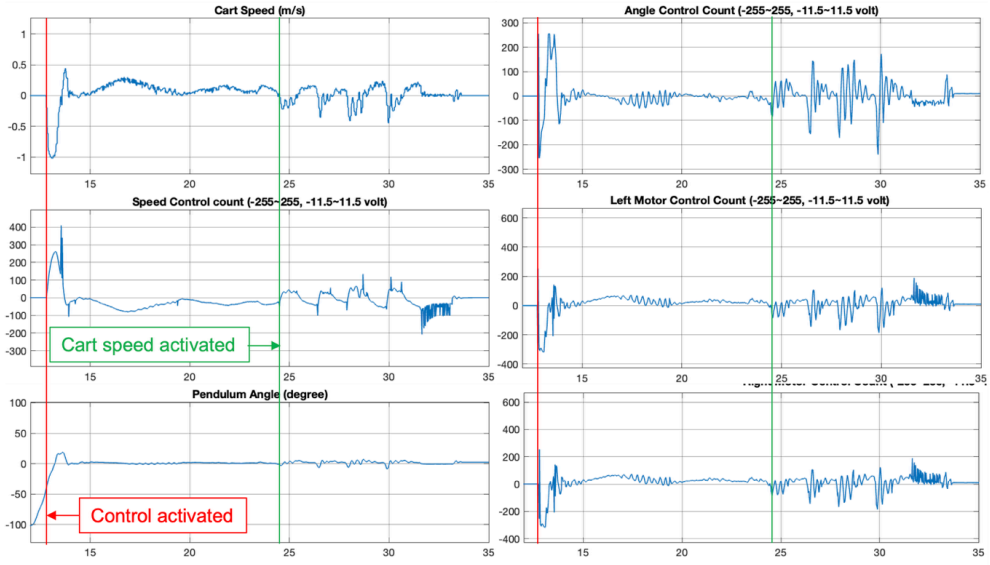


Figure 8.43: Mini-Segway cart driving responses.

Figure 8.43 shows the cart driving control responses with stabilization of pendulum position. The left column shows the mini-Segway cart speed, speed control count, and pendulum position (angle) from top to bottom. The right column shows the mini-Segway pendulum position control count, left and right motor total control counts from top to bottom. This is a stabilization and cart driving experiment. From the bottom-left pendulum position plot, the mini-Segway pendulum was lifted to the vertical stabilization slightly before the 12th second. When the pendulum angle θ_p is reduced to 45° (marked by the red vertical line), the segway control is activated. It takes about one second (from 13 to 14 second) for the mini-Segway to stabilize and remain in the upright position with θ_p close to zero with the cart speed slightly above zero. The cart reference speed changes from zero to $0.1\ (m/s)$ at around 24.5th second, from $0.1\ (m/s)$ to zero at around 25.5th second, from zero to $0.1\ (m/s)$ at around 26.5th second, from $0.1\ (m/s)$ to zero at around 27.2nd second, from zero to $0.1\ (m/s)$ at around 28.0th second, from $0.1\ (m/s)$ to zero at around 29.0th second, and from zero to $0.1\ (m/s)$ at around 29.8th second, from $0.1\ (m/s)$ to zero at around 31.8th second. For all four cart-speed activations, the cart speed control count increases immediately. The pendulum angle control count oscillates to maintain the pendulum angle at zero (upright position). After a few oscillations, cart speed reaches its reference speed of $0.1\ (m/s)$. After the 32nd second, with reference cart speed equal to zero, both cart speed and pendulum position remains close to zero after some correctional actions. The associated motor controls for both left and right motor control counts are close to zero after around 33.5th second; see the middle and bottom plots on the right of Figure 8.43.

As a summary, the tuned gains for simulation and experimental studies could be quite different due to system nonlinearity, sensor noise, modeling error, and PWM approximation at a low duty-cycle level. The interactions between stabilization and cart motion control for experiments is heavier than that for the simulation study. This could be due to the unmodeled nonlinear friction in the linear simulation study.

Section Summary

This section validates the dual-PID mini-Segway control strategy through experiments. First, it discusses the mini-Segway system hardware architecture, along with its sensors and actuators used to generate feedback signals and drive the mini-Segway motors, respectively. Next, the section introduces the dual-PID real-time control software in MATLAB/Simulink, along with the wheel speed signal processing and motor driving software. Reminder that the pendulum angle is calculated by utilizing both accelerometer and rate-gyro signals from MPU 6050. The wheel speed is calculated based upon the hall-effect sensors mounted on the wheel axes. The section then introduces the overall software architecture in Simulink. The Simulink model of the dual-PID control strategy runs within the host computer. The Arduino MKR 1000 Wi-Fi microcontroller collects the sensor signals and executes the motor drive commands on the mini-Segway, where communication between the host computer and the Wi-Fi microcontroller is through the Wi-Fi at a frequency of 100 Hz. Last, the section conducts an experimental study of the mini-Segway to validate the mini-Segway dual-PID control strategy.

Section Keywords

Practice Problems

1. Select all correct answers below to complete the sentence: The enabled PID controller (see Figure 8.36) used for pendulum stabilization is due to the following reasons.
   a. Enable PID control as needed.
   b. Protect the mini-Segway so that the PID controller is activated only when the pendulum angle is less than 45°.
   c. Prevent integration windup.
2. Complete the sentence by selecting correct answer below. The MPU 6050 sensor signals (three-axis acceleration and rotational speed signals) are communicated with Arduino MKR 1000 Wi-Fi through:
   a. Wi-Fi communication.
   b. SPI link.
   c. I2C bus link.
3. The large difference between simulation and experimental students of the tuned PID gains for pendulum position and cart driving control is due to the following reasons. Select all associated reasons from the list below.
   a. The controller output unit difference (armature voltage vs. PWM duty-cycle count).
   b. Mini-Segway nonlinearity such as Coulomb friction.
   c. Modeling errors of DC-motor and pendulum dynamics (assumption of a uniform bar).
4. Consider the difference in the control input units (armature voltage for simulation model and duty-cycle count for experimental study). When implement the designed model-based controller, is it required to have a correction factor for the model-based controller? If so, what is the factor?

This page titled 8.7: Experimental Study of the Mini-Segway Model-Free Control is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8.8: Model-Based Control Experiments

Learning Objectives

After completing this section, you will be able to:

1. Tune the model-based control gain experimentally.
2. Explain how the square-pulse sample and hold input (SPSHI) can improve mini-Segway stabilization performance.

In order to study the feasibility of using a low-cost microcontroller (Arduino MKR 1000 Wi-Fi board), we use a mini-Segway (MS) system platform to implement a regular and special sample and hold input (SHI). In this study, the inner-loop controller is a continuous-time linear quadratic regulator (CLQR). It is implemented with a 100 Hz sample rate with zero-order sample and hold input. The outer-loop is a discrete-time linear quadratic regulator (DLQR) that uses the square pulse sample and hold input (SPSHI) with a 5 Hz sample rate. It converts a non-minimum phase (NMP) continuous-time system (mini-Segway with inner-loop CLQR controller) into a discrete-time minimum-phase (MP) system. In turn, it is possible to utilize the high-gain outer-loop DLQR to further improve stabilization performance.

After we develop and implement the MS system platform (see Section 8.7), but before we implement the outer-loop DLQR controller with SPSHI, we implement and optimize the inner-loop CLQR with ZOSHI to stabilize the MS system. We established a baseline system performance based on CLQR weight matrices $Q_{pci} = diag[100,\ 0.1,\ 300,\ 0.1,\ 10,\ 0.01]$ and $R_{pci} = diag[1,\ 1]$ called the initial CLQR weighting matrices. Starting from this baseline controller with weighting matrix Q_{pci} fixed, we tuned weighting matrix R_{pc} by introducing a gain coefficient γ_c such that the tuned control weighting is as below.

$$R_{pc}(k) = \gamma_c(k)\ R_{pci} \tag{8.8.1}$$

where $\gamma_c\,(k) = 0.6,\ 0.7,\ \ldots,\ 1.2$ for $k = 1,\ 2,\ \ldots, 7$ with an increment of 0.1. With seven continuous-time state feedback controllers designed, we implemented these controllers as a Simulink control diagram for model-based control (see problem P8.29). We conducted experiments for all seven controllers designed with responses recorded for the first six controllers with $k = 1,\ 2,\ \ldots,\ 6$; the seventh controller is not stabilizing. The mini-Segway control inputs are DC-motor armature voltages (same for both motors). Control responses are pendulum displacement θ_p and cart displacement x_{cart}. These are plotted in Figure 8.44 with control input u_{aL} on the bottom plot for the left motor, pendulum displacement θ_p on the middle, and cart displacement x_{cart} on the top. In this study, both armature voltages for the left and right motors are the same. Only the left one is plotted in Figure 8.44. By carefully study these three plots, we can see that minimal pendulum position oscillation magnitude is achieved with $\gamma_c = 0.8$ and minimal cart displacement oscillation magnitude is achieved with $\gamma_c = 0.9$.

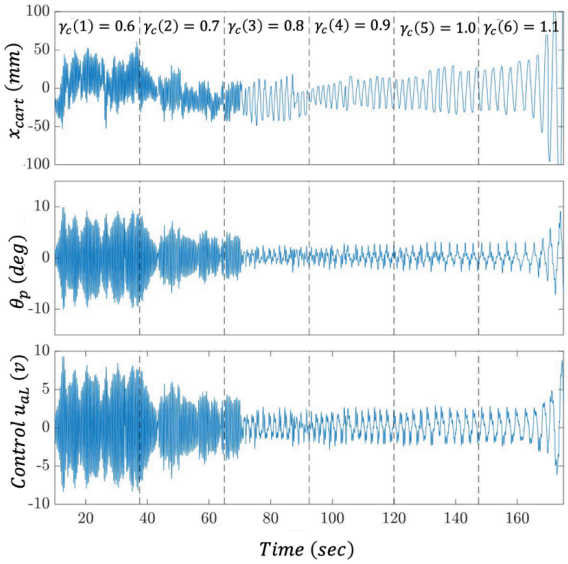


Figure 8.44: Mini-Segway stabilization with a sequence of CLQR controllers.

To help find out the optimal γ_c, Table 8.5 shows the mean absolute values of the cart oscillation magnitude $avg(|x_{cart}(k)|)$, pitch angle $avg(|\theta_p(k)|)$, and the corresponding control voltage $avg(|u_{eL}(k)|)$ for $\gamma_c(k)$ with $k = 1,\ 2,\ \ldots,\ 6$. The table confirms that the best performance among the six inner-loop CLQR controllers occurred for $k = 4$ (or $\gamma_c = 0.9$). The corresponding optimal control gain is shown below.

$$K_p = \begin{bmatrix} -7.45 & -9.91 & -15.35 & -0.95 & 2.36 & .088 \\ -7.45 & -9.91 & -15.35 & -0.95 & -2.36 & -.088 \end{bmatrix} \tag{8.8.2}$$

Table 8.5: Average magnitudes of responses and control signals.

$\gamma_c(k)$	0.6	0.7	0.8	0.9	1.0	1.1
$avg(\|x_{cart}(k)\|)\ \ (mm)$	19.35	15.32	18.63	12.17	16.58	unbounded
$avg(\|\theta_p(k)\|)\ \ (deg)$	4.31	2.56	1.08	0.85	0.89	unbounded
$avg(\|u_{eL}(k)\|)\ \ (v)$	3.96	2.44	1.16	1.00	1.10	saturated

Figure 8.44 shows that when $\gamma_c(k)$ is 0.6 (or k= 1), the resulting CLQR control gain is the highest. The mini-Segway system outputs (displacement and pendulum position) oscillate at very high frequencies. When $\gamma_c(k)$ is 0.7 (or k= 2), the control gain is slightly reduced. As a result, the oscillation magnitudes of both pendulum angle and car position are reduced, but still unstable. As a result, these two γ_c values are not feasible; they do not result in stabilization response. Next, for $\gamma_c(k)$ equal to 0.8, the closed-loop system takes about 5 seconds to stabilize. Thus, the closed-loop mini-Segway system stabilizes while it is unstable with $\gamma_c(k)$ equal to 0.7 (see Figure 8.44). The closed-loop system performance is further improved with γ_c equal to 0.9. The cart is also stabilized with an increased oscillation magnitude when γ_c is equal to 1.0. Further increasing γ_c value to 1.1 results in unbounded oscillation magnitude for cart displacement, indicating an unstable closed-loop system. Thus, the feasible (stabilizing) of γ_c is between 0.8 and 1.0. Comparing the CLQR control gain K_p in equation 8.8.2 designed with $\gamma_c = 0.9$ with the CLQR control gain K_p in equation ??? designed with $\gamma_c = 1.0$, we can see that control gain increased by about 5.4%.

In general, as the control weighting R_{cp} reduces, with the state weighting fixed, the CLQR control gain increases. If the open-loop system is stable, the CLQR optimal controller is able to stabilize the closed-loop system with very large control weighting. As the control weighting reduces, the control gain increases and closed-loop performance is improved for the design model. However, due to modeling error, when the control gain is increased to a certain point, the experimental results will be unstable. As a result, there exists an optimal control gain achieving the best experimental performance [9].

For the open-loop unstable systems like the mini-Segway, a certain level of control gain is required to stabilize the open-loop unstable system with R_{cp} bounded from above. When the gain is increased to a certain level (or R_{cp} is reduced to certain level), the closed-loop system will become unstable again due to modeling error. This is what happened for the CLQR controllers with different γ_c levels, where large and small R_{cp} levels (low and high control gains) both lead to unstable responses.

Now, for each stabilizing inner-loop CLQR controller with $\gamma_c = 0.8,\ 0.9$, and 1.0, the outer-loop DLQR controller was tuned accordingly to achieved optimal experimental performance. A baseline DLQR was designed first using weighting matrices as below.

$$Q_{db} = diag\,[100,\ 0.1,\ 80,\ 0.1,\ 10,\ 0.01]\,,\quad R_{dbi} = diag[1,\ 1] \tag{8.8.3}$$

With these feasible CLQR controllers ($\gamma_c = 0.8,\ 0.9$, and 1.0), the dual-loop feasible regions, which converts inner-loop continuous-time NMP (non-minimal phase) system into discrete-time MP (minimal phase) system, are very close to that shown on Figure 8.16. The smallest feasible sampling periods are 0.1136, 0.1162, and 0.1184 second, respectively, for $\gamma_c = 0.8,\ 0.9$, and 1.0. Because the feasible regions of the closed-loop systems with CLQR controllers ($\gamma_c = 0.8,\ 0.9$, and 1.0) using SPSHI (square pulse sample and hold input) are barely changed, the SPSHI parameter selection remains unchanged with $p = 0.2$ (duty-cycle) and $T_s = 0.2$ second (sample period).

Table 8.6: Average absolute magnitudes of cart displacement as functions of γ_c and γ_d.

$\gamma_d(k)$	∞	0.3	0.5	0.7	0.9	1.1
$avg(\|x_{cart}(k)\|)\ \ (mm)\,,\ \gamma_c = 0.8$	18.83	13.50	15.46	16.64	16.89	15.51
$avg(\|x_{cart}(k)\|)\ \ (mm)\,,\ \gamma_c = 0.9$	12.17	3.85	5.39	6.29	6.84	7.36
$avg(\|x_{cart}(k)\|)\ \ (mm)\,,\ \gamma_c = 1.0$	16.58	6.63	8.00	10.55	12.07	13.29

An experimental search of the optimal CLQR and DLQR controller combination is conducted. The three CLQR controllers are with $\gamma_c = 0.8,\ 0.9$, and 1.0. Let $\gamma_d\,(k) = 0.3,\ 0.5,\ 0.7,\ 0.9$, and 1.1 ($k = 1,\ 2,\ \ldots,\ 5$). For each given γ_c ($0.8,\ 0.9$, and 1.0), five DLQR controller with SPSHI are designed with state weighting matrix Q_{db} defined in equation 8.8.3 and the control weighting $R_{db}\,(k) = \gamma_d(k)R_{dbi}$ for $k = 1,\ 2,\ \ldots,\ 5$. This results in a total of fifteen DLQR controllers. Note that smaller γ_d values (such as 0.1 and 0.2) were tested. However, the corresponding DLQR gains were too large to be implemented due to control signal saturation with application of SPSHI for the mini-Segway system.

From the system bandwidth points of view, the cart displacement state has a natural frequency of 1.3 Hz. The pendulum position has a natural frequency of 33.9 Hz. The outer-loop discrete-time DLQR controller sampled at 5 Hz can only improve the cart displacement performance. Therefore, control gain tuning is focused to minimize cart oscillation magnitude during pendulum stabilization. Table 8.6 lists these results for each given γ_c and γ_d. The table indicates that the best performance among the three stable inner-loop CLQR and five associated outer-loop DLQR controllers is the one with $\gamma_c = 0.9$ and $\gamma_d = 0.3$. The corresponding optimal DLQR control gain is shown below, which is quite different from the design in Section 8.4; see equation ???.

$$K_d = \begin{bmatrix} -7.69 & -5.34 & -1.04 & -0.246 & 1.299 & .0489 \\ -7.69 & -5.34 & -1.04 & -0.246 & -1.299 & -.0489 \end{bmatrix} \tag{8.8.4}$$

Note that the controller gains associated with the pendulum motion (columns 3 and 4) are quite small, compared with the continuous-time CLQR controller. This indicates that the DLQR does not try to improve pendulum motion performance due to its low sample rate (5 Hz).

Table 8.6 indicates that $\gamma_c = 0.9$ and $\gamma_d = 0.3$ yield the best performance improvement. Here, the oscillation magnitude with inner-loop CLQR controller ($\gamma_c = 0.9$) only is 12.17 (mm), and dual-loop oscillation magnitude with inner-loop CLQR controller ($\gamma_c = 0.9$) and outer-loop DLQR controller ($\gamma_d = 0.3$) is 3.83 (mm). As a result, a 68.4% improvement is achieved. On the other hand, the improvement is 28.3% and 60.0% for using the inner-loop CLQR controller with $\gamma_c = 0.8$ and $\gamma_c = 1.0$, respectively.

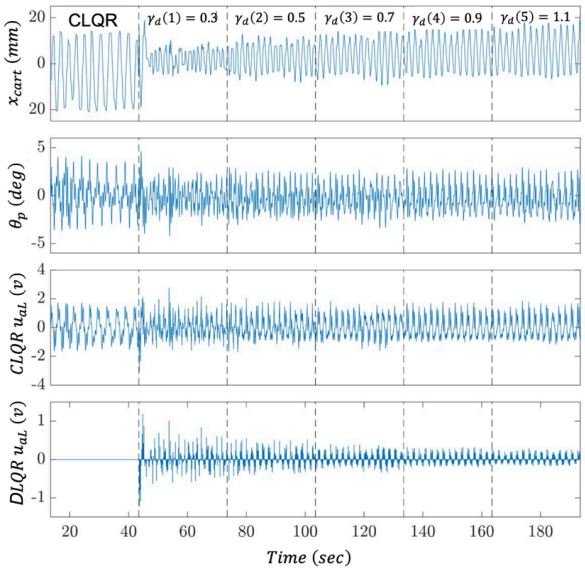


Figure 8.45: Mini-Segway stabilization with both CLQR and DLQR-SPSHI controllers.

Figure 8.45 shows the experimental results of the tuned outer-loop DLQR with $\gamma_c = 0.9$ and $\gamma_d = 0.3$. The experimental results in Figure 8.45 are divided into six segments. Segment I with $\gamma_d = \infty$ corresponds to the responses with inner-loop CLQR controller only; it used as the baseline controller. Ideally, with $\gamma_d = \infty$, the designed discrete-time DLQR controller gain is zero, leading to only the inner-loop CLQR controller active. Between Segments II and VI, γ_d increases from 0.3 to 1.1 with a 0.2 increment. The dual-loop experimental results with inner-loop CLQR and outer-loop DLQR-SPSHI show that the outer-loop DLQR controller with $\gamma_d = 0.3$ reduces the cart displacement magnitude significantly, while keeping the pendulum displacement performance almost unchanged due to the aforementioned inadequate sample frequency for pendulum motion dynamics. From Segments II to VI, with increased γ_d, the cart absolute averaged displacement magnitude increases as the DLQR-SPSHI control effort decreases.

The bottom two plots of Figure 8.45 show the overall control signals, including the inner-loop CLQR u_{aL} and outer-loop DLQR u_{aL} with SPSHI, respectively. The DLQR control signal with SPSHI is zero in Segment I because only the inner-loop CLQR controller exists. The outer-loop DLQR controller with SPSHI is activated, starting from Segment II. As γ_d increases, the outer-loop DLQR

control gain with SPSHI decreases and the associated mean absolute cart displacement magnitude increases. The mean absolute values of the cart displacement, inner-loop CLQR control input voltage u_{aL}, and outer-loop DLQR control voltage u_{aL} with SPSHI are summarized in Table 8.7, utilizing the CLQR controller designed with $\gamma_c = 0.9$.

The best (smallest) mean-absolute cart displacement is highlighted in bold in Table 8.7. From the trend shown in the top plot of Figure 8.45, when $\gamma_d = 0.3$, the cart oscillation magnitude reaches the local minimum. Meanwhile, the cart pitch angle is slightly larger than the single loop case with the inner-loop controller only. The numerical results show that the dual-loop controller with $\gamma_c = 0.9$ and $\gamma_d = 0.3$ reduces the mean absolute cart displacement amplitude from 12.17 (mm) to 3.85 (mm), a 68.3% reduction. The outer-loop DLQR-SPSHI control signal in the bottom plot of Figure 8.45 ranges from -1.0 to 1.0 volt, which does not affect the inner-loop control voltage in the third plot of Figure 8.45. From the bottom plot of Figure 8.45, we can observe that dual-loop control does not increase the overall control magnitude.

Table 8.7: Average absolute magnitudes of cart displacement as functions of γ_c and γ_d.

$\gamma_d(k)$	∞	0.3	0.5	0.7	0.9	1.1
$avg\left(\lvert x_{cart}\left(k\right)\rvert\right)\ \ (mm)$	12.17	**3.85**	5.39	6.29	6.84	7.36
$avg\left(\lvert \theta_p\left(k\right)\rvert\right)\ \ (deg)$	0.95	0.98	1.00	0.99	1.08	1.11
$avg\left(\lvert CLQR\ u_{eL}\left(k\right)\rvert\right)\ \ (v)$	0.69	0.47	0.52	0.55	0.56	0.57
$avg\left(\lvert DLQR\ u_{eL}\left(k\right)\rvert\right)\ \ (v)$	0	0.047	0.040	0.030	0.027	0.023

Section Summary

This section addresses the experimental studies of a model-based dual-loop control strategy. The inner-loop uses the standard zero-order sample and hold input (ZOSHI), and the outer-loop utilizes square-pulse sample and hold input (SPSHI) [13], [31]. Both control loops employ state-feedback controllers for regulating the mini-Segway pendulum position only. First, we use the inner-loop model-based continuous-time LQR (linear quadratic regulation) control to stabilize the pendulum; we optimize LQR input weight R_c (with fixed state weighting Q_c) to achieve the best performance experimentally. Next, we design the outer-loop discrete-time state-feedback LQR controller for the mini-Segway system. The continuous-time LQR controller is based upon the discrete-time model. It is obtained by discretizing the mini-Segway model with continuous-time LQR controller utilizing the SPSHI. See Figure 8.15 in Section 8.4 for the dual-loop LQR control system architecture using SPSHI. With the dual-loop control scheme shown in Figure 8.15, the average cart displacement magnitude is reduced by more than 68%. It uses the SPSHI for the outer-loop controller over the inner-loop only system while keeping the pendulum position performance unchanged. Even though the inner-loop CLQR is optimized for the best performance prior to implementing the outer-loop DLQR controller, a significant performance improvement can still be achieved by using the outer-loop control with SPSHI.

Section Keywords

Practice Problems

1. Modify the dual-PID Simulink control strategy shown in Figure 8.37 by replacing the dual-PID controller with the model-based one shown in equation ???. Assume that the cart turning speed and displacement are zero.
2. Select the correct answer below to complete the sentence. For the CLQR control design with difference γ_c values, the experimental results in Figure 8.44 indicate that the closed-loop system will be stable if:
   a. γ_c is greater than 0.7.
   b. γ_c is smaller than 1.1.
   c. γ_c is between 0.8 and 1.0.

This page titled 8.8: Model-Based Control Experiments is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

8.9: Chapter References

References

1. R. E. King, “The inverted pendulum,” *Amer. J. of Phys.*, vol. 33, no. 10, pp. 855–856, Oct. 1965, doi: 10.1119/1.1971009.
2. K. H. Lundberg and T. W. Barton, “History of inverted-pendulum systems,” IFAC Proc. Vol., vol. 42, no. 24, pp. 131–133, Oct. 2010, doi: 10.3182/20091021-3-JP-2009.00025.
3. R. C Dorf and R. H. Bishop, *Modern Control Systems*, 12th ed., Englewood Cliffs, NJ, USA: Prentice Hall, 2011, ISBN-13: 978-0-1360-2458-0.
4. G. Zhu, M. W. Pyclik, E. K. Bradley, and L. J. Brackney, “Closed-loop actuator control system having bumpless gain & anti-windup logic,” U.S. Patent US6424906, Jul. 2002.
5. J. G. Ziegler and N. B. Nichols, “Optimum settings for automatic controllers,” *Trans. of ASME*, vol. 64, pp. 759–768, Nov. 1942.
6. Desimone Subbaram Naidu, *Optimal Control Systems,* 1st ed., Boca Raton, FL, USA: CRC Press, 2003, ISBN-13: 978-1-3152-1442-9.
7. M. Athens and P. L. Flab, *Optimal Control: An Introduction to the Theory and Its Applications,* Garden City, NY, USA: Dover Publications, 2007, ISBN-10: 0-48-645328-6.
8. G. Zhu, M. A. Rote, and R. E. Skelton, “A convergent algorithm for the output covariance constraint control problem,” *SIAM J. on Control and Opt.,* vol. 35, no. 1, pp. 341–361, Jan. 1997.
9. G. Zhu, K. M. Grigoriadis, and R. E. Skelton, “Covariance control design for the Hubble Space Telescope,” *AIAA J. of Guid., Control, and Dyn.,* vol. 18, no. 2, pp. 230–236, May 1995.
10. C. Gamache, M. Van Nieuwstadt, J. Martz, and G. Zhu, “LQTI boost pressure and EGR rate control of a diesel air charge system with eBoost assistance,” *Int. J. of Engine Res.,* vol. 25, no. 9, pp. 1659–1670, Sep. 2024, doi: 10.1177/14680874241241364.
11. D. Hu, G. Li, G. Zhu, Z. Liu, and M. Tu, “LQT control of a commercial vehicle air brake system,” *IEEE Access* [Online], vol. 8, pp. 149741–149750, Aug. 2020, doi: 10.1109/ACCESS.2020.3016353.
12. Y. Wang, R. Jafari, G. Zhu, and R. Mukherjee “Sample and hold inputs for minimum-phase behavior of non-minimum-phase systems,” *IEEE Trans. on Control Syst. Tech.,* vol. 24, no. 6, pp. 2103–2111, Nov. 2016.
13. Y. Wang and G. Zhu, “Performance Improvement of an NMP Mini-Segway Using Sample and Hold,” *MPDI J. of Appl. Sci.* (*Special Issu. Recent Adv. on Mech. Vib. Control*), vol. 13, no. 2, p. 1070, Feb. 2023, doi: 10.3390/app13021070.
14. J. K. Astrom and L. Rundqwist, “Integrator windup and how to avoid it,” in *Proc. 1989 Amer. Control Conf.*, Pittsburgh, PA, USA, Jun. 21–23, 1989, pp. 1693–1698.
15. J. Zheng, Y. Guo, M. Fu, Y. Wang, and L, Xie, “Improved reset control design for a PZT position stage,” in *Proc. 2007 IEEE Int. Conf. on Control Appl.,* Singapore, Oct. 1–3, 2007, pp. 1272–1277.
16. J. G. Ziegler and N. B. Nichols, “Optimum settings for automatic controllers,” *Trans. of ASME,* vol. 64, no. 8, pp. 759–768, Nov. 1942.
17. S. Qu, T. He, and G. Zhu, “Gain-scheduled PID control optimized by iterative genetic algorithm of NSGA-II with application to a nonlinear valve system,” *MPDI J. of Appl. Sci. (Special Issue on Res. Appl. and Intelligent Control Alg.),* vol. 13, no. 5, p. 6444, May 2023, doi: 10.3390/app13116444.
18. L. Hua and G. Zhu, “Optimal transient longitudinal trajectory tracking for electrical connected vehicles,” *MPDI J. of Machines,* vol. 13, no. 2 [GrindEQ__163_], p. 13020263, Feb. 2025, doi: 10.3390/machines13020163.
19. M. Tomizuka, “Zero phase error tracking algorithm for digital control,” *ASME J. of Dyn. Syst., Meas., and Control*, vol. 109, no. 1, pp.65–68, Mar. 1987.
20. E. Gross, M. Tomizuka, and W. Messner, “Cancellation of discrete time unstable zeros by feedforward control,” *ASME J. of Dyn. Syst., Meas., and Control,* vol. 116, pp. 33–38, Mar. 1994.
21. B. Haack and M. Tomozuka, “The effect of adding zeros to feedforward controller,” *ASME J. of Dyn. Syst., Meas., and Control,* vol. 113, no. 1, pp. 6–10, Mar. 1991.
22. D. Torfs, J. De Schutter, and J. Swevers, “Extended bandwidth zero phase error tracking control of nonminimum phase systems,” *ASME J. of Dyn. Syst., Meas., and Control,* vol. 114, no. 3, pp. 347–351, Sep. 1992.
23. R. Jafari and R. Mukherjee, “Intermittent output tracking for linear single-input single-output non-minimum-phase systems,” in *Proc. Amer. Control Conf.*, Montréal, Canada, June 2012, pp. 5942–5947.
24. K. J. Aström, P. Hagander and J. Sternby, “Zeros of sampled systems,” *Autom. Control*, vol. 20, no. 1, pp. 31–38, Jan. 1984, doi: 10.1016/0005-1098[GrindEQ__84_]90062-1.
25. C. H. Wang, W.-Y. Wang, and C.-C. Hsu, “Minimum-phase criteria for sampled systems via symbolic approach,” *Int. J. of Syst. Sci.*, vol. 28, no. 8, pp. 767–775, Jul. 1997, doi: 10.1080/00207729708929437.
26. C. Hsu, T. Lu, “Minimum-phase criterion on sampling time for sampled–data interval systems using genetic algorithms,” *Appl.*

Soft Comput., vol. 8, no. 4, pp. 1670–1679, Sep. 2008, doi: 10.1016/j.asoc.2008.01.009.
27. Y. Fu and G. A. Dumont, "Choice of sampling to ensure minimum-phase behavior," *IEEE Trans. on Automat. Contr.*, vol. 34, no. 5, pp. 560–563, May 1989.
28. M. Ishitobi, "Criteria for stability of zeros of sampled systems," in *IEE Proc. of Control Theory and Appl.*, vol. 141, no. 6, pp.396–402, Nov. 1994.
29. T. Hagiwara, "Analytical study on the intrinsic zeros of sampled-data systems," *IEEE Trans. on Automat. Contr.*, vol. 41, no. 2, pp. 261–263, Feb. 1996.
30. C. E. Shannon, "Communication in the presence of noise," in *Proc. Inst. of Radio Eng.*, vol. 37, no. 1, pp. 10–21, Jan. 1949.
31. Y. Wang, G. G. Zhu, R. Mukherjee, "Experimental study of NMP sample and hold input using an inverted pendulum," in *Proc. ASME 2018 Dyn. Syst. Control Conf.*, Atlanta, GA, p. V001T01A004, Sep. 30–Oct. 3, 2018.
32. J. E. Bernard, "The simulation of Coulomb friction in mechanical systems," *Sage J. of Simulation,* vol. 34, no. 1, pp. 11–16, Jan. 1980, doi: 10.1177/003754978003400104.

Solutions to Practice Problems

1. The number of degrees of freedom is b) 3.
2. The answer is b) False. From the middle equation of ???, we can see that when the mini-Segway pendulum angle θ_p is not equal to zero, there is a term $\dot{\phi}^2$ in the equation, affecting the linear motion in x direction.
3. The answer is a) True. From the bottom equation of ???, we can see that when the mini-Segway pendulum control inputs $u_{aL} = u_{aR}$, the forcing term of the differential equation for ϕ is zero. As a result, $\dot{\phi}$ will be damped out or goes to zero.
4. The answer is b) False. From the top and middle equations of ???, we can see that mini-Segway linear motion in x direction and angular pendulum motion θ_p are highly coupled. This is the main reason we can use the cart linear motion to stabilize the mini-Segway pendulum.
5. The answer is discrete-time PID is preferred for implementation to a microcontroller. The main reason is that implementing a discrete-time PID controller into a microcontroller will significantly reduce the computational load.
6. The answer is c) 6f_h. Note that if a) or b) were selected, certain physical system dynamics could be filtered out, leading to slow derivative action for the PID controller. The rule is to filter out as much as noise but keep the system dynamics in the error signal $e(t)$.

1. The answer is a), b) and d). Note that for case c), if the integral is enabled, the unsaturated PID control output will continuously increase, causing integration windup.
2. The answer is a).
3. The answer is b) 5.
4. The statement is true. However, for the actual physical system, these two dynamics (cart turning $\dot{\phi}$ and cart-speed/pendulum-position $\dot{x}_{cart}/\theta_p$) are coupled with a quadratic term. During the process of linearization to obtain the state-space model, the coupling relation is eliminated due to the quadratic coupling relation.
5. The created mode is not controllable. This is mainly due to the fact that any controller designed cannot change/move the integration pole location.
6. The control gain related to the integration error state x_{ce} is not affected by the ε value. For example, when the ε value is adjusted from 0.0001 to 0.001, the control gain associated with the integration error state x_{ce} changed from 0.4880 to 0.4877 with a slight reduction of 0.0003 (about 0.0615%). This is very small compared with a ten-time increment of the ε value. As a result, ε value does not need to be fine-tuned.

1.

$$A_p^{-1}\left(e^{A_pT_s}-I\right)=T_s+\frac{A_pT_s^2}{2!}+\frac{A_p^2T_s^3}{3!}+\frac{A_p^3T_s^4}{4!}+\cdots \tag{8.9.1}$$

Therefore, even though A_p is not invertible, term $A_p^{-1}\left(e^{A_pT_s}-I\right)$ can be calculated (it exists). But the inverse of A_p does not exist because it is singular.

1. The answer is b) because there is at least one pole located at the origin of the complex plan, which makes the system matrix not invertible.
2. The answer is a).
3. The answer is a); see Figure 8.19.
4. The answer is c); see Figure 8.20.
5. The answer is b); see the Simulink nonlinear model shown in Figure 8.8.
6. True. See the discussion before Figure 8.19.

1.

The simulation results are plotted below. Due to increased gains on both cart speed and pendulum position, we can see that the initial oscillation magnitude is reduced, whereas the cart speed tracking performance remains almost unchanged.

1. The answers are b) and c).
2. The answer is c), the I2C bus link.
3. The answer is that all the factors listed from a) to c) could result in the gain difference.
4. The answer is: Yes, a correction factor is needed. The factor should be used to multiply to the designed model-based controller gain, and the factor is $k_c = 22.17$.
5. This modification can be achieved by replacing the dual-PID control logics with the state feedback one similar to the LQTI controller in Figure 8.26. (Delete the integration block and set both the cart-turning displacement and speed to zero).
6. The answer is c); γ_c is between 0.8 and 1.0.

This page titled 8.9: Chapter References is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

9: THREE-WHEEL CART - MECHATRONIC SYSTEM EXAMPLE 2

Learning Objectives

1. Describe the mechatronic system modeling approach through the three-wheel cart system modeling and tire lateral-slip dynamics.
2. Distinguish the dual- and triple-PID control architectures of forward/backward and turning motions for the three-wheel cart and compensation of unwanted lateral motion.
3. Explain the development process of system state-space model based on equations of motion developed in Section 9.1 for model-based tracking control design.
4. Demonstrate the simulation validation process for dual-PID, triple-PID, and LQTI three-wheel cart control strategies, along with construction of Simulink simulation validation models.
5. Demonstrate the experimental validation process for dual-PID, triple-PID, and LQTI three-wheel cart control strategies and construction of real-time Simulink control models.

This chapter introduces three-wheel cart system modeling, its cart speed and turning regulation using model-free and model-based control strategies, where the model-based turning speed control compensates tire lateral-slip by minimizing a quadratic cost function (LQTI).

9.0: Prelude

9.1: Three-Wheel Cart System Modeling

9.2: Dual- and Triple-PID Control of a Three-Wheel Cart

9.3: Model-Based Control of a Three-Wheel Cart

9.4: Simulation Study of PID and Model-Based Control Strategies

9.5: Experimental Investigation of PID Control Strategies

9.6: Experimental Investigation of Model-Based Control Strategy

9.7: Chapter References

This page titled 9: Three-Wheel Cart - Mechatronic System Example 2 is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

9.0: Prelude

Chapter 8 described the mechatronic system of a mini-Segway in detail from modeling and control system architecture to simulations and experimental validations. Chapter 9 presents a second mechatronic system example: a three-wheel cart. Here, the lateral wheel-slip will be considered in modeling, control, and experiment studies. Figure 9.1 illustrates the chapter's six sections. Section 9.1 presents a three-wheel cart mathematical model. Sections 9.2 and 9.3 present model-free and model-based control methodologies. Section 9.4 then validates these two control system methods in software-in-the-loop (SiL) simulation studies. Finally, Sections 9.5 and 9.6 validate experimentally these model-free and model-based control methodologies, respectively.

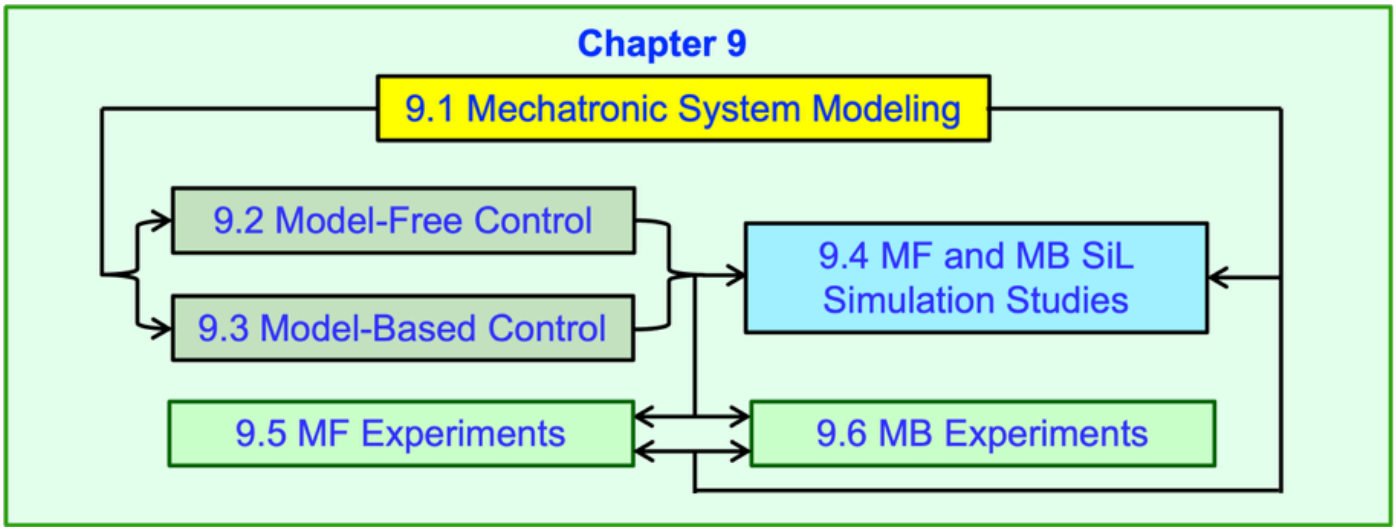


Figure 9.1: Organization of Chapter 9 sections.

Section 9.1 models our second example of a mechatronic system: a three-wheel cart with two driving wheels in the front and one swivel caster wheel in the back. The example uses the modeling techniques from Chapter 2. In Chapter 7, we modeled a three-wheel cart as a lumped mass. In this section, we model in detail a two-dimensional rigid body of the three-wheel cart with its linear (straight) motion, turning motion, and lateral-sliding motion. Due to the relatively hard material used for wheel tires, if the ground is not leveled, the three-wheel cart moves sideways, even when the two driving wheels have the same speed. Section 9.1 works this phenomenon into the model. The section also models the cart's electric and sensor characteristics; the mechanical dynamics are based on Newton's law. The model is then implemented in MATLAB and Simulink.

We've now developed the three-wheel cart model, factoring in the tire lateral-slip due to lateral force [1]. Section 9.2 proposes two control architectures using PID (proportional-integral-derivative) controllers, introduced in Section 3.4 [2], [3], [4]. The dual-PID architecture uses two PID controllers to regulate the cart's two driven wheel speeds. Both driven wheels track their respective reference speeds. The cart forward driving and turning can be controlled by adjusting these reference speeds. Because of the potential tire lateral-slip and unwanted lateral motion (turning), the second architecture (triple-PID) adds another PID controller. This helps to regulate the cart turning speed to the desired turning rate based upon the rate gyro feedback. With the help of integral gain, the unwanted turning motion can be compensated.

Section 9.3 addresses the model-based design process (see Section 3.5) for the three-wheel cart modeled in Section 9.1. It utilizes design methods for both LQR control (linear quadratic regulation) [5], [6], [7], [8] and LQTI control (linear quadratic tracking with integral) [9], [10], [11]. In order to apply these methods, the three-wheel cart model is converted into its state-space model form and then populated with the parameter values in Table 9.1. Next, an LQR regulator is designed for stabilizing the three-wheel cart. Two error integral states of the three-wheel cart forward/backward and turning speeds are then added to the state-space model [9], [11], with an LQTI controller designed to track them. This LQTI control strategy is a model-based one whereas the triple-PID control strategy discussed in Section 9.2 is a model-free strategy.

Section 9.4 conducts simulation validations of the designed controllers, including dual-PID control, triple-PID control, and LQTI control strategies. The dual-PID control strategy is validated with leveled and unleveled ($\theta_z \neq 0$ and $\theta_y \neq 90^\circ$) ground. The triple-PID control strategy is validated with zero turning reference and show that it is able to compensate for lateral-slip. The LQTI control strategy is validated with improved performance over the triple-PID strategy. Both triple-PID and LQTI strategies are able

to compensate the lateral-slip introduced by unleveled ground surface. Without integration compensation, the cart will move to a side even though the two wheels have the same linear speed.

Section 9.5 develops the real-time Simulink software for dual- and triple-PID control. Once software is implemented in the laptop (host computer), we debug and discuss the developed software. Then, we tune both dual- and triple-PID control gains to achieve the best performance. We test their associated control strategies in straight driving and turning on leveled ground surface, with test data collected in the Simulink environment. We also discuss test data sets from both dual- and trip-PID control strategies, especially for maintaining the desired turning rate.

Section 9.6 develops real-time Simulink software for model-based control. Once software is implemented into the laptop, we debug and discuss the developed control software. Then, we conduct the experimental study utilizing model-based LQTI control of the three-wheel cart, along with its control gain tuning through modifying the design weighting matrices. We collect the test data in the Simulink environment through the display scope, then compare the model-based test performance with the triple-PID one.

Orienting Questions

1. What is the method used to model the three-wheel cart with tire lateral-slip dynamics?
2. Why shall we compensate for the unwanted lateral motion of a three-wheel cart?
3. How can optimal model-based regulation and tracking controllers for a three-wheel cart system be designed?
4. In what ways can we validate PID and model-based control strategies in the SiL environment?
5. How do we validate model-free (PID) and model-based control strategies experimentally?

This page titled 9.0: Prelude is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

9.1: Three-Wheel Cart System Modeling

Learning Objectives

After completing this section, you will be able to:

1. Model the three-wheel cart system based upon its rigid body dynamics.
2. Describe the modeling process of the wheel/tire lateral-slip dynamics.

This section introduces the modeling process of the target three-wheel cart shown in Figure 9.2. A high-quality rechargeable sealed lead-acid battery (12V 12Ah) is on the back of middle deck. The mechatronic system electronics (Arduino MKR 1000 Wi-Fi board, MKR motor carrier board, MPU 6050 three-axis acceleration and rate gyro sensor board, and the MSU interface board) are in the front of the middle deck. Two DC-motor wheel assemblies with associated hall-effect speed sensors are mounted on the bottom-front. Similar to the mini-Segway system, the mini-USB connector on the Wi-Fi board is extended to the top of the three-wheel cart for easy accessing and connection reliability. This connector can be easily ripped off the board. With the full-size USB connector on the top of the cart, the USB connector is secured, even with large pulling force applied to it. In addition, a power switch and a power supply fuse are added to the top-back side of the cart. On the back-bottom, a swivel caster wheel is installed with its direction to be automatically aligned with the cart's driving direction. The swivel caster is passive without any actuator powering it.

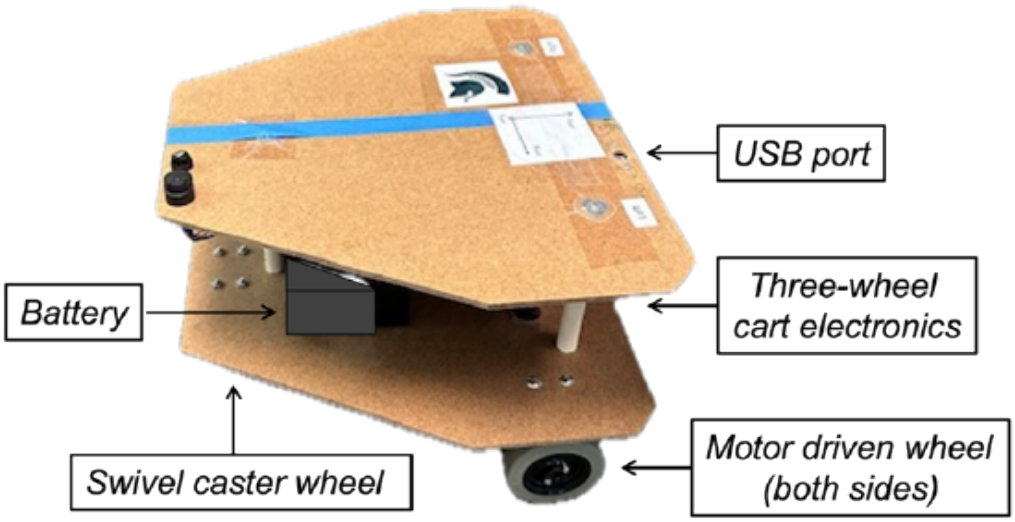


Figure 9.2: The three-wheel cart system.

The three-wheel cart in Figure 9.2 is a rigid-body two-actuator robot. It utilizes a low-cost microcontroller (Arduino MKR 1000 Wi-Fi), two motor-wheel sets, and a cart body. Figure 9.3 shows a free body diagram of the cart's system, where the left and right motors provide the driving torques, τ_L and τ_R, respectively. In addition, x_{cart}, y_{cart}, and z_{cart} are the direction of the cart's displacement and x_{abs}, y_{abs}, and y_{abs} are the direction of the cart's absolute displacements. Also, θ_L (shown on Figure 9.3) and θ_R (not shown) are the left and right wheel rotational angles, with direction defined in Figure 9.3 for the left wheel and the right wheel in opposite direction, respectively; ϕ is the cart body yaw angle, respectively. Finally, l, l_{cfw}, d_{cmg}, and h are the longitudinal distance between the front and back wheel axes, the longitudinal distance between the front wheel axis and center of mass; the vertical distance from the center of mass to ground, and the yaw distance between the two front wheels, respectively.

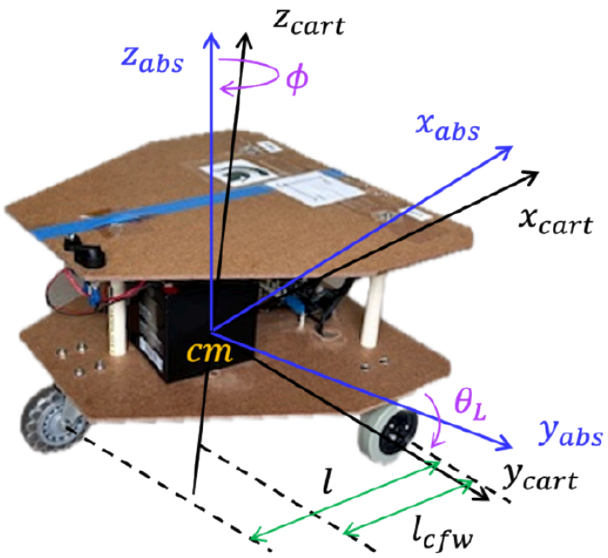


Figure 9.3: The diagram and axis definitions for the three-wheel cart.

With all parameters and variables defined in Figure 9.3, similar to the mini-Segway system in Chapter 8, we can conduct the following step-by-step modeling process.

We'll start with the DC motor and wheel dynamics. The motor and wheel assembly is the same as the one shown in Figure 8.4 with a different wheel mass m_w, axis moment of inertia I_w, and radius r_w. (See Table 9.1 for these numerical values.) Using the same approach as that in Section 8.1, we have the following, assuming that the DC-motor armature winding inductance $L_a \approx 0$.

$$I_w \frac{d\omega_L}{dt} = I_w \frac{d\dot{\theta}_L}{dt} = \tau_{mL} - b\omega_L - \tau_L \tag{9.1.1}$$

$$\tau_{mL} = \frac{-K_\tau K_m}{R_a}\omega_L + \frac{K_\tau}{R_a}u_{aL} = -\frac{K_\tau K_m}{R_a}\dot{\theta}_L + \frac{K_\tau}{R_a}u_{aL} \tag{9.1.2}$$

where u_{aL} is the left DC-motor armature voltage; R_a is the DC-motor armature winding resistance; τ_{mL}, τ_L, and $\tau_{fL} = b\omega_L$ are the left DC-motor output torque, load torque applied to the left wheel due to vehicle motion, and the left motor axis and bearing friction torque; respectively; I_w is the total left combined moment of inertia of left wheel, wheel axis, and left motor rotor; and θ_L and $\omega_L = \dot{\theta}_L$ are the rotational angular displacement and speed of the left wheel, respectively. Finally, K_τ and K_m are the DC-motor torque and electromotive force (EMF) coefficients, respectively, where motor output torque $\tau_{mL} = K_\tau i_{aL}$ and electromagnetic field voltage $e_{mL} = K_m\omega_L$. Table 9.1 lists numerical values of these parameters.

Consider the right wheel spinning in the opposite direction of the left one. Assuming that the right motor and wheel have the same parameters as the left one, we have the following equations for the right motor and wheel assembly.

$$I_w \frac{d\omega_R}{dt} = I_w \frac{d\dot{\theta}_R}{dt} = \tau_{mR} - b\omega_R - \tau_R \tag{9.1.3}$$

$$\tau_{mR} = \frac{-K_\tau K_m}{R_a}\omega_R + \frac{K_\tau}{R_a}u_{aR} \quad = -\frac{K_\tau K_m}{R_a}\dot{\theta}_R + \frac{K_\tau}{R_a}u_{aR} \tag{9.1.4}$$

where the subscript R is used for the right motor and wheel model above. Now, consider the motion of wheel moving forward. In this case, motor torque τ_{mR} (τ_{mL}) works against motor friction torque $\tau_{fR} = b\dot{\theta}_R = b\omega_R$ (or $\tau_{fL} = b\dot{\theta}_L = b\omega_L$) and drives wheel and three-wheel cart assembly by τ_R (τ_L). The friction torque is due to both bearing and gear friction. As a result, the following equation for the right wheel can be obtained.

$$\tau_R = \tau_{mR} - \tau_{fR} = \tau_{mR} - b\dot{\theta}_R = \frac{-K_\tau K_m}{R_a}\dot{\theta}_R + \frac{K_\tau}{R_a}u_{aR} - b\dot{\theta}_R \tag{9.1.5}$$

For the left motor wheel assembly, we have the following. Assume motor and wheel parameters are the same.

$$\tau_L = \tau_{mL} - \tau_{fL} = \tau_{mL} - b\dot{\theta}_L = \frac{-K_\tau K_m}{R_a}\dot{\theta}_L + \frac{K_\tau}{R_a}u_{aL} - b\dot{\theta}_L \tag{9.1.6}$$

During the preliminary driving tests of the three-wheel cart, we found that with synchronized left- and right-front wheel speeds, the three-wheel cart slides toward the low ground side when the driving surface is not leveled. This challenge is the main motivation for modeling the lateral motion of the three-wheel cart—due to the lateral gravity force applied to cartwheels when the ground is not leveled. Before diving into the model, however, let's review some of the literature.

Vehicle dynamics and system modeling has been developed for decades, with applications to the intelligent transportation system (ITS) aspects [12], autonomous driving [13], and connected vehicles [14]. Tire dynamics, a core vehicle characteristic, has been extensively studied. Studies include the Fiala model in the 1950s [15], Brush model [16], Dugoff in the 1970s [17], Magic Formula [18], UniTire [19], LuGre [20], and FTire model in the 1990s [21]. In general, tire models can be divided into three groups: theoretical, semi-empirical, or pure empirical. A set of equations and lookup tables are used to map the tire operating parameters, e.g., slip-rate to these forces applied to the tire.

Almost all vehicle-controlled motions are through its tire forces, which should be accurately calculated to achieve the desired control performance [22], [23], [24]. Due to the high cost and challenge of direct measurement, tire lateral forces are normally estimated [25] using model-based or empirical approach. For example, we can estimate tire forces, vehicle lateral-slip angle, and wheel cornering stiffness by measurements such as yaw rate, longitudinal/lateral accelerations, steering angle, and angular wheel velocities. These are available from sensors installed on modern automotives [26]. They use estimation methods such as sliding mode observer, Kalman filter [27], extended Kalman filter [28], unscented Kalman filter [29], and empirical approach [30].

To model the tire lateral-slip of a three-wheel cart, we will use an empirical approach due to the lateral force caused by the lateral gravity component on unleveled ground. The lateral force is perpendicular to the direction the wheel is pointing (rotating) and happens in the center of the tire contact patch. See Figure 9.4, where τ_L is the left motor output torque; f_{axL} and f_{azL} are forces applied to the left wheel axes in x_{cart} and z_{cart} directions, respectively; and f_{xL} and f_{yL} are forces applied to the left wheel ground-contact point in x_{cart} and y_{cart} directions, respectively.

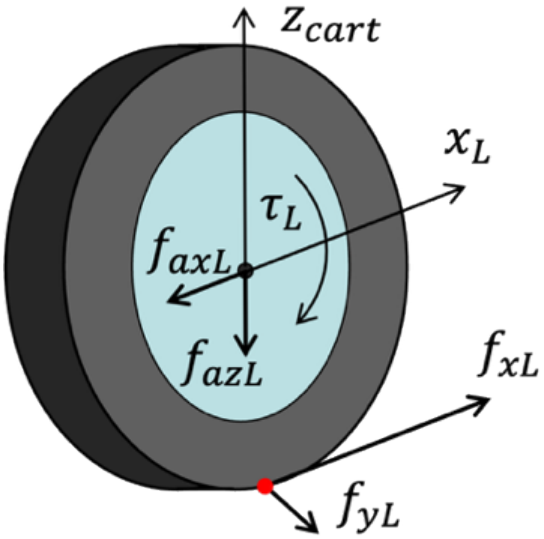


Figure 9.4: Forces applied to the left DC motor and wheel assembly on the road

The left wheel torque balance around ground contact point (red dot in Figure 9.4) yields the following by applying Newton's law.

$$\tau_L - f_{axL}r_w = \tau_{mL} - b\dot{\theta}_L - f_{axL}r_w = (I_w + m_w r_w^2)\ddot{\theta}_L \tag{9.1.7}$$

where r_w is the effective wheel-tire radius; and I_w and m_w are the wheel assembly (including gearbox and DC-motor rotor) moment of inertia around its axis and wheel assembly mass, respectively. Combining equations 9.1.6 and 9.1.7 leads to the following equation.

$$\left(I_\omega + m_\omega r_w^2\right)\ddot{\theta}_L = \frac{-K_\tau K_m}{R_a}\dot{\theta}_L + \frac{K_\tau}{R_a}u_{aL} - b\dot{\theta}_L - f_{axL}r_w = \tau_{mL} - b\dot{\theta}_L - f_{axL}r_w \tag{9.1.8}$$

Similarly, the following equation can be obtained using the same process for the right wheel, assuming all parameters are the same as the left one.

$$\left(I_\omega + m_\omega r_w^2\right)\ddot{\theta}_R = \frac{-K_\tau K_m}{R_a}\dot{\theta}_R + \frac{K_\tau}{R_a}u_{aR} - b\dot{\theta}_R - f_{axR}r_w = \tau_{mR} - b\dot{\theta}_R - f_{axR}r_w \tag{9.1.9}$$

Now, consider the force in the y_{cart} direction. Assume the three-wheel cart vertical axis z_{cart} is not in line with the reference axis z_{abs}. Gravity force f_g (see the blue arrow in Figure 9.5), in the opposite direction of z_{abs}, can be factorized into components in x_{cart} , y_{cart}, and z_{cart} directions, as shown in Figure 9.5.

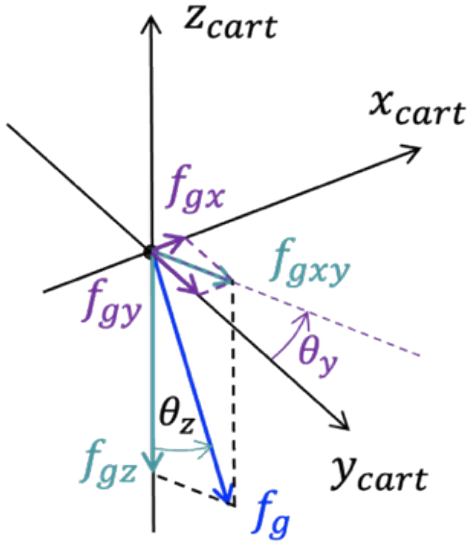


Figure 9.5: Gravity force projection to x_{cart}, y_{cart}, *and* z_{cart} *components*

The angle between z_{abs} and z_{cart} is defined as θ_z. In addition, f_{gz} and f_{gxy} are the projected gravity forces of f_g in the three-wheel cart coordinate in z_{cart} direction and the plane formed by x_{cart} and y_{cart} axes, respectively. Next, let the angle between f_{gxy} and y_{cart} be defined as θ_y. We can calculate the gravity force components f_{gx} and f_{gy} with known pair of θ_z and θ_y using the following equations and the relationships shown in Figure 9.5.

$$\begin{aligned} f_{gx} &= f_g \sin\theta_z \sin\theta_y \\ f_{gy} &= f_g \sin\theta_z \cos\theta_y \end{aligned} \tag{9.1.10}$$

The relation between lateral force and slip angle is reciprocal. The lateral force can be considered as a result of the slip angle, and the slip angle as a result of lateral force. Figure 9.6 defines the slip angle, where v_L is the left wheel speed, v_{xL} is the projection of the left wheel speed to the x_L axis; v_{yL} is the projection of the left wheel speed to the y_L axis; and the slip-angle α is defined as the angle between the x_L axis and left wheel speed vector v_L.

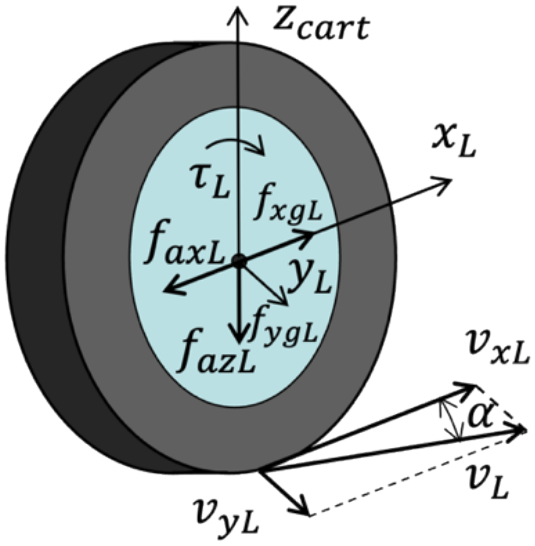


Figure 9.6: The left DC motor and wheel assembly on the road

Slip-angle is proportional to the lateral force for a given tire property. That is, the larger the lateral force, the larger the slip-angle, where the lateral force could be caused by centrifugal force generated by the vehicle turning motion or by external force. Centrifugal force [32] increases by decreasing the turn radius or increasing the vehicle's linear speed. The external force could be caused by unleveled road due to gravity or gust wind. As the slip angle α increases, some lateral sliding begins to occur in the tire tread. If the slip angle is further augmented, eventually the tire will break loose and slide over the contact surface, which could happen easily for a solid wheel. For the three-wheel cart used in this chapter, the wheel-tire surface is relatively hard without tread. As a result, the lateral-slip considered is mainly from the tire sliding over the contact surface.

In general, there are three ranges of tire behavior when slip-angle α is not zero. These are no sliding (deformation of tire), some sliding (transitional range), and total sliding (frictional range) [1]. Within the sliding ranges, the sliding between the tire and the ground is caused only by lateral force applied to the associated wheel. Figure 9.7 shows the relationship between slip-angle and lateral force for the three-wheel cart. In the transitional range, the lateral force increases as the slip angle goes up. In the frictional range, the lateral force peaks at its left boundary; instead, it goes down a bit as the slip-angle increases.

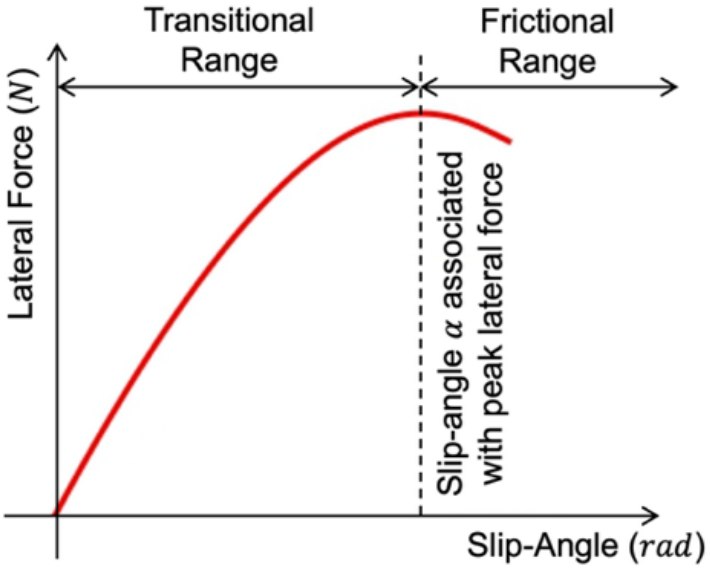


Figure 9.7: Relationship between slip-angle and lateral force

The lateral force caused by gravity due to unleveled ground is relatively small for the three-wheel cart. Thus, we assume in the modeling process that slip-angle α is a linear function of lateral force. Figure 9.7 indicates the relationship between slip-angle and lateral force is linear when the lateral force is small. Regarding slip angle α, where f_{ygL} is defined in Figure 9.6, we can assume

the following.

$$\alpha = c_{sa} f_{ygL} \tag{9.1.11}$$

In equation 9.1.11, c_{sa} is the slip-angle coefficient to be calibrated. The coefficient c_{sa} is also a function of the normal force applied to the wheel. In this study, because θ_z is small, we assume the normal force is a constant. Another advantage of modeling the lateral slip using the slip-angle is that the resulting lateral slip speed is proportional to the tire linear speed. This is natural because the higher the tire rotational speed, the lower the lateral friction coefficient. We assume that the rear wheel cannot generate any reaction force in y_{cart} direction because it is a swivel caster wheel; gravity force f_{yg} in y_{cart} direction is evenly distributed on both left and right wheels. Therefore, the lateral force f_{ygL} applied to the left wheel is $f_{ygL} = 0.5 f_{yg}$, where f_{yg} is defined in equation 9.1.10. In turn, we can calculate the slip-angle below for one wheel. Parameter c_{sa} and gravity force f_g can be combined as a single calibration if f_g is fixed.

$$\alpha = 0.5 c_{sa} f_g \sin\theta_z \cos\theta_y \tag{9.1.12}$$

Because the tire lateral slip speed is modeled using the slip-angle, we will study forward and turning motion first. Based on the obtained linear wheel speed, we can calculate the cart lateral speed based upon the slip angle α. In the following, we assume that there is no slip between wheel and ground in the tire directions x_L and x_R. We have $x_L = \theta_L r_w$ and $x_R = \theta_R r_w$ that yield the following.

$$\dot{x}_L = \dot{\theta}_L r_w, \quad \ddot{x}_L = \ddot{\theta}_L r_w, \quad \dot{x}_R = \dot{\theta}_R r_w, \quad \ddot{x}_L = \ddot{\theta}_R r_w \tag{9.1.13}$$

Equations 9.1.8 and 9.1.9 can be rewritten as functions of left and right wheel axis linear displacements, x_L and x_R, as below.

$$f_{axL} = \frac{-K_\tau K_m}{R_a r_w^2}\dot{x}_L + \frac{K_\tau}{R_a r_w} u_{aL} - \frac{b\dot{x}_L}{r_w^2} - \frac{I_\omega \ddot{x}_L}{r_w^2} - m_\omega \ddot{x}_L \tag{9.1.14}$$

$$f_{axR} = \frac{-K_\tau K_m}{R_a r_w^2}\dot{x}_R + \frac{K_\tau}{R_a r_w} u_{aR} - \frac{b\dot{x}_R}{r_w^2} - \frac{I_\omega \ddot{x}_R}{r_w^2} - m_\omega \ddot{x}_R \tag{9.1.15}$$

Axis forces, f_{axL} and f_{axR}, in equations 9.1.14 and 9.1.15, are applied to the three-wheel cart body as shown in Figure 9.8. Next, we find the equations of motion for the three-wheel cart with x_{cart} and ϕ representing the displacement and turning angle at the three-wheel cart center of mass location, respectively.

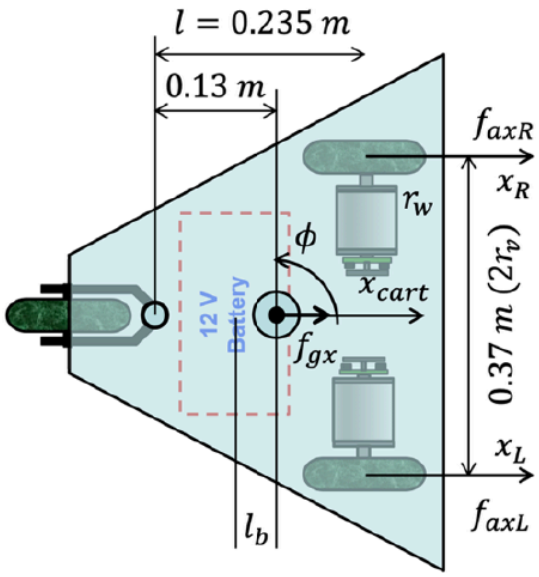


Figure 9.8: The three-wheel cart free-body diagram

Assume that the three-wheel cart mass is represented by m_{cart} (see Table 9.1). Wheel forces f_{axL} and f_{axR} in x_L and x_R directions, along with the gravity force f_{gx} in x_{cart} direction at three-wheel cart center of mass (dark-dot in Figure 9.8), can be used to develop the equation of motion in x_{cart} direction as below, utilizing Newton's law.

$$m_{cart}\ddot{x}_{cart} = f_{axL} + f_{axR} + f_{gx} \tag{9.1.16}$$

Utilizing the following relationships

$$\begin{aligned} \dot{x}_{cart} &= 0.5(\dot{x}_L + \dot{x}_R), \\ \ddot{x}_{cart} &= 0.5(\ddot{x}_L + \ddot{x}_R) \end{aligned} \tag{9.1.17}$$

and substituting equations 9.1.10, 9.1.14 and 9.1.15 into equation 9.1.16 yields the following equation.

$$m_{cart}\ddot{x}_{cart} = \frac{-2K_\tau K_m}{R_a r_w^2}\dot{x}_{cart} + \frac{K_\tau}{R_a r_w}(u_{aL} + u_{aR}) - \frac{2b\dot{x}_{cart}}{r_w^2} - \frac{2I_\omega \ddot{x}_{cart}}{r_w^2} - 2m_\omega \ddot{x}_{cart} \quad + f_{gx} \tag{9.1.18}$$

where f_{gx} is the gravity force applied to three-wheel cart body, as defined in equation 9.1.19 and also in equation 9.1.10.

$$f_{gx} = f_g \sin\theta_z \sin\theta_y \tag{9.1.19}$$

Substituting equation 9.1.19 into equation 9.1.18, reorganizing, and simplifying the resulting equation, yield the first equation of motion for the three-wheel cart as follows.

$$\left(m_{cart} + 2m_\omega + \frac{2I_\omega}{r_w^2}\right)\ddot{x}_{cart} + 2\left(\frac{K_\tau K_m}{R_a r_w^2} + \frac{b}{r_w^2}\right)\dot{x}_{cart} = \frac{K_\tau}{R_a r_w}(u_{aL} + u_{aR}) + f_g \sin\theta_z \sin\theta_y \tag{9.1.20}$$

Next, we consider the three-wheel cart turning motion. We start with turning motion ϕ_x due to left and right wheel speed difference in x_{cart} direction. Then we will consider the combined turning motion ϕ of ϕ_x and lateral-slip turning ϕ_{ts} to provide the second equation of motion. Let the combined three-wheel cart moment of inertia I_{cart} in the z_{cart} direction be

$$I_{cart} = 2\left(l_{cfw}^2 + r_v^2\right)m_w + \left(I_{zbat} + l_b^2 m_{bat}\right) + I_{cz} \tag{9.1.21}$$

The first term in equation 9.1.21 is the moment of inertia of two DC motor and wheel assemblies around the center of gravity. The second term is the moment of inertia of the battery around the center of gravity. The last term is the cart moment of inertia of the cart without the battery and two motor-wheel assemblies. Parameters m_w, l_{cfw}, and r_v are respectively the motor and wheel assembly mass, the distance from the center of the cart total mass to the front axis defined in Figure 9.3, and the half distance between the two front wheels defined in Figure 9.8. Parameters m_{bat}, l_b, and I_{zbat} are battery mass, distance between battery and cart centers of mass, and battery moment of inertia around its center of mass, respectively. Finally, parameter I_{cz} is the cart moment of inertia without battery and two motor-wheel assemblies. With the cart total moment of inertia I_{cart} defined, applying Newton's law leads to the following equation of motion.

$$I_{cart}\ddot{\phi}_x = f_{axL} r_v - f_{axR} r_v \tag{9.1.22}$$

Utilizing the following relationships

$$\dot{\phi}_x = \frac{(\dot{x}_L - \dot{x}_R)}{r_v}, \qquad \ddot{\phi}_x = \frac{(\ddot{x}_L - \ddot{x}_R)}{r_v} \tag{9.1.23}$$

and substituting equations 9.1.14 and 9.1.15 into equation 9.1.22 yields the following equation.

$$I_{cart}\ddot{\phi}_x = -\frac{r_v^2}{r_w^2}\left(b + \frac{K_\tau K_m}{R_a}\right)\dot{\phi}_x - \frac{r_v^2}{r_w^2}\left(I_\omega + m_\omega r_w^2\right)\ddot{\phi}_x + \frac{r_v K_\tau}{r_w R_a}(u_{aL} - u_{aR}) \tag{9.1.24}$$

The above equation is part of the second equation of motion for the three-wheel cart and can be simplified as the following.

$$\left(I_{cart} + \frac{r_v^2}{r_w^2}I_\omega + m_\omega r_v^2\right)\ddot{\phi}_x + \frac{r_v^2}{r_w^2}\left(b + \frac{K_\tau K_m}{R_a}\right)\dot{\phi}_x = \frac{r_v K_\tau}{r_w R_a}(u_{aL} - u_{aR}) \tag{9.1.25}$$

Next, we will investigate the cart rotation factor $\dot{\phi}_g$ due to tire lateral-slip based on slip-angle α, and use it to find the lateral slip-speed. The three-wheel cart has two fixed wheels; considering individual wheel tire-slip is not feasible. Instead, we assume a virtual wheel in the middle of the left and right wheels, with the resulting wheel longitudinal speed of $\dot{x}_{cart}$. The rear swivel caster will not slide because its direction will be changed to follow the cart motion automatically. The following formula calculates the lateral gravity force applied to the virtual wheel, assuming that no lateral gravity force is applied to the rear caster wheel.

$$f_{gy} = f_g \sin\theta_z \cos\theta_y \tag{9.1.26}$$

The resulting slip-angle can then be calculated as below.

$$\alpha = c_{sa} f_{gy} = c_{sa} f_g \sin\theta_z \cos\theta_y \tag{9.1.27}$$

The resulting lateral speed and associated angular speed can be calculated below based upon Figure 9.7.

$$\dot{y}_g = \dot{x}_{cart} \tan\alpha = \dot{x}_{cart} \tan\left(c_{sa} f_g \sin\theta_z \cos\theta_y\right) \ , \ \dot{\phi}_g = -\frac{\dot{y}_{ts}}{l_{cfw}} = -\beta_3 \dot{x}_{cart} \tag{9.1.28}$$

Let total angular speed $\dot{\phi}$ be the summation of $\dot{\phi}_x$ and $\dot{\phi}_g$. That is, $\dot{\phi} = \dot{\phi}_x + \dot{\phi}_g$. Note that $\ddot{\phi} = \ddot{\phi}_x + \ddot{\phi}_g$. Assuming slow change of θ_z and θ_y ($\dot{\theta}_z = \ddot{\theta}_z = 0$ and $\dot{\theta}_y = \ddot{\theta}_y = 0$) yields the following.

$$\ddot{\phi}_g = -\frac{\ddot{y}_g}{l_{cfw}} = -\frac{\tan\left(c_{sa} f_g \sin\theta_z \cos\theta_y\right)}{l_{cfw}} \ddot{x}_{cart} = -\beta_3 \ddot{x}_{cart} \tag{9.1.29}$$

Utilizing relations of $\dot{\phi} = \dot{\phi}_x + \dot{\phi}_g$ and $\ddot{\phi} = \ddot{\phi}_x + \ddot{\phi}_g$, equations 9.1.27 and 9.1.29 lead to the following equation (the second equation of motion).

$$\beta_1 \ddot{\phi} + \beta_2 \dot{\phi} + \beta_1 \beta_3 \ddot{x}_{cart} + \beta_2 \beta_3 \dot{x}_{cart} = \frac{r_v K_\tau}{r_w R_a} \left(u_{aL} - u_{aR}\right) \tag{9.1.30}$$

where

\[β_1=I_{cart}+\dfrac{r^2_v}{r^2_w}I_ω+m_ωr^2_v, \[4pt] β_2=\dfrac{r^2_v}{r^2_w}\left(b+\dfrac{K_τK_m}{R_a}\right), \ β_3=\dfrac

ParseError: invalid DekiScript (click for details)

```
Callstack:
    at (Bookshelves/Introductory_Engineering/Mechatronics:_Fundamentals_Design_Integration_and
```

{l_{cfw}} \label{9.31}\]

Simplifying both equations of motion in 9.1.30 and 9.1.20 results in the following two equations of motion for the three-wheel cart considering unleveled road surface.

\begin{aligned} ended with \end{array}

where α_1, α_2, and α_3 are defined below.

$$\begin{aligned} \alpha_1 &= m_{cart} + 2m_\omega + \frac{2I_\omega}{r_w^2}, \\ \alpha_2 &= \frac{K_\tau K_m}{R_a r_w^2} + \frac{b}{r_w^2}, \\ \alpha_3 &= f_g \sin\theta_z \sin\theta_y \end{aligned} \tag{9.1.31}$$

From equations ???, ???, and 9.1.31, we find that when $\theta_y = \pi/2$, then $\beta_3 = 0$. That is, the lateral force is zero, leading to zero slip-angle. In this case, the turning motion equation in ??? is shown in equation 9.1.32, indicating that the turning motion is caused only by the difference between the DC-motor armature voltages. When the difference is zero, the turning will be gradually stopped with nonzero initial turning speed or no turning motion with zero initial speed. That is, if $\theta_y = \pi/2$, the gravity force will not generate lateral motion due to zero tire slip in y_{cart} direction.

$$\ddot{\phi} = -\frac{\beta_2}{\beta_1} \dot{\phi} + \frac{r_v K_\tau}{\beta_1 r_w R_a} \left(u_{aL} - u_{aR}\right) \tag{9.1.32}$$

When $\theta_y = \pi/2$, then $\theta_z \neq 0$, and $\alpha_3 \neq 0$, resulting in additional force applied to the three-wheel cart to accelerate or decelerate the cart. If $\theta_y = 0$, then $\alpha_3 = 0$. Thus, the cart speed is affected only by motor driving torques. As a result, if $\theta_y = 0$, gravity force will not affect the movement in the x_{cart} direction.

Table 9.1 lists all model parameter definitions and associated values for the target three-wheel cart mechatronic system. It also lists key model parameters, including α_1, α_2, α_3, β_1, β_2, and β_3.

Table 9.1: Three-wheel cart parameter nomenclatures (names, values, and associated units).

Symbol	Description	Value (Unit)
τ_{mL} and τ_{mR}	Left and right DC-motor torques generated.	(Nm)
i_{aL} and i_{aR}	Left and right DC-motor armature currents.	(A)
e_{mL} and e_{mR}	Left and right DC-motor EMF voltages.	(V)
K_m	DC-motor EMF constant (at wheel axis)	$0.027\ (V \cdot s/rad)$
K_τ	DC-motor torque constant (at wheel axis)	$0.23\ (Nm/A)$
u_{aL} and u_{aR}	Left and right DC-motor armature voltages.	(V)
R_a	DC-motor armature resistance	$4.8\ (\Omega)$
L_a	DC-motor armature inductance (assumed to be 0)	$\sim 0\ (H)$
b	DC-motor axis and gear-box friction coefficient	$\sim 0\ (Nm \cdot s/rad)$
τ_L and τ_R	Left and right applied wheel torques	(Nm)
$\dot{\theta}_L$ and $\dot{\theta}_R$	Left and right wheel speeds	(rad/s)
θ_L and θ_R	Left and right wheel angular displacements	(rad)
I_w	Wheel axis moment of inertia (including rotor)	$6.09 \times 10^{-5} (Kgm^2)$
I_{zbat}	Battery moment of inertia $m_b\left(a^2+b^2\right)/12$	$8.57 \times 10^{-3} (Kgm^2)$
I_{cz}	Cart moment of inertia w/o wheel and battery	$0.0161\ (Kgm^2)$
I_{cart}	Three-wheel cart moment of inertia (z-direction)	$0.0534\ (Kgm^2)$
z_{cg}	Distance from center of gravity to ground	$0.138(m)$
l_{cfw}	Distance from center of gravity to front wheels	$0.105(m)$
l_b	Distance between cart and batter center of gravity	$0.035(m)$
l	Distance between front and rear wheel	$0.282(m)$
m_w	Wheel mass (including rotor/gear mass)	$0.265\ (Kg)$
m_{bat}	Battery mass	$3.293\ (Kg)$
m_{cart}	Three-wheel cart total mass	$4.488\ (Kg)$
f_L and f_R	Left and right wheel traction (driving) forces	(N)
f_{axL} and f_{axR}	Left and right horizonal hinge forces	(N)
r_w	Wheel effective radius (all)	$0.038\ (m)$
r_v	Half front-wheel width	$0.1885\ (m)$
$x_{cart}\ (x)$	Three-wheel cart center of mass displacement	(m)
ϕ	Three-wheel cart turning angle	(rad)
α_1	EOM cart acceleration ($\ddot{x}_{cart}$) coefficient	$5.1023\ (s^2/m)$
α_2	EOM cart speed ($\dot{x}_{cart}$) coefficient	$1.5885\ (s/m)$
$\alpha_3\left(\theta_z, \theta_y\right)$	Gravity component in x_{cart} direction	(N)
β_1	EOM cart turning acceleration ($\ddot{\phi}$) coefficient	$24.6696\ (s^2/rad)$
β_2	EOM cart turning speed ($\dot{\phi}$) coefficient	$0.0564\ (s/rad)$
$\beta_3\left(\theta_z, \theta_y\right)$	Gravity component coefficient in x_{cart} direction	$(rad/m-s)$
c_{sa}	Wheel slip coefficient	0.1

The model in equation ??? is linear if $\theta_z = 0$ or both θ_z and θ_y are constant. That is, the three-wheel cart system is linear if both θ_z and θ_y are constant, under the assumption there is no longitudinal tire slip for both driving wheels. When neither θ_z nor θ_y is constant, the three-wheel cart system is a linear-parameter-varying (LPV) system [33], [34]. LPV control techniques can be used for model-based control design. However, when the rates of variation for both θ_z and θ_y are slow compared with the system states, x_{cart}, $\dot{x}_{cart}$, ϕ, and $\dot{\phi}$, multiple model-based controllers can be designed for an LTI (linear time-invariant) system model with fixed θ_z and θ_y, and control gain can be interpreted for a gain-scheduling control implementation [35]. For this application, LPV control is recommended; see Section 9.3.

Figure 9.9 shows a Simulink implementation of the mathematical three-wheel cart model described by equation ???. Two second-order differential equations are implemented with four integrators. Parameters α_1, α_2, α_3, β_1, β_2, and β_3 are calculated based upon equations ??? and 9.1.31. The Simulink model in Figure 9.9 has four inputs. They are left and right motor armature voltage inputs (u_{aL} and u_{aR}) and the unleveled ground angles (θ_z and θ_y) with respect to the cart vertical z-axis and lateral y-axis (defined in Figure 9.7).

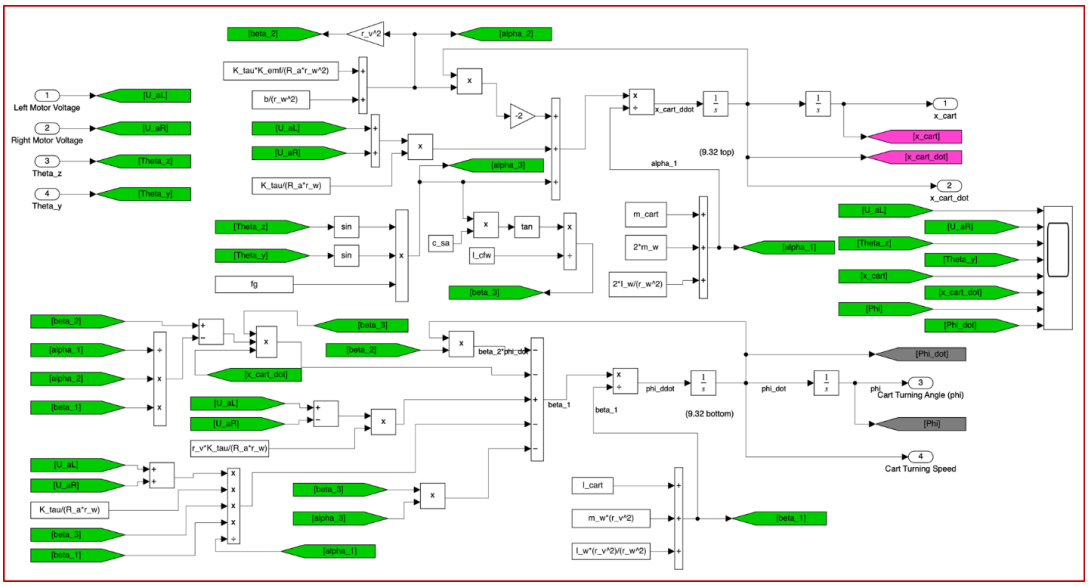


Figure 9.9: Simulink realization of developed three-wheel cart model

The Coulomb friction in the motor gear boxes is not modeled in the three-wheel cart model because there is no high-frequency forward and backward motions for the three-wheel cart. The Coulomb friction model in the mini-Segway model is used mainly to handle the forward and backward model for stabilizing the mini-Segway pendulum motion at high frequency. The simulation study of the Simulink model in Figure 9.9 will be conducted in Section 9.4 of this Chapter.

Section Summary

This section models the second mechatronic system example: a three-wheel cart with two driving wheels in the front and one swivel caster wheel in the back. The section uses the modeling techniques from Chapter 2. Chapter 7 modeled a three-wheel cart as a point mass. This section developed in detail a two-dimensional rigid body model of the three-wheel cart with its linear (straight) motion, turning motion, and lateral-slip motion. Due to relatively hard tire material, if the ground is not leveled, the three-wheel cart slides aside even when the two driving wheels have the same speed. This section models this phenomenon to reflect this. The three-wheel cart motor electric characteristics are modeled in detail, and the mechanical dynamics is modeled based upon Newton's law. The model is then implemented in MATLAB and Simulink.

Section Keywords

Practice Problems

1. What is the number of degrees of freedom (DOFs) for the three-wheel cart? Select the correct answer.
   a. 2

   b. 3
   c. 4
2. From the three-wheel cart equations of motion shown in equation ???, we can conclude if the following statement is true or not. The three-wheel cart turning motion ϕ will not be affected by its linear motion ($\dot{x}_{cart}$) in x_{cart} direction.
   a. True
   b. False
3. Select True or False. If both DC-motor armature voltages u_{aL} and u_{aR} in equation ??? are the same, the three-wheel cart turning motion ϕ will be damped out (going to zero).
   a. True
   b. False
4. Select True or False. The three-wheel linear motion $\dot{x}_{cart}$ in x_{cart} direction and its turning motion $\dot{\phi}$ are not coupled.
   a. True
   b. False

This page titled 9.1: Three-Wheel Cart System Modeling is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

9.2: Dual- and Triple-PID Control of a Three-Wheel Cart

Learning Objectives

After completing this section, you will be able to:

1. Describe the dual-PID control architecture for the three-wheel cart.
2. Provide the architecture of triple-PID control of a three-wheel cart for compensating unwanted lateral motion.

This section uses the PID (proportional-integral-derivative) control technique to control the three-wheel cart for forward/backward and turning motions. One possible control strategy is to adopt the PID speed control strategy similar to that used for controlling mini-Segway (see Figure 8.12). Here, the vehicle forward and backward motion is controlled by a single PID controller using the error between the reference (desired) and actual speed. The turning motion is controlled in open-loop by adding and subtracting a bias armature voltage to left and right motors, respectively. The main disadvantage of this control strategy is its inability to compensate for the difference between left and right motors and associated drive electronics. That is, even though the same armature reference signals are applied for both motors, the three-wheel cart may not be driven in a straight line. Therefore, this section will not use this method. Instead, two new control methods utilizing PID controllers are proposed and studied as below.

- **Method 1:** Independent closed-loop PID speed control of both left and right driving wheels. Two independent PID controllers regulate both left and right wheel speeds, respectively. Forward and backward cart straight motions are realized by providing the same reference speed to both PID controllers. Turning motion is enabled by using the different reference speed signals for left and right wheels.
- **Method 2:** Triple PID controllers are used for left and right wheel speed control with additional turning rate compensation. The wheel speed control with two PID controllers is the same as that in Method 1. The third PID controller is added for closed-loop control of the turning motion to compensate for the lateral motion (turning) induced by gravity on unleveled ground.

Let's first review the detailed PID control technique along with its control parameter tuning methods, addressed in both Section 3.4 and reference [2]. The transfer function of a continuous-time PID controller can be expressed in the following form. Let $e(t)$ be a continuous-time error signal (reference minus measured control target). Then, in the frequency domain, a continuous-time PID control law can be described below in the form of a Laplace transform with $U_{PID}(s)$ as control output.

$$U_{PID}(s) = \left(K_p + \frac{K_i}{s} + K_d s\right) E(s) = C_{PID}(s) E(s) \tag{9.2.1}$$

where K_p, K_i, and K_d are proportional, integral, and derivative gains, respectively; $E(s)$ and $U_{PID}(s)$ are the Laplace transforms of the PID error input $e(t)$ and control output $u_{PID}(t)$ in time-domain, respectively; and $C_{PID}(s)$ is the continuous-time transfer function of the PID controller. As discussed in Section 8.2, a low-pass filter is open placed before the derivative operation to reduce the amplification of noise due to derivative operation. When saturation operation is used after the PID control output, the integration windup phenomenon could occur [3], leading to slow PID control response. The windup occurs when the PID controller output is saturated and the integrator continuously integrates the error, making the saturation worse. When the error signal sign is reversed, it takes a long time to get the integration output to reflect it, causing slow PID control response. To avoid integration windup of PID controllers, an anti-windup logic was developed by modifying the PID controller strategy [3] and [36]. The discrete-time PID control law can be described by the following z-transform.

$$U_{PID}(z) = \left(K_p + \frac{K_i T_s}{1 - z^{-1}} + \frac{K_d}{T_s}(1 - z^{-1})\right) E(z) = C_{PID}(z) E(z) \tag{9.2.2}$$

where K_p, K_i, and K_d are discrete-time proportional, integral, and derivative gains, respectively; T_s is the sample period of the discrete-time PID controller; and $E(z)$ and $U_{PID}(z)$ are z-transforms of discrete-time error signal $e(k)$ and PID control output $u_{PID}(k)$, respectively. Similarly, a low-pass filter before derivative operation and anti-windup logic can be combined with the discrete-time PID controller; see Section 8.2.

Now, we can discuss control Method 1, utilizing dual-PID control architecture for the three-wheel cart system, where two PID controllers are used to control both left and right wheel speeds.

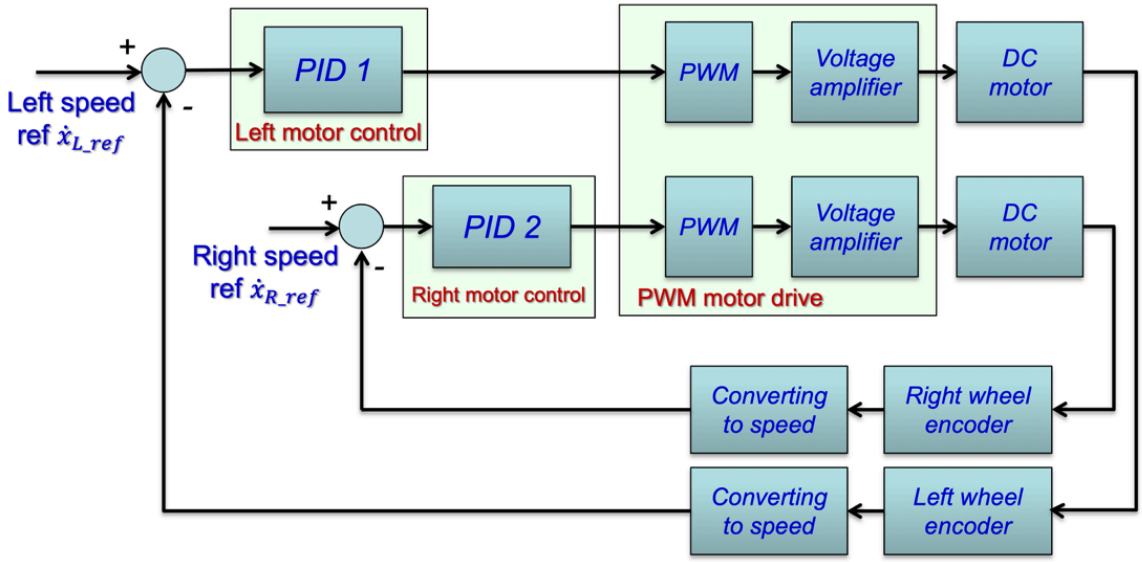


Figure 9.10: Dual-PID control architecture for a three-wheel cart.

Figure 9.10 shows the entire architecture of the dual-PID control system. It includes all associated hardware: two DC motors and two-wheel speed encoders; the MPU 6050 three-axis accelerometers and rate-gyros are not used in this case (but will be in Method 2). We can see that both left and right wheel speed signals are measured by two hall-effect sensors, respectively. In order to achieve the desired three-wheel cart speed, we convert the desired cart linear speed signal r_{cart} to designed left and right wheel speed signals by dividing the signal by the wheel radius r_w, assuming that each wheel does not slip on the surface.

$$\dot{\theta}_{L\_cart} = \dot{\theta}_{R\_cart} = \frac{r_{cart}}{r_w} \tag{9.2.3}$$

This control method relies heavily on the assumption that both wheel speed measurements are accurate without lateral-slip. Also, the left and right wheel radiuses are the same, which are generally true. The cart turning motion is controlled in closed loop by converting the turning rate to speed increments for the left and right wheels ($\Delta\dot{\theta}_L$ and $\Delta\dot{\theta}_R$). In this case, we can calculate the cart turning speed as a function of left and right wheel speeds.

$$r_\phi = \left(\Delta\dot{\theta}_L - \Delta\dot{\theta}_R\right)\frac{r_w}{r_v} \quad \rightarrow \quad \Delta\dot{\theta}_L = -\Delta\dot{\theta}_R = r_\phi\frac{r_w}{2r_v} \tag{9.2.4}$$

With $\dot{\theta}_{L\_cart}$, $\dot{\theta}_{R\_cart}$, $\Delta\dot{\theta}_L$, and $\Delta\dot{\theta}_R$ calculated based upon the equations 9.2.3 and 9.2.4, we can calculate the reference wheel speed signals ($\dot{\theta}_{L\_ref}$ and $\dot{\theta}_{R\_ref}$) to achieve desired linear speed $\dot{x}_{cart\_ref}$ and turning speed $\dot{\phi}_{Ref}$.

$$\begin{aligned} \dot{\theta}_{L\_ref} &= \dot{\theta}_{L\_cart} + \Delta\dot{\theta}_L, \\ \dot{\theta}_{R\_ref} &= \dot{\theta}_{R\_cart} + \Delta\dot{\theta}_R \end{aligned} \tag{9.2.5}$$

The control gains for the two PID controllers can be set to the same due to similar dynamic characteristics. The individual PID gain tuning can be based upon the direct-tuning method introduced in Section 3.4, Ziegler-Nichols step response, and ultimately sensitivity methods. Ziegler-Nichols methods can also be found in references [2] and [37]. PID gain tuning based upon the machine learning method [38] can also be used for gain tuning. Because two independent PID controllers were used to control the left and right motors (wheels), their PID control gains need to be tuned together using the same control gains for both controllers.

One disadvantage of the dual-PID control strategy is that the cart turning motion is controlled in open-loop by modifying the reference wheel speed signals. This lead to the inability to compensate the turning due to lateral-slip. Figure 9.11 shows the addition of another PID control loop. Here, the third PID controller is used to control the turning speed $\dot{\phi}$ in a closed-loop by creating a compensation reference signal to two-wheel speed PID controllers so that the desired turn speed can be achieved. As discussed in Section 9.1, unleveled ground could contribute to the lateral-slip occurring due to gravity force projected to the lateral direction, which causes unwanted turning motion. The third PID controller helps regulate the turning speed to its reference level. It also helps compensator the turning motion caused by lateral motion due to unleveled ground. The feedback signal was measured by the rate-gyro in the vertical cart direction z_{cart}. Specifically, when the reference signal for turning speed is set to zero, the turning PID controller prevents the cart turning by maintaining the turning rate at zero.

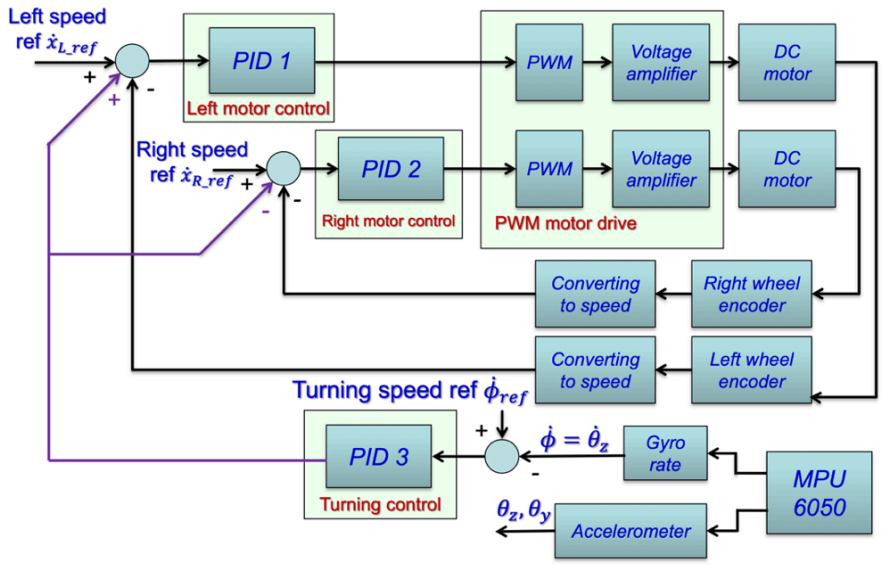


Figure 9.11: Triple-PID control architecture for a three-wheel cart.

Section 9.1 derives the lateral motion model due to unleveled ground. The gravity projection angles θ_z and θ_y help define the gravity components in the cart forward and lateral directions (x_{cart} and y_{cart}). In fact, both angles can be obtained by three-axis acceleration outputs (a_x, a_y, and $-a_z$) of MPU 6050 in x_{cart}, y_{cart} and z_{cart} directions, respectively, assuming small cart acceleration. Based upon the geometry shown in Figure 9.5, we can calculate angle θ_z using the following formula, assuming that the accelerometer in the z-direction is in the opposite direction of z_{cart}.

$$\theta_z = \tan^{-1}\left(\frac{\sqrt{a_x^2 + a_y^2}}{a_z}\right) \tag{9.2.6}$$

Similarly, the angle θ_y can be calculated below based on the geometry shown in Figure 9.5.

$$\theta_y = \tan^{-1}\left(\frac{a_x}{a_y}\right) \tag{9.2.7}$$

When $\theta_z \neq 0$ and $\theta_y \neq 90°$, these calculated angles can be used for feedforward control to compensate for the lateral motion caused by the unleveled ground due to nonzero lateral force. Because the three-axis acceleration measurement is also affected by the cart acceleration, the calculated θ_z and θ_y may not be accurate under large cart acceleration operation. If LPV (linear parameter-varying) gain scheduling control strategies [33], [34] are used, the calculated θ_z and θ_y can be used to calculate the gain scheduling parameters in real-time. For practical application of the three-wheel cart control, Method 2 is recommended. Both cart and turning speed are controlled in a closed-loop, enabling compensation for system noises such as unleveled road and hardware parameter difference (e.g., DC motors).

Section Summary

With the three-wheel cart model developed, considering the tire lateral-slip due to lateral force [1], this section proposes an architecture of using multiple PID controllers for the three-wheel cart speed and turning control [2], [3], [4]. One control architecture uses two PID controllers (dual-PID) to regulate the two driven wheel speeds so that both driven wheel speeds track their reference ones. The cart forward driving and turning can be controlled by adjusting the reference speeds of the two driven wheels independently. As discussed during the modeling process, the unleveled ground can lead to tire lateral-slip, causing unwanted lateral motion. Therefore, the second approach adds another PID controller (triple-PID) to regulate the cart turning speed to the desired turning rate. With the help of the integral gain of the turning PID controller, the unwanted turning motion can be compensated.

Section Keywords

Practice Problems

1. State the reason why the speed control strategy used for the mini-Segway (see Section 8.2) was not selected for the three-wheel cart application.
2. Select the correct answer from the list below to complete the sentence. For control Method 1, both driving wheel speed in the x_{cart} direction and the cart turning speed in the z_{cart} direction are:
   a. Controlled in open loop.
   b. Controlled in closed loop.
   c. Controlled in closed loop and open loop, respectively.
3. Select the correct answer from the list below to complete the sentence. For control Method 2, the driving wheel speed in the x_{cart} direction and the turning speed in the z_{cart} direction are:
   a. Controlled in open loop.
   b. Controlled in closed loop
   c. Controlled in closed loop and open loop, respectively.
4. Find the correct gravity projection angles θ_z and θ_y for given accelerometer outputs $a_x = 1.0$, $a_y = 1.754$, and $a_z = 9.6\ (m/s^2)$ from the list below.
   a. $\theta_z = 0.2073\ (rad)$ and $\theta_y = 0.5182\ (rad)$
   b. $\theta_z = 0.5182\ (rad)$ and $\theta_y = 0.2073\ (rad)$
   c. $\theta_z = 0.2073\ (rad)$ and $\theta_y = 0.3182\ (rad)$

This page titled 9.2: Dual- and Triple-PID Control of a Three-Wheel Cart is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

9.3: Model-Based Control of a Three-Wheel Cart

Learning Objectives

After completing this section, you will be able to:

1. Develop a state-space model with varying parameters for model-based control design based upon the developed nonlinear model in Section 9.1.
2. Design a state feedback LQR (linear quadratic regulation) controller for the three-wheel cart system.
3. Generate a state feedback LQTI (linear quadratic tracking with integral) controller utilizing double integrators.

Before starting the model-based control design, we need to develop a state-space model of the three-wheel cart based upon two differential equations shown in equation ??? with α_3 and β_3 as functions of θ_z and θ_y defined in Figure 9.5. Note that the state-space system model is in the following form.

$$\dot{x}_p = A_p x_p + B_p u + D_p v \tag{9.3.1}$$

$$y = C_p x_p + H_p u \tag{9.3.2}$$

where x, u, and y are the state vector, control input vector, and the system output vector, respectively; and A_p, B_p, C_p, and H_p are state-space model system, input, output, and direct output matrices, respectively; and $D_p v$ is disturbance input vector. For the three-wheel cart system, define the state, control input, and system output vectors as following.

$$x_p = \left[x_{cart}, \dot{x}_{cart}, \phi, \ \dot{\phi}\right]^T, u = \left[u_{aL}, \ u_{aR}\right]^T, y = \left[\dot{x}_{cart}, \dot{\phi}\ \right]^T \tag{9.3.3}$$

With the vectors defined above, the system matrices and disturbance input vector of the three-wheel cart model are provided below.

$$A_p = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\dfrac{\alpha_2}{\alpha_1} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\left(\dfrac{2\alpha_2}{\alpha_1} + \dfrac{\beta_2}{\beta_1}\right)\beta_3 & 0 & -\dfrac{\beta_2}{\beta_1} \end{bmatrix}, B_p = \begin{bmatrix} 0 & 0 \\ \dfrac{K_\tau}{\alpha_1 R_a r_w} & \dfrac{K_\tau}{\alpha_1 R_a r_w} \\ 0 & 0 \\ \dfrac{K_\tau}{R_a r_w}\left(\dfrac{r_v}{\beta_1} - \dfrac{\beta_3}{\alpha_1}\right) & \dfrac{-K_\tau}{R_a r_w}\left(\dfrac{r_v}{\beta_1} + \dfrac{\beta_3}{\alpha_1}\right) \end{bmatrix}$$

$$C_p = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, H_p = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, (D_p v)^T = \alpha_3(\theta_z, \theta_y) \begin{bmatrix} 0 & \dfrac{1}{\alpha_1} & 0 & \dfrac{\beta_3(\theta_z, \theta_y)}{-\alpha_1} \end{bmatrix} \tag{9.3.4}$$

where parameters α_1, α_2, α_3, β_1, β_2, and β_3 are defined in equations ??? and ???; and $v\,(\theta_z, \theta_y) = \alpha_3(\theta_z, \theta_y)\beta_3(\theta_z, \theta_y)$. Utilizing the parameter values provided in Table 9.1 and assuming $\theta_z = 0$ and $\theta_y = 0$ (leveled ground), the numerical system matrices can be found below for matrices $A_p = A_{p0} + A_{p\theta}\beta_3(\theta_z, \theta_y)$ and $B_p = B_{p0} + B_{p\theta}\beta_3(\theta_z, \theta_y)$, with output and direct feedforward matrices C_p and D_p given in equation 9.3.4.

$$
\begin{aligned}
A_{p0} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.3113 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -0.0023 \end{bmatrix}, \\
B_{p0} &= \begin{bmatrix} 0 & 0 \\ 0.2471 & 0.2471 \\ 0 & 0 \\ 0.0096 & -0.0096 \end{bmatrix} \\
A_{p\theta} &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -0.6249 & 0 & 0 \end{bmatrix}, \\
B_{p\theta} &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.2471 & -0.2471 \end{bmatrix}, \\
D_p &= \begin{bmatrix} 0 \\ 0.1960 \\ 0 \\ -0.1960 \end{bmatrix}
\end{aligned} \tag{9.3.5}
$$

The three-wheel cart system is open-loop unstable with double integrators for the cart linear and angular displacements. However, for controlling both linear speed $\dot{x}_{cart}$ and angular speed $\dot{\phi}$, we can reduce the control design model order to a second-order system by defining the state, control, and output vectors as below

$$
x_{cp} = \left[\dot{x}_{cart}, \dot{\phi}\right]^T, u_{cp} = \left[u_{aL},\ u_{aR}\right]^T, y = \left[\dot{x}_{cart}, \dot{\phi}\ \right]^T \tag{9.3.6}
$$

with the state space model defined below

$$
\begin{aligned}
\dot{x}_{cp} &= A_{cp}x_{cp} + B_{cp}u_{cp} + D_{cp}v \\
y &= C_{cp}x_{cp} + H_{cp}u
\end{aligned} \tag{9.3.7}
$$

where $A_{cp} = A_{cp0} + A_{cp\theta}\beta_3(\theta_z, \theta_y)$ and B_{cp}= $B_{cp0} + B_{cp\theta}\beta_3(\theta_z, \theta_y)$, and the constant matrices A_{cp0}, $A_{cp\theta}$, B_{cp0}, $B_{cp\theta}$, D_{cp}, C_{cp}, and H_{cp} are defined below.

$$
\begin{aligned}
A_{cp0} &= \begin{bmatrix} -0.3113 & 0 \\ 0 & -0.0023 \end{bmatrix}, B_{cp0} = \begin{bmatrix} 0.2471 & 0.2471 \\ 0.0096 & -0.0096 \end{bmatrix}, \ D_{cp} = \begin{bmatrix} 0.1960 \\ -0.1960 \end{bmatrix} \\
A_{cp\theta} &= \begin{bmatrix} 0 & 0 \\ -0.6249 & 0 \end{bmatrix}, B_{cp\theta} = -0.2471 \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}, C_{cp} = \boldsymbol{I}_{2\times 2}, H_{cp} = \boldsymbol{0}_{2\times 2}
\end{aligned} \tag{9.3.8}
$$

The model shown in equations 9.3.7 and 9.3.8 is an LPV (linear parameter-varying) system [33], [34]. For optimal control result, we shall design an LPV gain-scheduling controller [33], [34]. In this study, we will design the LQR stabilization and LQTI controllers. For continuous-time stabilizing regulation, the LQR controller minimizes the following cost function

$$
J_{cp} = \frac{1}{2}\int_0^\infty \left(x_{cp}^T(t)\, Q_{cp} x_{cp}(t) + u_{cp}^T(t) R_{cp} u_{cp}(t)\right) dt \tag{9.3.9}
$$

where $Q_{cp} \geq 0$ and $R_{cp} > 0$ are the weighting matrices to be selected. We want to design a state-feedback controller because both states ($\dot{x}_{cart}$ and $\dot{\phi}$) can be measured by two motor-mounted encoders for $\dot{x}_{cart}$ and rate-gyro outputs from MPU 6050 for $\dot{\phi}$. As a result, control output u is in the form of $u_{cp} = K_{cp}x_{cp}$, where $K_{cp} = \ -R_{cp}^{-1}B_{cpo}^T P_{cp}$, and $P_{cp} \geq 0$ is the stabilizing solution of the following Riccati equation (see Section 3.5.

$$
P_{cp}A_{cp0} + A_{cp0}^T P_{cp} - P_{cp}B_{cp0}R_{cp}^{-1}B_{cp0}^T P_{cp} + Q_{cp} = 0 \tag{9.3.10}
$$

For the initial design, we choose an initial set of LQR weighting matrices Q_{cp} and R_{cp} as $Q_{cp} = diag\ [10,\ 10]$ and $R_{cp} = 1$. From

Section 3.5, we know the selection of both weighting matrices Q_{cp} and R_{cp} affects the resulting performance directly. These weighting matrices will be tuned in Section 9.4 to achieve desired performance in simulation studies. For the given weighting matrices, the resulting state feedback control gain is shown below.

$$K_{cp} = \begin{bmatrix} 1.6932 & 2.1195 \\ 1.6932 & -2.1195 \end{bmatrix} \tag{9.3.11}$$

The associated closed-loop system pole locations are listed below. The closed-loop system is asymptotically stable with poles located further to the left of complex plane, comparing with the open-loop system.

$$s_1 = -1.1481, s_2 = -0.0430 \tag{9.3.12}$$

The control law described by equation 9.3.11 only stabilizes the three-wheel cart by regulating its initial states ($\dot{x}_{cart}$ and $\dot{\phi}$) to zero. This LQR strategy cannot drive the three-wheel cart in a way similar to the triple-PID speed controller discussed in Section 9.2. Now let's develop a model-based control strategy utilizing LQTI control method to have capabilities similar to the triple-PID control law. We want the three-wheel cart to track the desired cart speed r_{cart} and turning speed r_{ϕ} simultaneously. We must add two tracking error states to the system state vector defined in 9.3.3. These are $\dot{x}_{ce} = r_{cart} - \dot{x}_{cart}$ (three-wheel cart speed) and $\dot{\phi}_{ce} = r_{\phi} - \dot{\phi}$ (turning speed). A six-state vector $x_{pi} = \left[x_{cart}, \dot{x}_{cart}, \phi, \dot{\phi}, \dot{x}_{ce}, \dot{\phi}_{ce}\right]^T$ can be formed. Note that r_{cart} and r_{ϕ} are cart and turning speed reference signals, respectively. They are also used in Section 9.2 for the triple-PID control strategy to calculate the left and right wheel reference speed signals. As a result, the updated state-space model is now in the following form with $r = [r_{cart},\ r_{\phi}]^T$.

$$\begin{aligned} \dot{x}_{pi} &= A_{pi}x_{pi} + B_{pi}u_{pi} + D_{pi}v + B_i r \\ y &= C_{pi}x_{pi} + H_{pi}u \end{aligned} \tag{9.3.13}$$

with all its input vector u and output vector y defined in equation 9.3.3, disturbance input $v(\theta_z, \theta_y) = \alpha_3(\theta_z, \theta_y)\beta_3(\theta_z, \theta_y)$ defined in the paragraph after equation 9.3.4, and the reference input r defined in the paragraph above equation 9.3.13. The associated system matrices of equation 9.3.13 are defined below.

$$A_{pi} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\dfrac{\alpha_2}{\alpha_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\left(\dfrac{2\alpha_2}{\alpha_1} + \dfrac{\beta_2}{\beta_1}\right)\beta_3 & 0 & -\dfrac{\beta_2}{\beta_1} & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}, \quad B_{pi} = \begin{bmatrix} 0 & 0 \\ \dfrac{K_\tau}{\alpha_1 R_a r_w} & \dfrac{K_\tau}{\alpha_1 R_a r_w} \\ 0 & 0 \\ \dfrac{K_\tau}{R_a r_w}\left(\dfrac{r_v}{\beta_1} - \dfrac{\beta_3}{\alpha_1}\right) & \dfrac{-K_\tau}{R_a r_w}\left(\dfrac{r_v}{\beta_1} + \dfrac{\beta_3}{\alpha_1}\right) \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{9.3.14}$$

$$D_{pi} = \begin{bmatrix} 0 \\ \dfrac{1}{\alpha_1} \\ 0 \\ \dfrac{-1}{\alpha_1} \\ 0 \\ 0 \end{bmatrix}, B_i = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, C_{pi}^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad H_{pi} = \mathbf{0}_{4\times 2} \tag{9.3.15}$$

where parameters α_1, α_2, α_3, β_1, β_2, and β_3 are defined in equations ??? and ??? in Section 9.1. Utilizing the parameter values provided in Table 9.1,we can find the numerical system matrices below for matrices $A_{pi} = A_{pi0} + A_{pi\theta}\beta_3(\theta_z, \theta_y)$ and $B_{pi} = B_{pi0} + B_{pi\theta}\beta_3(\theta_z, \theta_y)$ with output, reference input, disturbance input, and direct feedforward matrices C_{pi}, B_i, D_{pi}, and H_{pi} given in equation 9.3.15.

$$A_{pi0}=\begin{bmatrix}0&1&0&0&0&0\\0&-0.3113&0&0&0&0\\0&0&0&1&0&0\\0&0&0&-0.0023&0&0\\0&-1&0&0&0&0\\0&0&0&-1&0&0\end{bmatrix},\quad B_{pi0}=\begin{bmatrix}0&0\\0.2471&0.2471\\0&0\\0.0096&-0.0096\\0&0\\0&0\end{bmatrix}$$
$$A_{pi\theta}=\begin{bmatrix}A_{p\theta}&\mathbf{0}_{4\times2}\\\mathbf{0}_{2\times4}&\mathbf{0}_{2\times2}\end{bmatrix},B_{pi\theta}=\begin{bmatrix}B_{p\theta}\\\mathbf{0}_{2\times2}\end{bmatrix},D_{pi}=\begin{bmatrix}D_p\\\mathbf{0}_{2\times2}\end{bmatrix} \tag{9.3.16}$$

where matrices $A_{p\theta}$, $B_{p\theta}$, and D_p are defined in equation 9.3.8. The state-space model with dual-error integrators defined in equation 9.3.13 is not suitable for LQTI control design; any stabilization controller will regulate the state vector to zero. With the nonzero cart and turning speed references, the state vector cannot be regulated to zero at steady state. As a result, when designing a three-wheel cart LQTI controller capable of tracking both desired cart speed r_{cart} and turning speed r_{ϕ}, we drop states x_{cart} and ϕ in the state vector and tracking error states ($\Delta\dot{x}_{cart}=\dot{x}_{cart}-r_{cart}$ and $\Delta\dot{\phi}=\dot{\phi}-r_{\phi}$) for state feedback tracking control design with the following system matrices.

$$A_{cpi}=\begin{bmatrix}-0.3113&0&0&0\\0&-0.0023&0&0\\-1&0&0&0\\0&-1&0&0\end{bmatrix},B_{cpi}=\begin{bmatrix}0.2471&0.2471\\0.0096&-0.0096\\0&0\\0&0\end{bmatrix},\quad B_{cpi}=\begin{bmatrix}0&0\\0&0\\1&0\\0&1\end{bmatrix} \tag{9.3.17}$$

where the associated state vector is $\Delta x_{cpi}=\left[\Delta\dot{x}_{cart},\ \Delta\dot{\phi},x_{ce},\ \phi_{ce}\right]^T$. The nominal system state equation is $\Delta\dot{x}_{cpi}=A_{cpi}\Delta x_{cpi}+B_{cpi}u_{cpi}+B_{ci}r$, where $u_{cpi}=\left[\Delta u_{aL},\ \Delta u_{aR}\right]^T=u_{cp}-u_{cp0}$. In addition, $u_{cp0}=\left[u_{aL0},\ u_{aR0}\right]^T$ is the control vector maintaining the cart and turning speeds at the desired levels, and as defined earlier, $u_{cp}=\left[u_{aL},\ u_{aR}\right]^T$ is the control vector for both DC motors.

Because it is a state feedback LQTI design, no output equation is needed. For the continuous-time cart speed tracking control design [9], [10], [11], a state feedback LQTI controller can be designed with the following cost function. All four physical states in x_{cpi} are either directly measurable or the cart speed and turning speed error states can be calculated in real-time within the microcontroller.

$$J_{cpi}=\frac{1}{2}\int_0^{\infty}\left(\Delta x_{cpi}^T(t)\,Q_{cpi}\Delta x_{cpi}(t)+u_{cpi}^T(t)R_{cpi}u_{cpi}(t)\right)dt \tag{9.3.18}$$

where $Q_{cpi}\geq0$ and $R_{cpi}>0$ are the weighting matrices to be selected. As a result, control output u_{cpi} is in the form of $u_{cpi}=K_{cpi}\Delta x_{cpi}$, where $K_{cpi}=-R_{cpi}^{-1}B_{cpi}^TP_{cpi}$, and $P_{cpi}\geq0$ is the stabilizing solution of the following Riccati equation.

$$P_{cpi}A_{cpi}+A_{cpi}^TP_{cpi}-P_{cpi}B_{cpi}R_{cpi}^{-1}B_{cpi}^TP_{cpi}+Q_{cpi}=0 \tag{9.3.19}$$

For the initial design, an initial set of LQR weighting matrices Q_{cpi} and R_{cpi} was chosen as $Q_{cpi}=diag\left[10,\ 10,\ 10^3,\ 10^3\right]$ and $R_{cpi}=1$. The added integration modes (third and fourth states x_{ce} and ϕ_{ce}) are not controllable and they are not stable due to $A_{cpi}(3:4,3:4)=\mathbf{0}_{2\times2}$). Therefore, there is no positive definite solution for Riccati equation 9.3.19. An approximation of the integration is used for LQTI control design by setting $A_{cpi}(3:4,3:4)=-0.001I_{2\times2}=-\varepsilon I_{2\times2}$ ($\varepsilon=0.001$), making the integration model stable. The resulting state feedback gain is shown below with control $u_{cpi}=K_{cpi}\Delta x_{cpi}$.

$$K_{cpi}=\begin{bmatrix}9.1604&48.1422&-22.3509&-22.3124\\9.1604&-48.1442&-22.3509&22.3124\end{bmatrix} \tag{9.3.20}$$

We can see that the control gains shown in 9.3.20 associated with cart and turning speed errors (the first two states) are much greater than these from the LQR design in equation 9.3.11, even though the associated weightings are the same. The closed-loop system pole locations are listed below in equation 9.3.21. Integration action likely causes significant increase of these gains. We can also see that the closed-loop system is asymptotically stable. The two pairs of complex pole locations are also more aggressive, compared with these for these real poles of LQR design. In addition, these two pairs of closed-loop poles have a damping ratio close to $\sqrt{2}/2=0.707$, indicating good step response performance (see Chapter 3).

$$\begin{aligned}s_{1,2}&=-0.4638\pm0.4628i,\\s_{3,4}&=-2.4197\pm2.2794i\end{aligned} \tag{9.3.21}$$

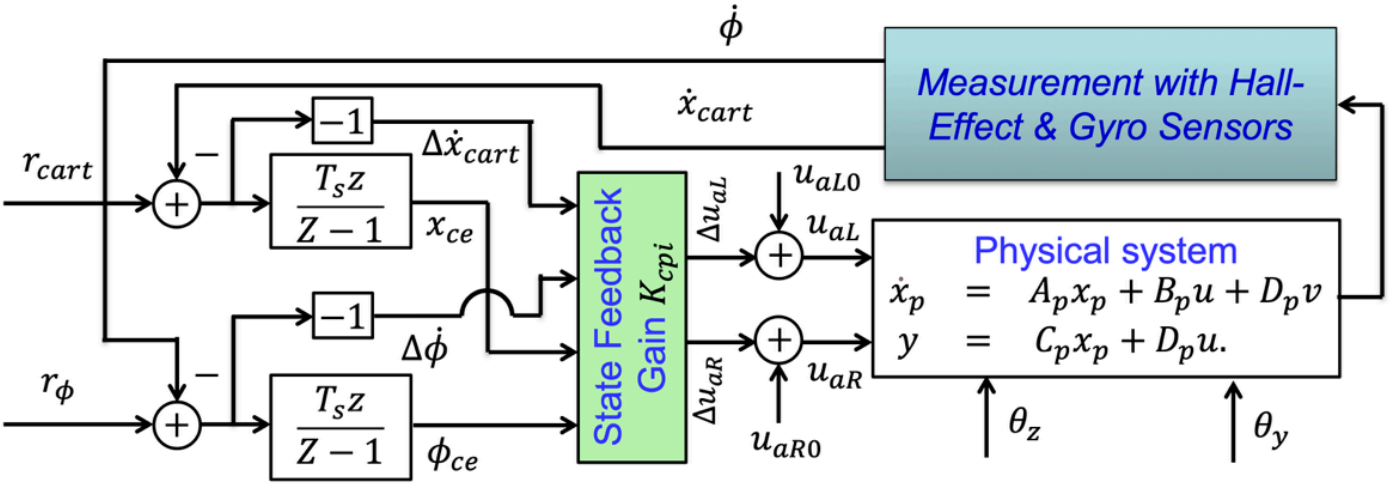


Figure 9.12: The LQTI control system architecture.

Figure 9.12 shows a complete architecture of the three-wheel cart LQTI control system. The physical system (physical domain) is represented by the original state-space model (four states) from equation ??? with physical sensors (MPU 6050 and two wheel encoders) and actuators (two DC motors). The rest of the block diagram in Figure 9.12 is located in the information domain of mechatronic systems. This control system, comparable to the triple-PID control system discussed in Section 9.2, is capable of cart driving and turning. The cart linear and turning speeds are controlled in closed-loop using the LQTI controller outputs $u_{cpi} = [\Delta u_{aL},\ \Delta u_{aR,}\ ]^T$ plus the offset control $u_{cp0} = [u_{aL0},\ u_{aR0,}\ ]^T$, where the combined control is $u_{cp} = [u_{aL},\ u_{aR,}\ ]^T = u_{cp0} + u_{cpi}$. The cart speed signal ($\dot{x}_{cart}$) is calculated by averaging both wheel linear speeds based upon the two encoders installed on the motor shaft (Section 7.3), assuming no slip between wheel and road surface. The turning speed is directly measured by the rate gyro sensor of MPU 6050 in the z_{cart} direction. Ground level angles θ_z and θ_y can also be calculated based the acceleration signals from MPU 6050, assuming that the cart acceleration is small compared with the gravity. Both tracking control integration actions are completed using digital integration of error signals $\Delta\dot{x}_{ce}$ and $\Delta\dot{\phi}_{ce}$.

Section Summary

This section applies the model-based design process (see Section 3.5) to the three-wheel cart modeled in Section 9.1. It utilizes both LQR [5], [6], [7], [8] and LQTI [9], [10], [11] control design techniques. First, it converts the three-wheel cart model, equation ???, into its state-space form. It then populates the form with the parameter values provided in Table 9.1. Next it designs an LQR controller for stabilizing the three-wheel cart. Last, two error integral states of the three-wheel cart and turning speeds are added to the state-space model [9], [11]. An LQTI controller is designed to track the cart's forward/backward and turning speeds. This strategy is a model-based one comparable to the triple-PID control strategy (Method 2) discussed in Section 9.2 with both forward/backward and turning speed controlled in closed-loop. It will be validated in simulation studies in Section 9.4, along with the control design weighting adjustment to optimize controller stability and tracking performance.

Section Keywords

Practice Problems

1. What is the order of the state-space model in equation ???? Select the correct answer.
   a. 4
   b. 5
   c. 6
2. From the state-space model in equation ???, we can conclude if the following statement is true or not. The cart motion state $\dot{x}_{cart}$ and turning speed $\dot{\phi}$ are independent.
   a. True
   b. False
3. To make the three-wheel cart track both cart and turning speed references, two additional error integral states x_{ce} and ϕ_{ce} were created. Are both created states (modes) controllable? Select Yes or No and explain why.

4. Why do the poles associated to the two added states, x_{ce} and ϕ_{ce}, need to be modified to $-\varepsilon$ with $\varepsilon > 0$ so that an LQTI controller can be designed?

This page titled 9.3: Model-Based Control of a Three-Wheel Cart is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

9.4: Simulation Study of PID and Model-Based Control Strategies

Learning Objectives

After completing this section, you will be able to:

1. Build a Simulink simulation model for the three-wheel cart to generate individual wheel speed feedback outputs.
2. Develop the Simulink SiL (software-in-the-loop) environment for the three-wheel car's duel-PID control.
3. Set up the Simulink SiL environment for the three-wheel cart's triple-PID control.
4. Develop the Simulink SiL environment for the three-wheel cart's linear quadratic tracking with integral (LQTI) control.

This section discusses the simulation studies of the three control strategies of the three-wheel cart developed in Sections 9.2 and 9.3. These are dual-PID, triple-PID, and LQTI control designs. In order to conduct simulation studies, Figure 9.13 develops a new Simulink three-wheel cart model. It makes both left and right wheel speed signals available as model outputs so that the dual- and triple-PID control strategies can be implemented.

The Simulink model in Figure 9.13 was developed based upon the state-space model developed in Section 9.3 described by equations ??? and ???. When $\theta_z = 0$, the linear time invariant (LTI) model is described by state-space model matrices A_{p0}, B_{p0}, C_p, and H_p defined in equation ???. When $\theta_z \neq 0$, the effect of the wheel-slip is modeled based upon state-space matrices $A_{p\theta}$, $B_{p\theta}$, and D_p defined in equation ???. The three-wheel cart turning motion is a sum of two motions. One is the speed difference $\dot{\phi}_x$ between left and right wheels, called Phi_dot_Linear in Figure ???. The other is the turning due to wheel lateral-slip $\dot{\phi}_g$ on unleveled ground, called Phi_dot_g in Figure ???. We calculate both left and right wheel speed signals based upon the cart speed and turning motion due to the wheel-speed difference. The Simulink model shown in equation ??? does not have individual wheel speed output.

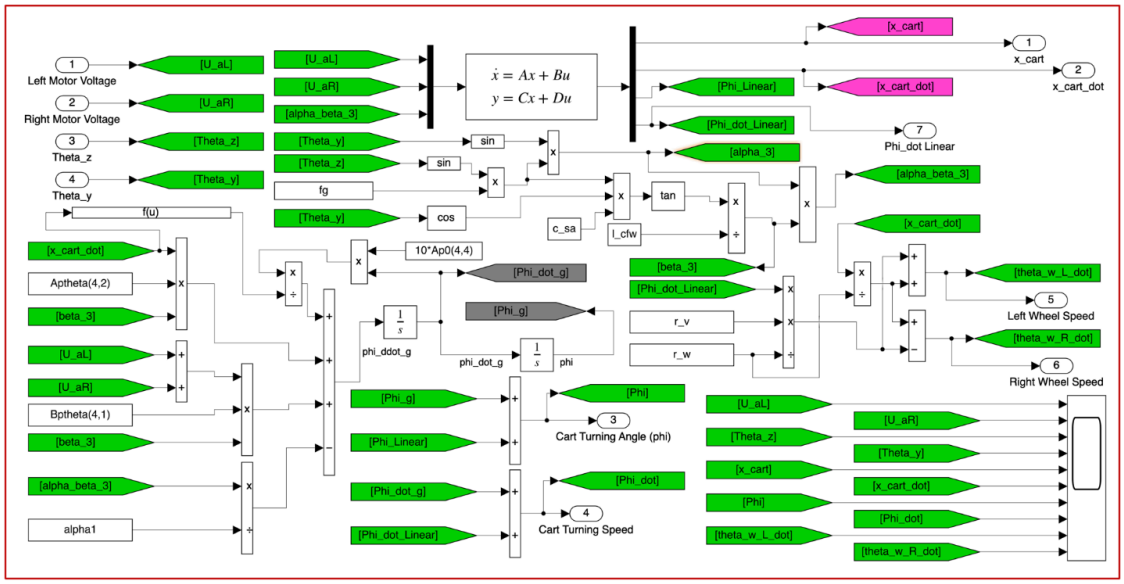


Figure 9.13: Simulink three-wheel cart model for SiL simulations.

The three-wheel cart Simulink model in Figure 9.13 is capable of cart turning angle and speed outputs. Now it is ready to conduct SiL (software-in-the-loop) simulation studies for the dual-PID control strategy. Figure 9.14 shows the SiL simulation Simulink diagram. It includes the details of the model from Figure 9.13 and the two PID controller models from Figure 8.11, with their outputs saturated at ± 12 volts. There are two inputs to the two PID controllers. One is the three-wheel cart speed reference and the other is its turning speed. The three-wheel cart speed reference is converted to individual wheel speed reference by dividing the wheel radius. The turning angular speed is converted to the positive and negative wheel reference signals by dividing two times of wheel radius over the cart turning radius. Both saturation blocks limit the motor armature voltages to ± 12 volts. Zero-

order sample and hold devices are used to send control signals to both DC motors and read feedback signals for both PID controllers with a sample period of 0.01 second. The simulated sixteen outputs are saved to a MATLAB data structure named as "out.mat."

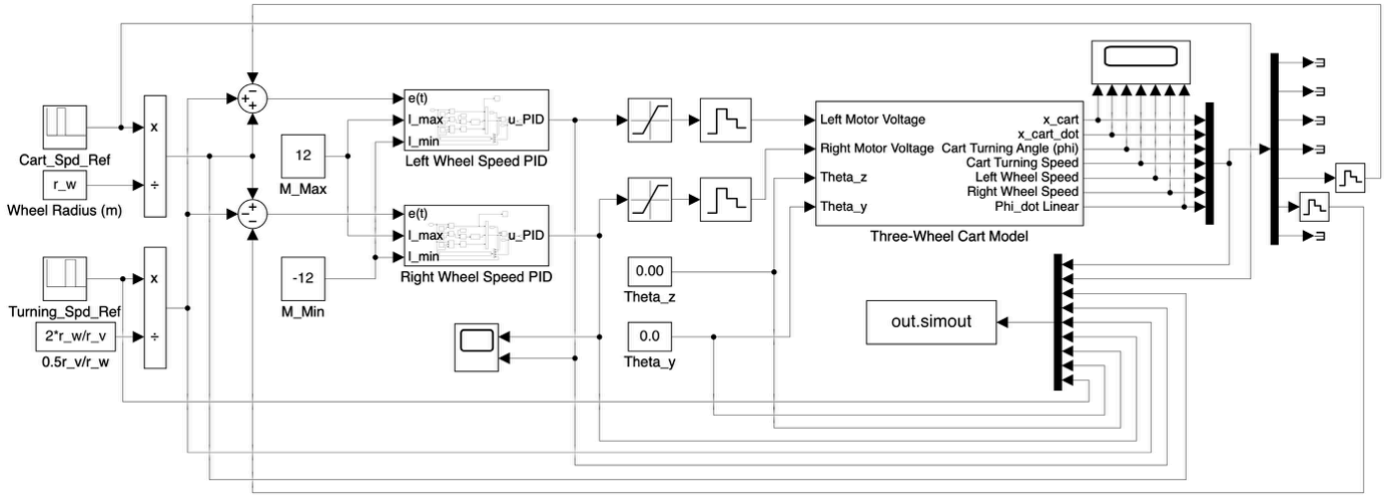


Figure 9.14: Dual-PID SiL simulation environment in a MATLAB/Simulink diagram.

The PID control gains of the left and right wheel speed controllers, shown in Figure 9.14, are carefully tuned. Table 9.2 shows the resulting gains. The gains listed in the last column are not for this study; they will be used later for the triple-PID control strategy.

Table 9.2: PID gains for wheel speed and turning speed PID controllers.

PID type	Left wheel speed	Right wheel speed	Turning speed
Proportional	$K_{pL} = 3$	$K_{pR} = 3$	$K_{pT} = 300$
Integral	$K_{dL} = 0.1$	$K_{dR} = 0.1$	$K_{dT} = 20$
Derivative	$K_{iL} = 1$	$K_{iR} = 1$	$K_{iT} = 100$

For the dual-PID control strategy, we will conduct two simulation studies with the same reference signals for both three-wheel cart speed and turning speed, but with different ground level angles of θ_z and θ_y. For the first study, we set both θ_z and θ_y to zero. That is, there is no wheel-slip due to leveled ground. For the second study, we set θ_z and θ_y to $0.02\ rad$ and zero respectively. Figure 9.15 shows the simulation result for the first study.

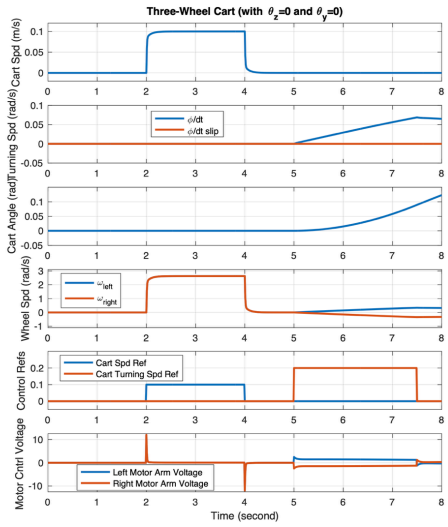


Figure 9.15: SiL simulation with dual-PID control ($\theta_z = \theta_y = 0$).

Figure 9.15 has six subplots. From top to bottom, they are three-wheel cart speed $\dot{x}_{cart}$; cart combined turning speed $\dot{\phi}$ and turning speed $\dot{\phi}_g$ due to wheel lateral-slip; cart angle ϕ; left and right wheel speed signals ($\dot{\theta}_{wL}$ and $\dot{\theta}_{wR}$); cart speed and turning speed references (r_{cart} and $r_{\dot{\phi}}$); and left and right motor control signals (u_{aL} and u_{aR}). We can see that the first two seconds of simulations are with the zero reference speeds for both cart and turning speeds. As a result, all signals shown are at rest (zero). Between the 2nd and 4th second, the card reference speed r_{cart} becomes $0.1\ (m/s)$; see the fifth plot. Held at 0.1, we can see that the car speed accelerates to $0.1\ (m/s)$ and decelerates to zero within about 0.4 second, with similar patterns for both left and right wheel speeds. Large motor control voltages are used during acceleration and deceleration.

Between the 5th and 7.5th second, the card reference turning speed becomes 0.2 (rad/s); see the fifth plot. Held at 0.2, we can see that the car tried to accelerate its turning speed to 0.2 (rad/s), but did not reach its 0.2 (rad/s) target at the 7.5th second. Deceleration starts at the 7.5th second; see the second plot. Both left and right wheel speeds have similar patterns; see the fourth plot. The cart turning angle keeps increasing during this process. Due to relatively slow dynamics of cart turning, the turning response is relatively slow compared with the cart speed. The turning response will be significantly improved by the triple-PID strategy. In addition, during the turning, we can see that the motor control voltages are in opposite directions.

The second simulation of the dual-PID control strategy is for the case with $\theta_z = 0.02\ (\text{rad})$ and $\theta_y = 0$. That is, the ground is slightly unleveled with about 3.6 degrees. PID wheel speed control gains used are the same as these used for the study shown in Figure 9.15 and Table 9.2. The only change is the change of θ_z from zero to $0.02\ (\text{rad})$.

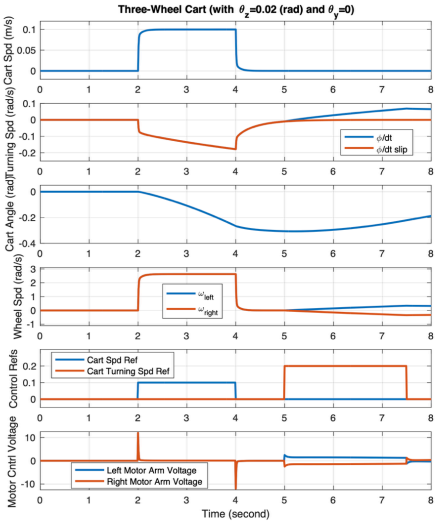


Figure 9.16: SiL simulation with dual-PID control ($\theta_z = 0.02\ rad$ and $\theta_y = 0$).

Figure 9.16 shows the results of this study. It again has six subplots. These are three-wheel cart speed; cart turning speed and turning speed due to wheel lateral-slip; cart angle; left and right wheel speed signals; cart speed and turning speed references; and left and right motor control signals. The first two seconds of simulations are with the zero-reference speed for both cart and turning speeds. As a result, all signals simulated are at rest (zero). Between the 2nd and 4th second, the card reference speed becomes $0.1\ (m/s)$; see the fifth plot. Held at 0.1, we can see that the car speed accelerates to 0.1 (m/s) and decelerates to zero within about 0.4 second with similar patterns for both left and right wheel speeds. Similar to the case of $\theta_z = 0$, we can see large transient motor control voltages at the beginning of acceleration and deceleration. Unlike the case of $\theta_z = 0$ in Figure 9.15, and due to $\theta_z = 0.02$ (rad), wheel lateral-slip occurs between the 2nd and 4th second, peaks at the 4th second (about $-0.18\ rad/s$), and recovers after the 4th second (or between the 4th and 5th) due to reduced cart speed.

Between the 5th and 7.5th second, the card reference turning speed becomes 0.2 (rad/s); see the fifth plot. Held at 0.2, we can see that the car turn speed accelerates to 0.2 (rad/s) slowly, but does not reach its target. Deceleration starts at the 7.5th second; see the second plot. There are opposite patterns for the left and right wheel speeds. see the fourth plot. The cart turning angle keeps increasing. Similar to the previous case, the turning response is relatively slow compared with the cart speed due to relatively slow dynamics of the cart turning.

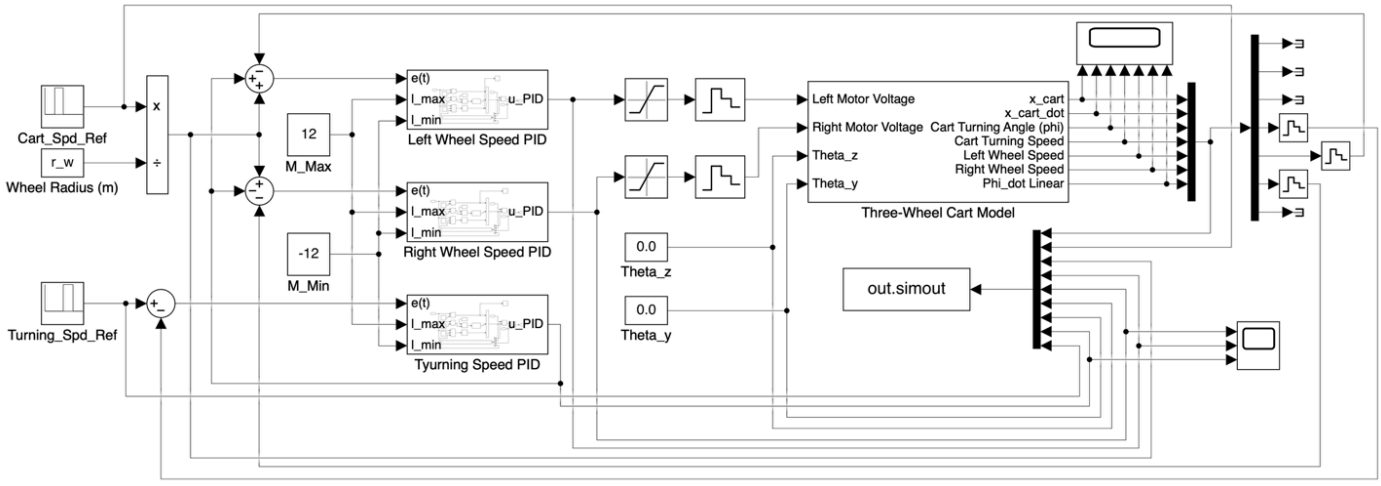


Figure 9.17: Triple-PID SiL simulation environment in MATLAB/Simulink diagram.

Using the three-wheel cart Simulink model developed in Figure 9.13, we can now conduct an SiL simulation for the triple-PID control strategy. Figure 9.17 shows the Simulink diagram. The details of the three-wheel cart model were shown in Figure 9.13. The

two PID controller models used for controlling both left and right motors have their output saturated at ± 12 volts (see Figure 8.11). An additional PID controller is added. Its cart turning speed error is input and its corrected wheel speed difference is output; see Figure 9.11 for control strategy architecture.

The three PID controllers have two inputs. One is the three-wheel cart speed reference and the other is its turning speed; see the left side of Figure 9.17. The three-wheel cart speed reference is converted to individual wheel speed reference by dividing the wheel radius. The turning reference speed is used by the third PID controller directly to generate the turning speed error. Both voltage saturation blocks limit the motor armature voltages to ± 12 volts. Zero-order sample and hold devices are used for control signals going to the DC motors and three feedback signals going to the PID controllers with a sample period of 0.01 second. The simulated fifteen outputs are saved to a MATLAB data structure named as "out.mat."

For the triple-PID control strategy, we will conduct two simulation studies with the same reference signals for both three-wheel cart speed and turning speed, but with different ground level angles of θ_z and θ_y. For the first study, both θ_z and θ_y are set to zero. There is no wheel lateral-slip due to leveled ground. For the second study, both θ_z and θ_y are set to 0.02 rad and zero, respectively.

Figure 9.18 shows the simulation result for the first study. It has six subplots. From top to bottom, they are three-wheel cart speed; cart turning speed and turning speed due to wheel lateral slip; cart angle; left and right wheel speed signals; cart speed and turning speed references; and left and right motor control signals. As before, we can see that the first two seconds of simulations are with the zero reference speeds for both cart and turning speeds. As a result, all signals shown are at rest (zero).

Between the 2nd and 4th second, the card reference speed becomes 0.1 (m/s); see the fifth plot. Held at 0.1, we can see that the car speed accelerates to 0.1 (m/s) and decelerates to zero within about 0.3 second, with similar patterns for both left and right wheel speeds. Large motor control voltages are used at the beginning of acceleration and deceleration. Between the 5th and 7.5th second, the card reference turning speed becomes 0.2 (rad/s); see the fifth plot. Held at 0.2, we can see that the car turning speed accelerates to 0.2 (rad/s) and reaches its target in about one second. Deceleration starts at the 7.5th second (second plot from top) with an opposite pattern for left and right wheel speeds (fourth plot from top). The turning response is much fast than the dual-PID control case, but is relatively slow compared with the cart speed due to slow dynamics of cart turning. In addition, during the turning, the motor control voltages are in the opposite direction and saturated at the beginning of acceleration and deceleration (see the bottom plot).

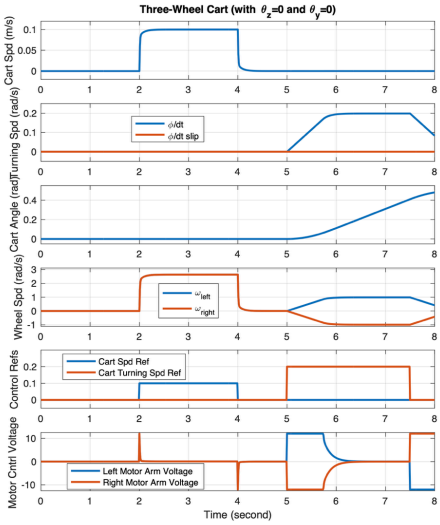


Figure 9.18: SiL simulation with triple-PID control ($\theta_z = \theta_y = 0$).

The second simulation of the triple-PID control strategy is for the case with $\theta_z = 0.02\ (\mathrm{rad})$ and $\theta_y = 0$. That is, the ground is slightly unleveled with about 3.6 degrees. Table 9.2 shows the gains of the three PID controllers used in this study. The only change is the change of θ_z from zero to $0.02\ (\mathrm{rad})$.

Figure 9.19 shows the results of this study. It has six subplots. They are from top to bottom three-wheel cart speed; cart turning speed and turning speed due to wheel lateral slip; cart angle; left and right wheel speed signals; cart speed and turning speed references; and left and right motor control signals. The first two seconds of simulations are with the zero-reference speed for

both cart and turning speeds. As a result, all signals simulated are at rest (zero). Between the 2nd and 4th second, the card reference speed becomes 0.1 (m/s); see the fifth plot. Held at 0.1, we can see that the car speed accelerates to 0.1 (m/s) and decelerates to zero within about 0.5 second with similar patterns for both left and right wheel speeds. Due to wheel lateral-slip, the cart speed response becomes slow. Similar to the case of $\theta_z = 0$, large transient motor control voltages can be observed at the beginning of acceleration and deceleration.

This is different from the case of $\theta_z = 0$. Because $\theta_z = 0.02$ (rad), wheel-slip occurs between the 2nd and 4th second; see the red-line in the second plot. It recovers after the 4th second (or between the 4th and 5th). However, due to the third PID controller, the overall cart turning speed remains close to zero (peak at $0.03\ rad/s$) between the 2nd and 5th second. The associated turning angle is close to zero; see the third plot. Left and right wheel speeds are quite different to compensate for the wheel slip. Between the 5th and 7.5th second, the card reference turning speed becomes 0.2 (rad/s); see the fifth plot. It is held at 0.2. We can see that the car turning speed is accelerating to 0.2 (rad/s) within one second and decelerating at the 7.5th (2nd plot) with similar patterns for both left and right wheel speeds (fourth plot). The cart turning angle keeps increasing. In this case, the turning response remains relatively close to the cart speed due to closed-loop control.

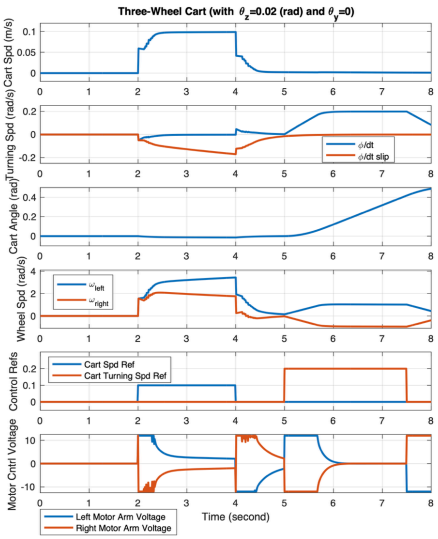


Figure 9.19: SiL simulation with triple-PID control ($\theta_z = 0.02\ rad$ and $\theta_y = 0$).

Our last simulation study is for the model-based LQTI strategy presented in Section 9.3. See equation ??? for the state feedback gain and equation ??? for the associated closed-loop pole locations. We found that the LQTI control gain in equation ??? is small and cannot reach desired performance. As a result, the set of LQTI weighting matrices Q_{cpi} and R_{cpi} was adjusted to $Q_{cpi} = diag\ \left[10^4,\ 10^6,\ 10^3,\ 10^3\right]$ and $R_{cpi} = 1$ with $A_{cpi}\,(3:4, 3:4)$ set to $A_{cpi}\,(3:4, 3:4) = -0.001 I_{2\times 2} = -\varepsilon I_{2\times 2}$. Thus, a positive definite solution for Riccati equation ??? exists. Large weightings on the cart and turning speed errors (q_{11} and q_{22}) are used to increase proportional gains associated with these errors. The resulting state feedback gain is shown below with control $\Delta u_{cpi} = -K_{cpi}\Delta x_{cpi}$.

$$K_{cpi} = \begin{bmatrix} 70.7185 & 708.5808 & -22.2894 & -21.6632 \\ 70.7185 & -708.5808 & -22.2894 & 21.6632 \end{bmatrix} \tag{9.4.1}$$

The control gains in equation 9.4.1 are significantly increased, compared with these shown in equation ???. These weighting coefficients in matrix Q_{cpi} are manually adjusted based on the simulation results.

The associated closed-loop system pole locations are listed in equation 9.4.2. The closed-loop system is asymptotically stable. Poles s_1 and s_2 are associated with the closed-loop turning dynamics, and s_3 and s_4 are associated with the closed-loop driving dynamics. The four real-pole locations are comparable with those for the LQTI design in equation ??? with two very fast pole locations, indicating that the closed-loop system is stable.

$$
\begin{aligned}
s_1 &= -0.0316, \\
s_2 &= -0.3162 \\
s_3 &= -13.5764, \\
s_4 &= -34.9452
\end{aligned}
\tag{9.4.2}
$$

This study utilizes the same three-wheel cart Simulink model from Figure 9.13. The simulation results are comparable with the dual- and triple-PID control strategies. For the LQTI controller implementation, we use a sample rate of $100\ Hz$ (or sample period of 0.01 second) for both control inputs and feedback measurements. Zero-order sample and hold devices are used. For the LQTI control strategy implementation, these four feedback states are multiplied by control gain K_{cpi}. The first two states are formed by cart and turning speed signals, sampled at $100\ Hz$ from the three-wheel cart model, minus their references. We calculate the other two states (integrals of cart and turning speed signal) as part of the control strategy using digital integration logics. In the integration calculation, coefficients K and T_s are set to 1.0 and 0.01 second, respectively; see Figure 9.20 for the digital integration blocks. The feedforward control signals u_{aL0} and u_{aR0} are set to zero. Nonzero feedforward control will be used in experimental validation in Section 9.6. Figure 9.20 shows the entire Simulink block diagram. The simulated data are saved into a MATLAB data structure in a data file called "out.mat." To ensure that both motors are within their operational ranges, motor armature voltages (control outputs) are saturated to ± 12 volts.

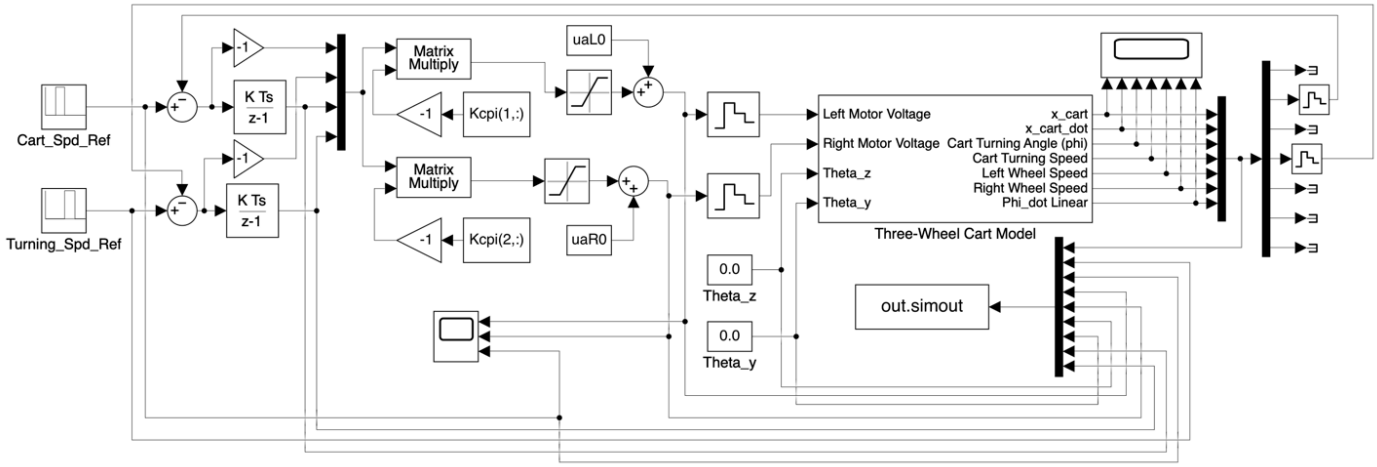


Figure 9.20: SiL simulation with LQTI control.

For the LQTI control strategy, we will conduct two simulations with the same reference signals as those for the triple-PID strategy, but with different ground level angles of θ_z and θ_y. For the first study, both θ_z and θ_y are set to zero. That is, there is no wheel lateral-slip due to unleveled ground. For the second study, both θ_z and θ_y are set to $0.02\ rad$ and zero, respectively.

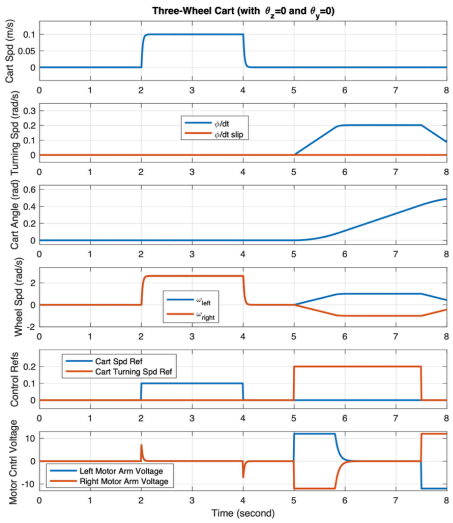


Figure 9.21: SiL simulation with LQTI control ($\theta_z = \theta_y = 0$).

Figure 9.21 shows the simulation result for the first study. The figure has six subplots. From top to bottom, they are three-wheel cart speed; cart turning speed and turning speed due to wheel lateral slip; cart angle; left and right wheel speed signals; cart speed and turning speed references; and left and right motor control signals. As before, we can see that the first two seconds of simulations are with the zero reference speeds for both cart and turning speeds. As a result, all signals shown are at rest (zero). Between the 2nd and 4th second, the card reference speed becomes 0.1 (m/s); see the fifth plot. Held at 0.1, we can see that the car speed accelerates to 0.1 (m/s) and decelerates to zero within about 0.15 second response time with similar patterns for both left and right wheel speeds. This response is faster than the triple-PID case of 0.3 second (see Figure 9.18) mainly due to the coordinated control of both three-wheel cart speed and its turning rate and high control gain. This fast response is achieved with smaller transient armature voltage peak and valley of both DC motors than those of the triple-PID strategy.

Between the 5th and 7.5th second, the card reference turning speed becomes 0.2 (rad/s); see the fifth plot. Held at 0.2, we can see that the car turning speed accelerates to 0.2 (rad/s). It reaches its target in about one second, which is similar to the triple-PID control case. In this case, both motor armature voltages are saturated with the best performance possible. The turning deceleration starts at the 7.5th second; see the second plot. It has similar patterns for both left and right wheel speeds; see the fourth plot. There is no overshoot for both cart speed and turning rate responses due to all real closed-loop system poles. Similar to the triple PID case, during the turning, the motor control voltages are in the opposite direction and saturated at the beginning of acceleration and deceleration (see the bottom plot). In addition, both cartwheels spin in the opposite direction during the turning, as expected.

The second simulation of the LQTI control strategy is for the case with $\theta_z = 0.02\ (\text{rad})$ and $\theta_y = 0$ with the same LQTI controller shown in equation $9.4.1$. That is, the ground is slightly unleveled with about 3.6 degrees. The only change made in this simulation is the change of θ_z from zero to $0.02\ (\text{rad})$.

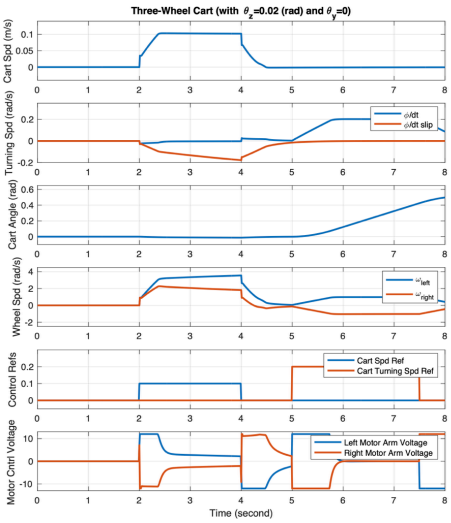


Figure 9.22: SiL simulation with triple-PID control ($\theta_z = 0.02\ rad$ and $\theta_y = 0$).

Figure 9.22 shows the simulation results of this study. It has six subplots. They are, from top to bottom, three-wheel cart speed; cart turning speed and turning speed due to wheel lateral slip; cart angle; left and right wheel speed signals cart speed and turning speed references; and left and right motor control signals. The first two seconds of simulations are with the zero-reference speed for both cart and turning speeds. As a result, all signals simulated are at rest (zero). Between the 2nd and 4th second, the card reference speed becomes $0.1\ (m/s)$; see the fifth plot. Held at 0.1, we can see that the car speed accelerates to $0.1\ (m/s)$ and decelerates to zero within about 0.5 second with similar patterns for both left and right wheel speeds. Due to wheel lateral-slip, the cart speed response slows from 0.15 to 0.5 second (θ_z= 0.02), which is comparable to the triple-PID case. Different from the case of $\theta_z = 0$, due to θ_z= $0.02\ (\text{rad})$, wheel lateral-slip occurs between the 2nd and 4th (see red line in the second plot) and recovers after the 4th second (or between4th and 5th), resulting in large opposite motor controls used to compensate the wheel-slip; see the bottom plot. Here, the overall cart turning speed remains close to zero (peak at $0.015\ rad/s$ and smaller than that of triple-PID case) between 2nd and 5th second. The associated turning angle remains close to zero; see the third plot. Compared with the triple-PID case (see Figure 9.19), the LQTI control uses less control effort at the beginning (between 2nd and 2.35th second) with smaller turning speed error magnitude than the triple-PID case. This shows the advantage of the LQTI control strategy.

Between the 5th and 7.5th second, the card reference turning speed becomes 0.2 (rad/s); see the fifth plot. We can see that the car turning speed accelerates to its target of 0.2 (rad/s) within 0.8 second (similar to the triple-PID case). Deceleration starts at the 7.5th second (second plot) with similar patterns for both left and right wheel speeds (fourth plot). The cart turning angle keeps increasing. The turning response is much slower than the cart speed due to slow closed-loop pole locations associated with the turning motion.

Section Summary

This section conducts simulations to validate the designed controllers, including dual-PID control (Method 1), triple-PID control (Method 2), and LQTI control strategies. The dual-PID control strategy is validated with leveled and unleveled ($\theta_z \neq 0$ and $\theta_y \neq 90^\circ$) ground. The triple-PID control strategy is validated next with unleveled ground, showing it is able to compensate for lateral gravity force over the dual-PID strategy. The LQTI strategy is then validated with improved performance over the triple-PID strategy. Both triple-PID and LQTI strategies can compensate for wheel lateral-slip introduced by lateral force due to unleveled ground surface. Without compensation, the cart will move to the side even though the two wheels have the same linear speed due to unleveled ground. As a summary, without wheel lateral-slip, these three control strategies present close performance, where the LQTI is the best. However, with wheel lateral-slip, the LQTI performs slightly better than the triple-PID with dual-PID strategy being the worst. In addition, the LQTI control uses less control effort than the triple-PID. The main reasons that the LQTI strategy outer-performs the triple-PID strategy are coordinated control of cart and turning speeds as well as optimal control to minimize the weighted control effort and performance. Note that LQTI strategy can handle the coupled dynamics better than the dual- and triple-PID controllers, but the control design weight selection requires certain experience to handle it.

Section Keywords

Practice Problems

1. The Simulink three-wheel cart model in Figure 9.9 cannot be used for SiL simulation study. Select the answer that correctly explains why it cannot be used.
   a. The model is nonlinear.
   b. The model does not have two driving wheel speed outputs.
   c. The model does not consider the wheel lateral-slip dynamics.
2. What is the main disadvantage of the dual-PID control strategy? Select the correct answer.
   a. The strategy cannot track the reference cart speed.
   b. The strategy has a slow cart turning response.
   c. The strategy cannot compensate for wheel slip motion.
3. How can the triple-PID controller compensate for the wheel-slip dynamics? Select the correct answer.
   a. Two wheel-speed PID controllers are used to compensate for the wheel-slip motion.
   b. The third PID controller takes the turning speed error as input. It then generates difference wheel speed reference, used to compensate wheel-slip motion, for the two-wheel speed PID controllers.
   c. The third PID controller takes the turning speed error as input. It then controls left and right DC motors directly.
4. Why does the LQTI controller have better performance than the triple-PID controller? Select the correct answer.
   a. The LQTI design weighting matrix Q_{cpi} is not optimized.
   b. The slow response of the sequential PID controllers.
   c. The LQTI control weighting is optimized with coordinated control for both cart and turning speeds.

This page titled 9.4: Simulation Study of PID and Model-Based Control Strategies is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

9.5: Experimental Investigation of PID Control Strategies

Learning Objectives

After completing this section, you will be able to:

1. Develop the real-time Simulink software for both dual- and triple-PID control strategies of the three-wheel cart.
2. Describe the electronic hardware architecture of the three-wheel cart.
3. Record the real-time test data in the Simulink environment.

This section mainly addresses the experimental aspects of dual- and triple-PID control strategies for the three-wheel cart. These include the development of real-time control software in MATLAB/Simulink with modified PID control and driving logics; test results for the dual-PID control strategy; and triple-PID control strategy. The electronic hardware for three-wheel cart control systems is similar to that shown in Figure 8.34, but with a different battery (SLA 1105, 12V, 12 Ah). See Figure 9.23; detailed descriptions can be found in Section 8.7.

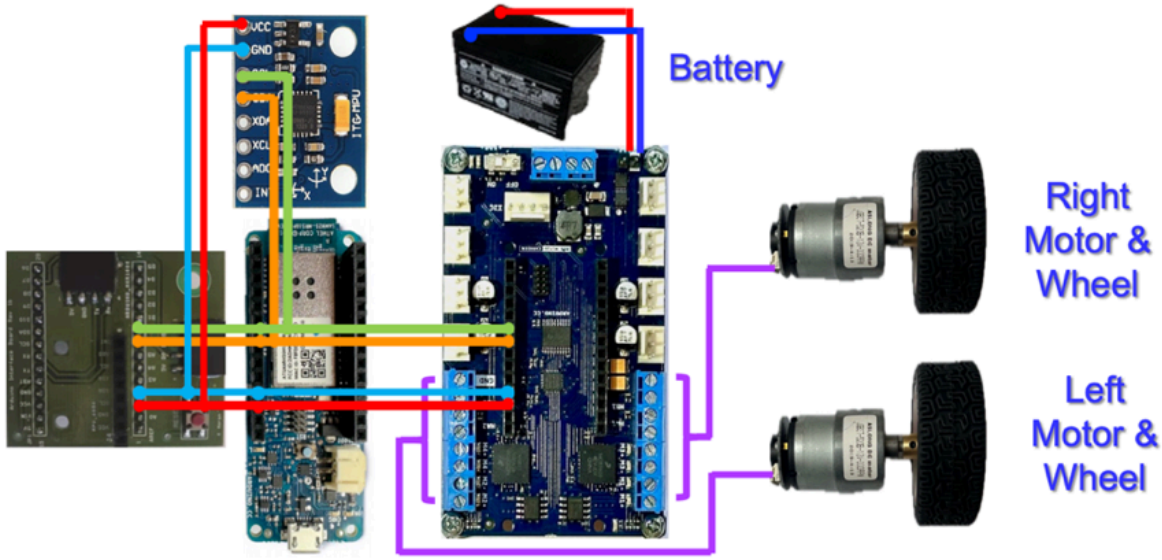


Figure 9.23: Architecture of a three-wheel cart electronic system.

Before validating dual- and triple-PID control strategies for the three-wheel cart, we need to develop a Simulink model for real-time control application to the Arduino microcontroller. Section 9.2 discusses both dual- and triple-PID control strategies. Section 9.4 presents their SiL simulation validation results, along with their Simulink control architectures developed based upon the SiL Simulink model from Section 9.1. Figures 9.10 and 9.11 show the dual- and triple-PID control strategy diagrams shown, respectively, while Figures 9.14 and 9.17 show the associated Simulink simulation environments, respectively. We can thus develop in Simulink a real-time Arduino-based control software with its block diagram; see Figure 9.24.

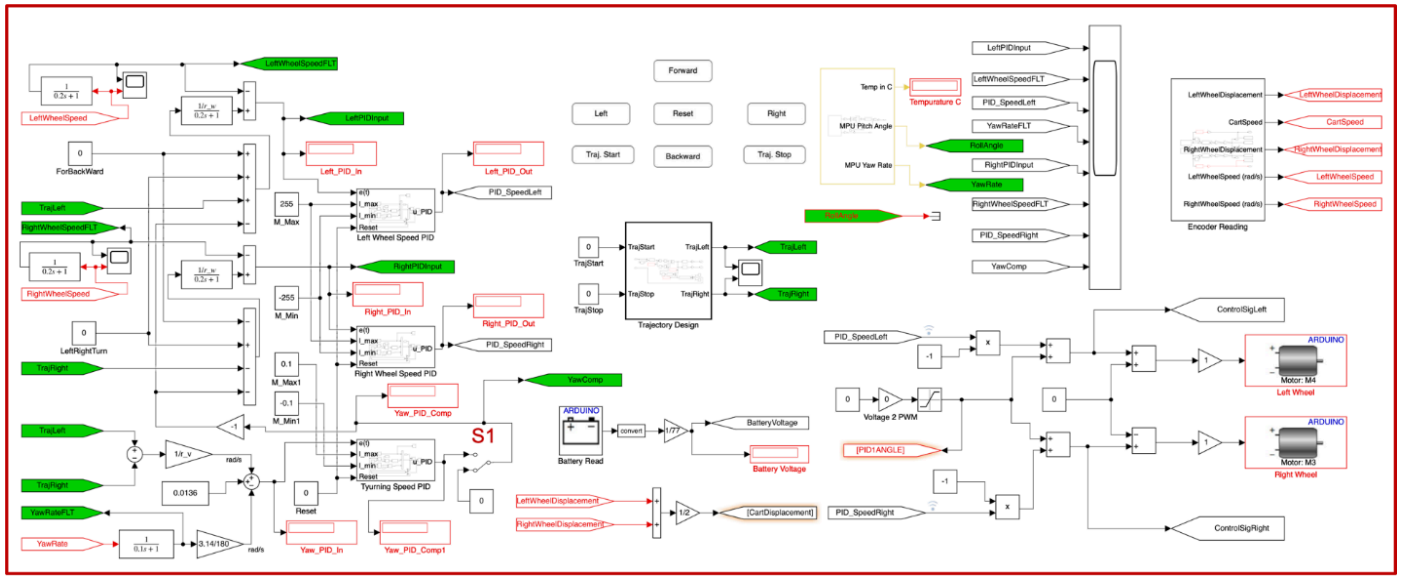


Figure 9.24: Real-time PID Simulink three-wheel cart control model.

Figure 9.24 shows a control system architecture similar to the one in Figure 8.41 for the mini-Segway. The Simulink control software runs inside the host computer, the Arduino MKR 1000 Wi-Fi collects the sensor data, sending it to the host computer as the feedback signals at a rate of 100 Hz via Wi-Fi. The host computer sends the motor control commands at a rate of 100 Hz to the Arduino Wi-Fi, which executes the motor control command based on received control commands through the Arduino MKR motor carrier board.

Figure 9.24 has three PID controllers. The left and right motor speed PID controllers are the same used in simulation with added integration reset logic to combat the DC-motor gear-box friction; see Figure 9.25. The digital differential and integral operations are used, and the anti-windup logic is used for all PID controllers. Due to the relatively large gearbox friction, the DC-motor actuator could remain stopped even though the control PWM duty-cycle is not zero. Even with the PID control error close to zero due to nonzero integration gain, the PID control output may not return to zero. This leads to undesirable heat generation for the DC-motor actuator. The added manual reset logic can set the integration value to zero after the experiment using the push button "Reset."

The left half of Figure 9.24 consists of mainly three PID controllers: the turning speed PID (bottom), and the left and right cart speed PID controllers (top and middle). Figure 9.25 shows the Simulink models of the left and right PID controllers. The bottom turning PID compensator is the same as those for the left and right speed ones. The S1 switch on the right side of the turning PID controller switches the cart turning compensation control on and off to realize triple- and dual-PID strategies, respectively. When the switch is at the turn-on position, the output of the turning speed PID controller is sent to both left and right PID controllers as the reference compensation inputs, resulting in triple-PID strategy. When the switch is at the off position, the turning speed compensation is set to zero, leading to the dual-PID strategy. All three PID controllers are anti-windup with outputs saturated to ± 255 counts for both left and right wheel speed PID controllers and to $\pm 0.1\ (rad/s)$ for the turning compensation PID controller.

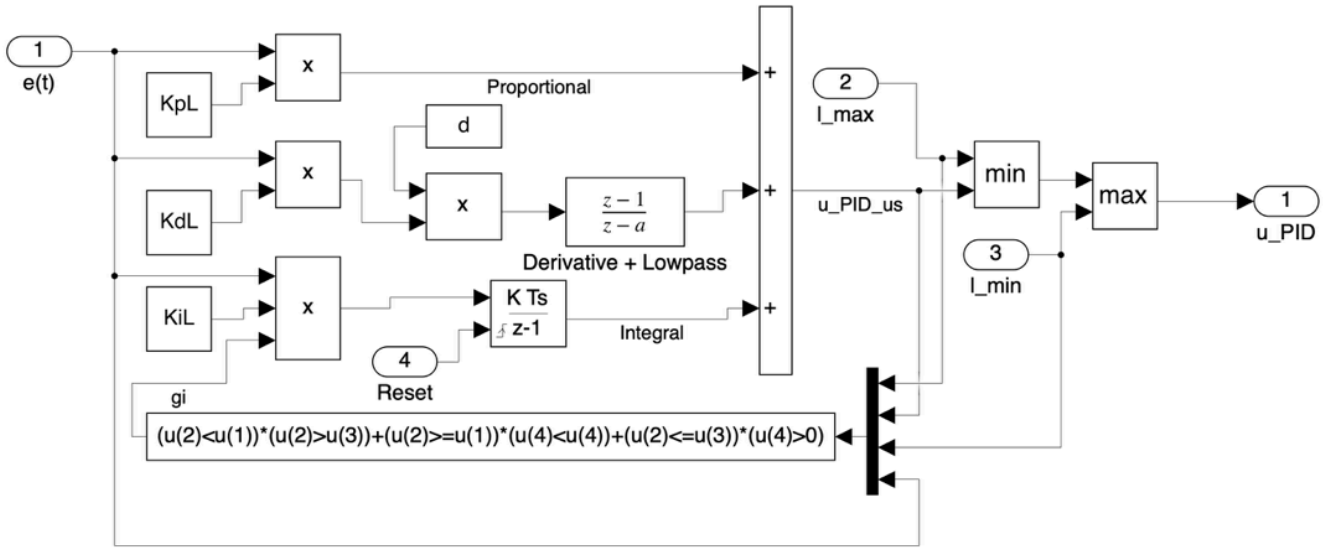


Figure 9.25: Anti-windup PID controller with manual integration reset logic.

Four reference signals can be used for reference inputs of both left and right wheel speed PID controllers. They are forward/backward driving commanded by the associated push buttons (Forward and Backward); left/right turn commanded by the associated push buttons (Left and Right); scheduled motion provided by the block right below these seven buttons; and turning compensation input. The scheduled motion is enabled and disenabled by the "Traj. Start" and "Traj. Stop" push buttons, respectively. In the Simulink model, these push buttons set the link Simulink constant blocks to the calibrated values. For example, the Forward push button sets the "ForBackWard" constant to its calibrated value of $-0.15\ (m/s)$.

These feedback signals often contain measurement noises. Low-pass filters with the following transfer function reduce measurement noises of wheel speed feedback signals.

$$G_{LF}(s) = \frac{1}{0.2s+1} \tag{9.5.1}$$

where the lowpass filter cutoff frequency $\omega_c = 5\ (rad/s)$, or $f_c = 0.796\ (Hz)$. Note that the leading frequency from equation ??? is $0.31\ (rad/s)$. As a result, the cutoff frequency of the low-pass filter used for wheel speed is more than 10 times the system bandwidth, which is reasonable. During the experiments, we found that the MPU 6050 yaw rate signal has a DC bias that changes as a function of temperature. A constant block is added to get rid of the bias. In addition, a low-pass filter, shown below, is used with double bandwidth of equation 9.5.1 to smooth the signal.

$$G_{LF}(s) = \frac{1}{0.1s+1} \tag{9.5.2}$$

The yaw rate sensor bias can also be calibrated automatically by subtracting the offset value with a calibrated value obtained when the three-wheel cart is not moving. In order to smooth the step reference signal without affecting the driving performance, the same low-pass filter shown in equation 9.5.1 is used for both left and right wheel speed PID controllers. Because the smoothed reference signal has a much high bandwidth compared with the system bandwidth, this low-pass filter will not affect the three-wheel cart performance.

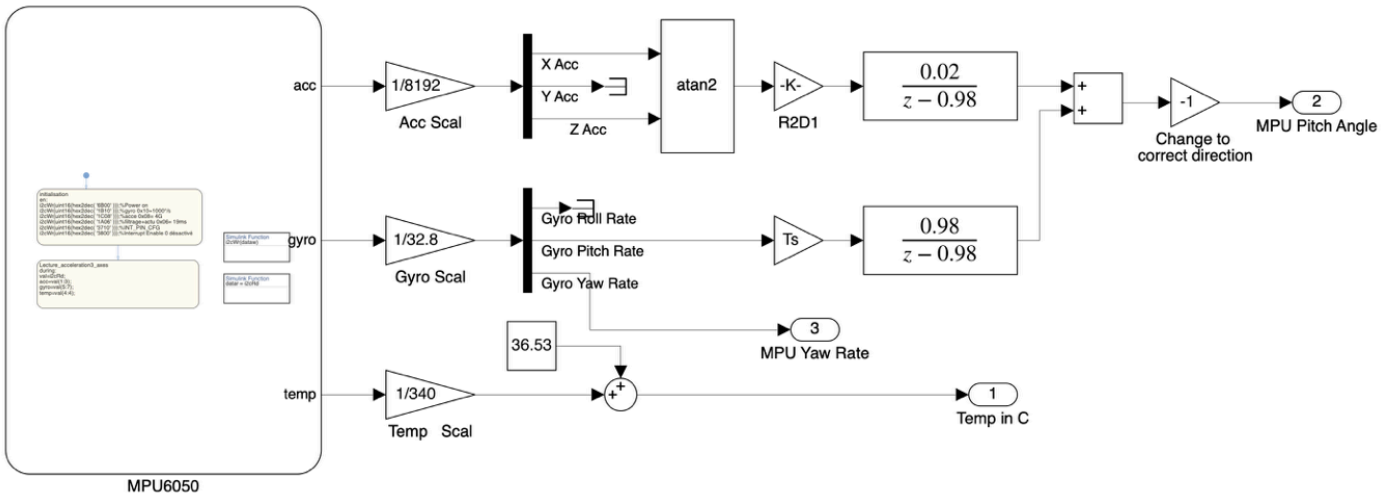


Figure 9.26: Three-wheel yaw rate calculation block.

The measured wheel speed signals of the three-wheel cart are calculated in the block called “Encode Reading,” at the top-right of Figure 9.24 (next to the seven push buttons). This block is the same as the one used in the mini-Segway control algorithm; see Figure 8.39. Therefore, it will not be discussed in detail here. From Sections 9.2 and 9.3, we can see that the yaw rate of the three-wheel cart is needed for feedback control of the three-wheel cart yaw rate. As a result, we must extract the yaw rate gyro signal from the MPU 6050. See Figure 9.26 for the modification of the MPU 6050 signal processing block, where the gyro yaw rate from MPU 6050 is pulled out as the third output of the block.

A new block is added to the Simulink control diagram for the three-wheel cart. Called “Trajectory Design,” it is just below the push buttons in the middle of Figure 9.24. Figure 9.27 shows the Simulink diagram of this block with two inputs (TrajStart and TrajStop) controlled by two associated push buttons (“Traj. Start” and “Traj. Stop”) used to start and stop running the designed vehicle trajectory, respectively.

Figure 9.27 has three groups of Simulink models marked by red-dashed boxes: Floating time generation; Stop trajectory by the Traj. Stop button or reaching max time allowed; and left/right wheel speed reference generation. The basic principle of generating reference signals is to use two lookup tables for providing left and right wheel reference speed signals with a generated floating time signal. The “Floating time generation” block generates a clock signal starting from zero, using a resettable integrator, when the “Traj. Start” button is pushed. This clock signal is used by both left and right wheel speed reference lookup tables as an input signal. The clock signal will be stopped (reset) by the output of the “Stop trajectory by TrajStop signal or max time allowed” block. The reset signal becomes active by the “Traj. stop” button or reaching the maximum time allowed. Based upon the generated enabled clock signal, the “Left and right wheel speed reference generation” block generates two-wheel reference speed signals using two lookup tables. The reference signal can be calibrated by altering two lookup table contents.

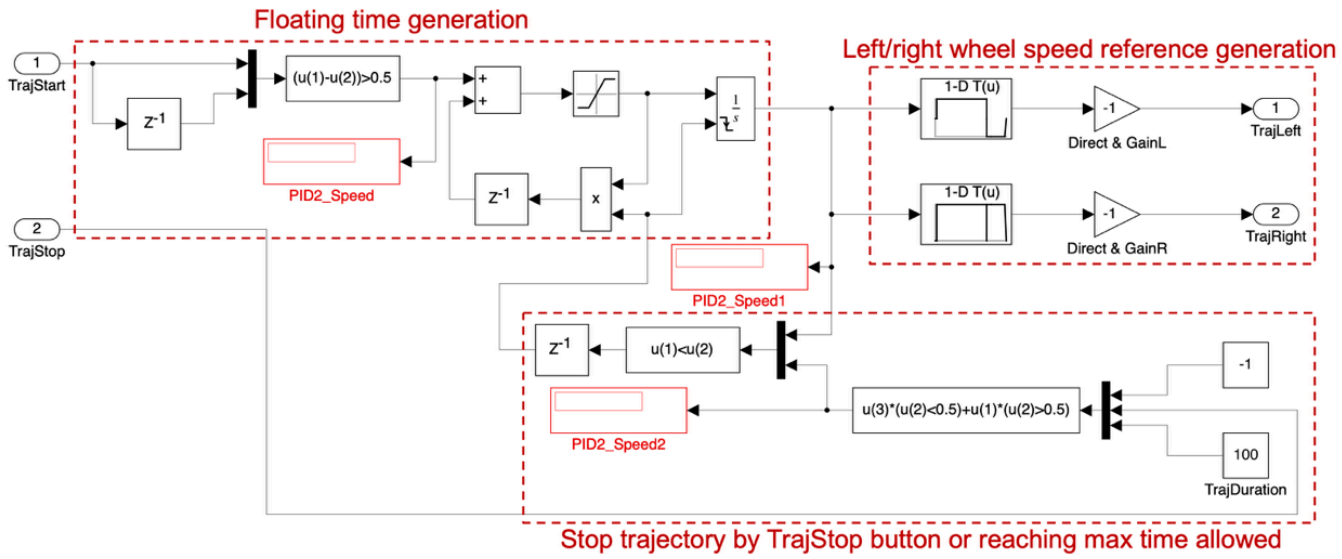


Figure 9.27: Three-wheel cart driving trajectory design block.

With the real-time control Simulink software developed for both dual- and triple-PID control strategies, we are ready for experimental studies. Note that control output for the real-time Simulink model is in counts between -255 and 255, while the simulation software is between -12 and 12 volts, which is more than 20 times difference (or 21.25 to be exact). The first study is for the dual-PID control strategy. Table 9.3 shows the left and right wheel speed PID control gains tuned experimentally. In the simulation studies from Section 9.4, the proportional gain for both left and right wheel speed PID controllers was set to 3 (see Table 9.2). For this experiment, the tuned proportional gain for the dual-PID control strategy shown in Figure 9.10 is 60, which is 20 times the gain used in the simulation study. The two gains used in both simulations and experiments are very close with equivalent gain of $60/21.25$. For the derivative gain, the simulation gain used is 0.1 (see Table 9.2), and the tuned derivative gain for experiments is 1.0, 10 times the simulation gain (0.1). However, considering the gain (21.25) of converting control output from voltage to counts at beginning of the paragraph, the equivalent gain is 0.047 $(1/21.25)$, about half of the simulation one. For the integral gain, the simulation gain used is 1.0 (see Table 9.2). The tuned integral gain for experiments is 30 with an equivalent gain of 1.412 $(30/21.25)$ times the simulation gain (1.0). In short, the differences in PID gains between the simulation and experiment are very small, indicating that the mathematical model is fairly accurate.

Table 9.3: Tuned real-time PID gains for wheel speed and turning speed PID controllers.

PID type	Left wheel speed	Right wheel speed	Turning speed
Proportional	$K_{pL} = 60$	$K_{pR} = 60$	$K_{pT} = 0.24$
Integral	$K_{dL} = 1$	$K_{dR} = 1$	$K_{dT} = 0.02$
Derivative	$K_{iL} = 30$	$K_{iR} = 30$	$K_{iT} = 0.02$

Multiple experiments were conducted for tuning the PID gains of both left and right wheel speed PID controllers. Large proportional or integral gains lead to unstable speed control. In particular, the DC-motor gearbox has very large Coulomb friction, making the control loop fairly nonlinear. As a result, the derivative gain is needed to overcome it. The final tuned PID control gains are very close to those used in the simulation studies, where proportional gain is almost the same, derivative gain is half of the simulation one, and the integral gain is one and half times of the simulation one. The reduced derivative gain could be due to sensor noise. The enlarged integral gain is used to overcome Coulomb friction to reach desired level of motor control PWM duty-cycle quickly.

Figure 9.28 shows the results of the three-wheel cart experiment with a reference signal from the trajectory generation block activated on vertical red dash-dot lines around the 6.8th second. This reference signal starts with a forward drive command of vehicle speed at 0.15 (m/s) for five seconds (see the vertical dash-dotted line). It is followed by a left turn with right wheel linear speed of 0.15 (m/s) and left wheel linear speed of -0.15 (m/s) for two seconds. As a result, the reference three-wheel cart speed is set to zero around the 13.8th second marked by vertical dark-green dash-dot lines.

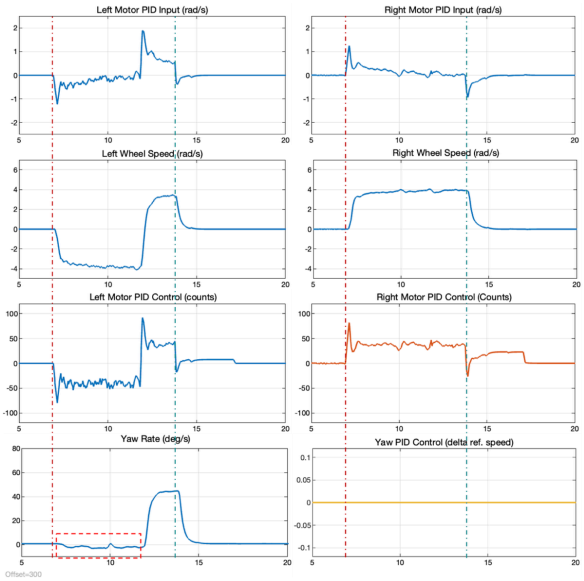


Figure 9.28: Three-wheel cart dual-PID control responses.

The dual-PID control strategy experiment was conducted by setting the switch "S1," to connect to the zero constant block. (See the low-middle part of *Figure 9.24*, left of the battery icon.) *Figure 9.28* shows the time responses of the three-wheel cart with the reference input from the previous paragraph and the dual-PID control gains from *Table 9.3*. Both top plots of *Figure 9.28* present the left and right PID controller input error signals. To move the three-wheel cart forward, the two motors need to run in the opposite direction, with the left in negative and right in positive. For the first 5 seconds, it takes about 2.5 seconds for both wheels to reach the reference target of 3.95 (rad/s). This is the reference cart speed (0.15 m/s) divided by the wheel radius ($0.038\ m$); see the second-row plots of *Figure 9.28*. It takes less than a half second to reach at least 80% of the target speed. The small input errors (peaked at 1.25 rad/s) of both the left and right wheel speed PID controllers are due to the filtered reference signals. From the PID control output signals shown on the third row of *Figure 9.28*, we can see that the magnitudes of both PID control outputs increased significantly to accelerate the cart and then reduced to maintain the cart speed.

For the second part (cart turning) starting around the 11.8th second and ending around the 13.8th second, the reference wheel speed signal for the right motor is changed from 3.95 to -3.95 (rad/s). The reference wheel speed signal for the left motor remains unchanged at 3.95 (rad/s). The small speed dip of the right wheel (second row right plot of *Figure 9.28*) is due to the zero-reference speed between the 16.5th and the 16.52th second from trajectory generation. We can see that after entering the turning phase at the 16.5th second, the left wheel PID input error increases and decreases while the wheel speed changes the direction and tried hard to follow the reference. Meanwhile, the left wheel speed control PID output changes direction to reverse the wheel speed. The right wheel speed remains at its reference one with a small dip at the switch point.

There is no yaw rate tracking for the dual-PID control strategy. Therefore, the yaw compensation inputs to the left and right wheel speed controllers are zero. However, the yaw rate does not remain at zero even during the first phase straight driving due to uneven drive trains of the left and right wheels; see the bottom-left plot of *Figure 9.28*. In the second phase of turning driving between the 11.8th and 13.8th second, the yaw rate increases sharply at the beginning of the turning to track the (yaw) rate target. It then reduces to its initial level when the turning command is terminated at the 13.8th second.

Now, consider the triple-PID control strategies with the switch "S1" in the Simulink real-time control model (see Figure 9.24) connected to inputs of two wheel-speed PID controllers. The same left and right wheel speed PID control gains are used (see Table 9.3). Before conducting these experiments, we must first tune the PID controller gains for yaw-rate compensation (connected to switch "S1"). The tuned gains are shown in Table 9.3. These gains were tuned a little conservatively due to high control gains for both wheel-speed PID controllers. By comparing tuned PID controller gains shown in Tables 9.2 and 9.3, we can see there is about 1250 times difference for proportional gains. This is similar to the case for the dual-PID control strategy. The PID control output for turning compensation in the real-time Simulink strategy is wheel linear speed (m/s), whereas the simulation one (in Figure 9.17) is the wheel angular speed (rad/s). Between these two signals, there is a gain difference of $r_w^{-1} = 26.3$.

In addition, the left and right wheel speed control output unit for simulation is DC-motor battery voltage with a range of ± 12 volts. The real-time Simulink software is in PWM count with a range of ± 255. There is a gain difference of 21.25. Therefore, the total gain difference is 589 (26.3×21.25). As a result, the real-time Simulink software gains of the turning PID controller is about 589 times smaller than the corresponding simulation gains. In this case, the equivalent simulation proportional gain in real-time control

software is 0.51, about double of the tuned gain of 0.24. The equivalent simulation derivative gain is 0.034, about 1.7 times of the tuned gain of 0.02. The equivalent simulation integral gain is 0.17, about 8.5 times of the tuned gain of 0.02. The small integration gain for the real-time software could be due to the double integration action of the sequential PID controllers. In general, real-time software gains are close to those used in simulation studies. Real-time turning PID control gains are about two time smaller than these used in simulation studies.

For the triple-PID control experiment study, the driving reference signals are the same as those used for the dual-PID control strategy. The forward driving is at the 5.9th second with a reference speed of 0.15 (m/s) for five seconds. The turning motion starts at the 10.9th second with a reference angular turning rate of about 90 (deg/s) for two and half seconds. The triple-PID control experimental responses are shown in Figure 9.29. Both Figures 9.28 and 9.29 have the same signal axis ranges for easy comparison. For the first phase (straight forward driving), the left and right wheel-speed PID control outputs (Figure 9.29) are similar to the dual-PID case (Figure 9.28). Both left and right wheel speed responses become even for the triple-PID strategy due to the compensation of the turning PID controller (compare the second rows of Figures 9.29 and 9.28).

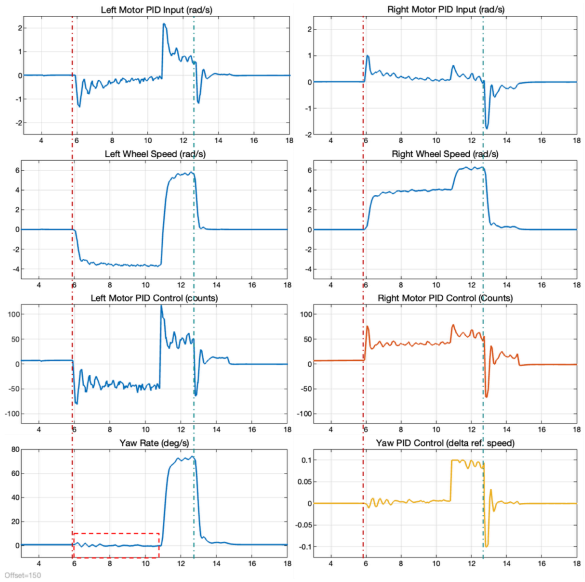


Figure 9.29: Three-wheel cart triple-PID control responses.

For the second phase (turning motion) starting at the 10.9th second, the yaw rate increases sharply and reaches about 72 (deg/s) in about 1.2 second for the triple-PID strategy. Meanwhile, it takes similar time to reach to about 45 (deg/s) for the dual-PID strategy. The smaller regulation error of triple-PID strategy is mainly due to the compensation signal from the turning PID controller; see the bottom-right of Figure 9.29. The compensation signal is saturated to ± 0.1 (m/s) of the linear wheel speed, which is about 6.7% of the wheel speed reference signal magnitude of 0.15 (m/s). Even with this small compensation magnitude, the cart turning rate was increased from 45 to 72 (deg/s). Due to small integration gains and slow turning dynamics, however, the turning speed did not reach its reference level. In this case, a feedforward PID control may be needed.

Figure 9.30 shows zoom-in plots of the yaw-rate responses of the bottom-left plots of Figures 9.28 and 9.29 during the forward straight driving. The red-dashed boxes cover the straight driving duration horizontally and the magnitude vertically. The half height of these boxes centered at zero yaw rate represents the error magnitude.

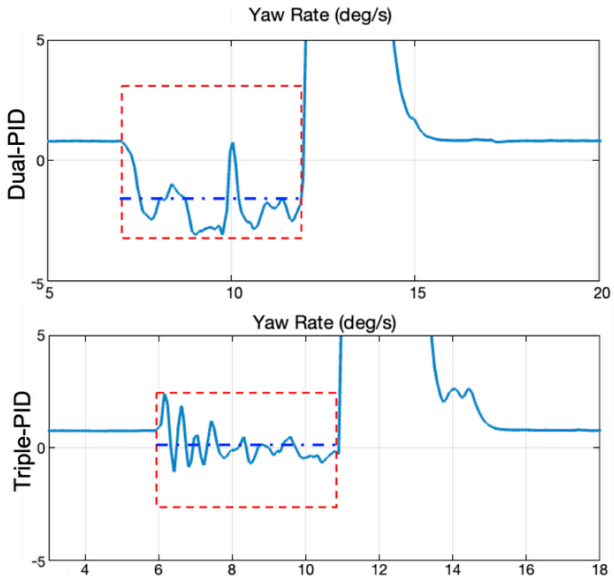


Figure 9.30: Zoomed in yaw rate during straight driving.

Table 9.4 presents the maximum yaw-rate magnitude of dual- and triple-PID control strategies under straight forward driving commands. The dual-PID magnitude is 29.6% larger than the triple-PID one, indicating an improvement of the triple-PID control strategy. In addition, the mean values of the turning rate for dual- and triple-PID strategies are 1.67 and 0.1 deg/s, respectively, where the triple-PID strategy reduces the mean turning rate by 94%. From Figure 9.30, one can see that the turning rate error converges to zero for the triple-PID strategy while the dual-PID strategy has a constant error.

Table 9.4:Turning rate comparison between dual- and triple-PID strategy.

Dual-PID Magnitude	Triple-PID Magnitude	Dual-PID Mean	Triple-PID Mean
3.33 deg/s	2.57 deg/s	1.67 deg/s	0.1 deg/s

The triple-PID control strategy reduces the turning rate error over the dual-PID strategy significantly. As a result, it is able to keep the cart driving at a desired turning rate. In addition, because the simulation model is often oversimplified, the simulation control gains may need to be changed or retuned based upon the experimental setup and real-time software.

Section Summary

This section develops real-time Simulink control software of the dual- and triple-PID control strategies. After the software is implemented into the host laptop, it is debugged and discussed in detail. Both dual- and triple-PID control gains are tuned to achieve the best performance possible. Both dual- and trip-PID control strategies are tested in straight and turning driving conditions on leveled ground surfaces. The test data is collected in the Simulink environment. The section discusses both test data sets in detail, especially reviewing the performance of maintaining the desired turning rate.

Section Keywords

Practice Problems

1. What is the purpose of the low-pass filter in equation $9.5.1$ and what is its corner frequency? Select the correct answer.
   a. Filtering out the wheel speed measurement noisy with a corner frequency of 5.0 Hz
   b. Filtering out the wheel speed measurement noisy with a corner frequency of 0.796 Hz
   c. Smoothing the speed reference signal with a corner frequency of 5.0 Hz
2. What is the purpose of the low-pass filter in equation $9.5.2$ and what is its corner frequency in rad/s? Select the correct answer.
   a. Getting rid of the DC bias in the MPU 6050 yaw rate signal with a corner frequency of 10 Hz
   b. Filtering the MPU 6050 yaw rate signal with a corner frequency of 1.59 Hz

   c. Getting rid of the DC bias in the MPU 6050 yaw rate signal with a corner frequency of 1.59 Hz
3. For the dual-PID strategy, why can't the three-wheel cart keep the turning rate at zero even though the commanded wheel linear speeds are the same? Select all correct answers.
   a. Neither of the wheels of the three-wheel cart are aligned.
   b. The characteristics of two DC motors are different.
   c. The Coulomb friction characteristics of the two DC motors are different.
4. Select the correct answer to fill in the blank in the following sentence.

The triple-PID strategy is able to _____ over the dual-PID one under the straight-forward driving command listed below.

   a. reduce magnitude of turning rate.
   b. reduce mean of turning rate.
   c. reduce both magnitude and mean of turning rate.

This page titled 9.5: Experimental Investigation of PID Control Strategies is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

9.6: Experimental Investigation of Model-Based Control Strategy

Learning Objectives

After completing this section, you will be able to:

1. Implement the model-based control strategy of the three-wheel cart for real-time application.
2. Tune the model-based LQTI controller experimentally.

This section addresses the experimental aspects of the model-based LQTI control of the three-wheel cart. These include the development of real-time control software in MATLAB/Simulink with modified LQTI control and driving logics; and test results for the LQTI control strategies. The electronic hardware for the three-wheel cart control system is the same as that shown in Figure 9.23. However, the real-time control software was redeveloped to implement the model-based LQTI strategy. The software was developed based on the control architecture shown in Figure 9.12. It was then found during initial experiments that the control gains shown in equations ??? and ??? cannot stabilize the three-wheel cart for either straight or turning operation. This is due to very large Coulomb friction torque in the gearbox between the motor and wheel axes. Thus, the control gain needs to be significantly reduced to have a stable closed-loop system. References [8] and [39] provide detailed discussions on the limitation of control gain due to modeling uncertainty. As the control gain increases, its sensitivity to modeling error also increases, leading to unstable closed-loop system. Thus, the LQTI controller is redesigned with weighting matrices $Q_{cpi} = diag\ [10,\ 10,\ 10,\ 5]$ and $R_{cpi} = 1$. The resulting control gain is shown below.

$$K_{cpi} = \begin{bmatrix} 3.1689 & 8.6980 & -2.2323 & -0.6983 \\ 3.1689 & -8.6980 & -2.2323 & 0.6983 \end{bmatrix} \tag{9.6.1}$$

Due to significant reduction of LQTI control gain, the closed-loop system response is relatively slow; see equations 9.6.1 and ??? . To improve closed-loop system response performance, we use the feedforward controls shown in Figure 9.12. In this case, the final control outputs (u_{cp1} and u_{cp2}) to the physical system are defined below.

$$u_{cp} = [u_{aL},\ u_{aR}]^T = u_{cpi} + u_{cp0} = [\Delta u_{aL},\ \Delta u_{aR}]^T + [u_{aL0},\ u_{aR0}]^T \tag{9.6.2}$$

where Δu_{cp1} and Δu_{cp2} are outputs of the LQTI state feedback controller for the left and right motors, respectively; and u_{aL0} and u_{aR0} are feedforward control signals as functions of reference signals r_{cart} and r_{ϕ}, respectively. In the SiL simulation study, both feedforward controls were set to zero. In the controller implementation, the feedforward controls u_{aL0} and u_{aR0} are functions defined by one-dimensional lookup tables that can be calibrated using the steady-state motor armature voltage commands. With the feedforward control architecture, the state feedback LQTI controller needs to compensate only the needed transient control system deviations from its nominal operation condition related to reference signals r_{cart} and r_{ϕ}. As a result, values of both integrators in the LQTI control scheme will be closed zero at the steady state.

With the feedforward LQTI control architecture defined in Figure 9.12, the next step is to develop real-time control software in Simulink for the purpose of conducting experiments. To do so, we modify the triple-PID control software shown in Figure 9.25. Figure 9.31 shows the top-layer software. The reading blocks of rate-gyro sensor and hall-effect wheel-speed sensor are the same as that in the control architecture shown in Figure 9.25, and will not be discussed in this section. Both wheel speed signals are filtered by the same low-pass filter defined in equation ???. It has a corner frequency of 0.8 Hz. The rate-gyro signal is filtered by a low-pass filter defined in equation ??? with a corner frequency of 1.59 Hz. The cart longitudinal speed is calculated in the same way as used for the PID.

We use two reference sources to generate the cart speed reference single r_{cart}: push buttons of “Forward” and “Backward”; and trajectory outputs of “TrajLeft” and TrajRight.” See the left side of Figure 9.31. When the “Forward” button is pushed, a calibrated value is assigned to the “ForBackWard” block. For example, the assigned -0.3 refers to a 0.3 (m/s) forward driving speed. When the “backward” button is pushed, a different calibrated value is assigned to the same “ForBackWard” block. For example, the assigned 0.3 refers to a 0.3 (m/s) backward driving speed. Trajectory outputs, “TrajLeft” and TrajRight,” refer to the left and right wheel linear speed references. The average of these two signals is used to generate the linear cart speed reference single r_{cart}.

Two reference sources are used for generating the cart turning reference single r_ϕ: push buttons of “Left” and “Right;” and trajectory outputs of “TrajLeft” and TrajRight.” See the left side of Figure 9.31. When the “Left” button is pushed, a calibrated value is assigned to the “LeftRightTurn” block in Figure 9.31. For example, the assigned -0.2 refers to a 0.2 (rad/s) turning reference speed. When the “Right” bottom is pushed, a different calibrated value is assigned to the same “LeftRightTurn” block. For example, the assigned 0.2 is referred to a -0.2 (rad/s) turning speed. The difference of trajectory outputs “TrajLeft” and TrajRight” are used to calculate the turning speed reference signal r_ϕ by dividing the half distance (r_v) between the two driving wheels. The “Reset” button connected to the “Reset” block in Figure 9.31 is used to reset both cart and turning speed error integrators. The “Traj. Start” and Traj. Stop” buttons, connected to the two input blocks of the “Trajectory Design” block, are used to start a driving trajectory and stop it, respectively. The “Trajectory Design” is the same as that in the PID Simulink model shown in Figure 9.24. All seven push buttons have the same functionality as those of the PID strategy.

Figure 9.31: Real-time model-based Simulink three-wheel cart control model with feedforward control.The blue-dashed box in the middle of Figure 9.31 is new for the model-based strategy. Its main function is to generate the feedforward control signals for cart speed and turning. Two lookup tables generate the feedforward control signals (u_{aL0} and u_{aR0}) utilizing the cart and turning speed reference signals (r_{cart} and r_ϕ) as inputs to the two lookup tables, respectively. The cart-speed feedforward signal (from the top lookup table in Figure 9.31’s blue dashed-box) is sent to both motors in PWM counts with the same sign. The sign is changed later (after the two switches) so that these two motors spin in the opposite direction to provide straight driving. Similarly, the cart-turning rate feedforward signal (from the bottom lookup table in Figure 9.31’s blue dashed-box) is sent to both motors in the form of PWM counts. To spin the two motors in the same direction to turn the three-wheel cart, the feedforward signal was fed to both DC motors with different signs. The signals that are input to the two lookup tables are the same as those input to the LQTI controller (to be addressed next). They are generated by adding the reference signal from push bottoms and trajectory outputs with a low-pass filter defined by equation ???.

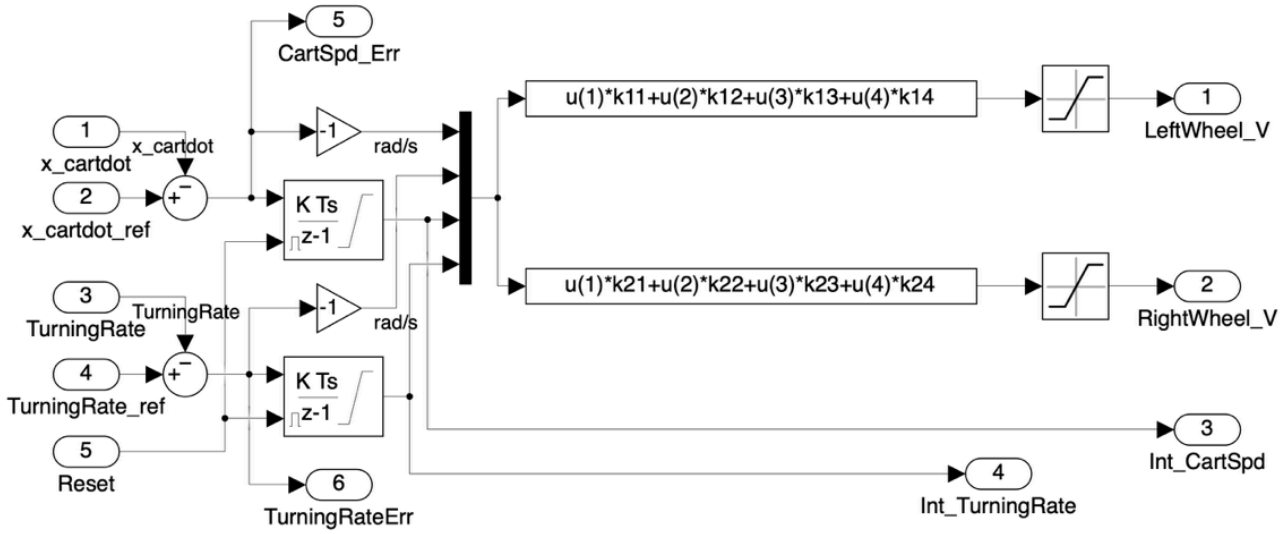


Figure 9.32: Real-time Simulink implementation of LQTI control strategy.

Now, let’s look at LQTI controller implementation in Simulink. Figure 9.32 shows an implementation of the LQTI controller strategy similar to the Simulink block diagram in Figure 9.20. The only difference is in the vector multiplication operations. In Figure 9.20, the LQTI control gain vectors were multiplied using the Simulink vector multiplication blocks. However, the real-time Simulink software (MATLAB 2019b) used for experiments cannot compile it. Instead, user-defined functions help complete the task of multiplying the state vector by the control gain vector. See both user-defined functions in Figure 9.32, where these coefficients ($k_{ij},\ i = 1,\ 2\ \&\ j = 1,\ 2,\ 3,\ 4$) are defined in equation 9.6.3 with K_{cpi} to be the designed LQTI control gain defined in equation 9.6.1.

$$K_{cpi} = \begin{bmatrix} -k_{11} & -k_{12} & -k_{13} & -k_{14} \\ -k_{21} & -k_{22} & -k_{23} & -k_{24} \end{bmatrix} \tag{9.6.3}$$

Once the real-time Simulink control software is developed, the software is debugged and simple experiments (such as straight forward and backward driving, left and right turning) are conducted to validate the software’s functionality. We can then conduct the trajectory driving experiment, similar to what is used in Section 9.5 to validate the dual- and triple-PID strategies.

After the initial experimental validation, we are ready to conduct a scheduled driving test similar to those for dual- and triple-PID strategies. Figure 9.33 shows three-wheel cart speed and turning responses, and associated controls. The horizontal axis range (15 seconds) in Figure 9.33 is the same as that in Figure 9.29. The forward driving started at the 18.8th second with a reference cart speed of 0.15 (m/s) for five seconds. The turning motion started at the 24.2th second with a reference angular turning rate of about 0.45 (rad/s) for two and half seconds..

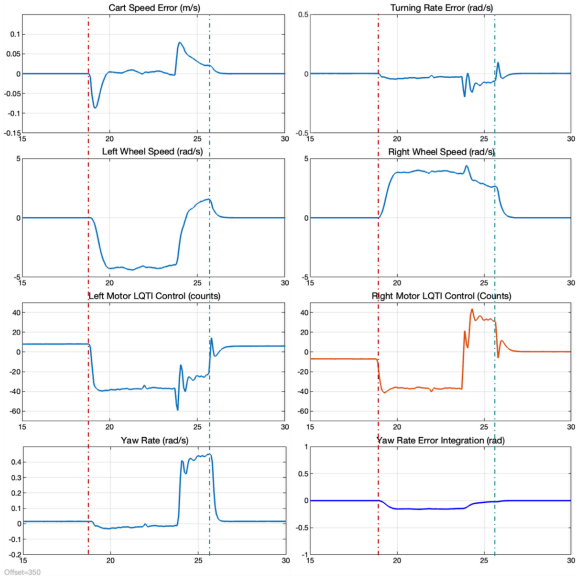


Figure 9.33: Model-based LQTI three-wheel cart test results.

For the first phase (straight forward driving), both left and right wheel-motor control outputs are similar to these for triple-PID case with about 40 PWM counts; see the third row of Figure 9.33. The right motor has a sign reversal, comparing Figure 9.33 and Figure 9.29. However, both the left and right motor control signals from the LQTI controller are much smoother than those from the triple-PID controllers, probably due to low LQTI control gain with feedforward control. As a result, both left and right wheel speed responses are also much smoother than those from the triple-PID controllers. The speed error response indicates that the response time is about 1.1 seconds while the triple-PID needs more than 2 seconds; see the top-left plot in Figure 9.33. Meanwhile, the turning error, shown in the top-right plot of Figure 9.33, remains relatively small.

For the second phase (turning motion) starting at the 24.2th second, the left-wheel speed is reversed while the right-wheel speed remains almost unchanged, with a small peak at the start of turning to reduce tracking error; see the second row of Figure 9.33. Comparing Figure 9.33 with Figure 9.29, we can see that the LQTI control provides a slight advantage over that of the triple-PID one, with a response time less than one second. During the turning between the 24.2th and 26.7th seconds, the track error reduces to zero; see the bottom right of Figure 9.33. Both LQTI and triple-PID control outputs for the right motor are larger than the left motor during the turning, which is mainly used to compensate the difference between left and right motor drive systems. The turning (or yaw) rate error is large at the beginning and end of the turning due to transient operations.

The LQTI control strategy provides an improved response performance over the triple-PID strategy. Both strategies are able to track the desired cart linear and turning reference speeds. However, the LQTI strategy is much simpler than the triple-PID one. Model-based controllers are sensitive to modeling error, and the gains used in simulation studies with good closed-loop performance may not be suitable for experimental study due to modeling error. Although we did not conduct experiments with an unleveled surface, the control strategy is partially validated by demonstrating the capability of tracking the yaw rate. Compared with the dual-PID control strategy, both triple-PID and LQTI control strategies are able to drive the three-wheel cart in a straight line with zero yaw rate reference signal.

Section Summary

This section develops real-time Simulink control software of the model-based LQTI control strategy. After it is implemented into the experimental host laptop, we debug the developed software. Then, we conduct the experimental study utilizing model-based LQTI control of the three-wheel cart. We also reduce the LQTI control gain by modifying the design state weighting matrix. We then collect test data in the Simulink environment through the display scope. Finally, we compare the model-based test responses with those of the triple-PID strategy.

Section Keywords

Practice Problems

1. Why can't the LQTI control model used in simulation studies be used for real-time experimental study? Select the correct answer.
   a. Due to different Simulink versions used in simulation and experiment.
   b. Due to modeling error from the very large Coulomb friction.
   c. Due to the control model difference.
2. What is the purpose of adding the feedforward control to the LQTI control strategy? Select the correct answer.
   a. To improve the system response performance with reduced LQTI gain.
   b. To balance the two motors.
   c. To improve the stability.
3. Why does the control gain multiplication model in the Simulink simulation model (see Figure 9.20) need to be modified for real-time Simulink implementation (see Figure 9.27)? Select all correct answers.
   a. To improve numerical calculation accuracy.
   b. To improve numerical calculation efficiency.
   c. The real-time Simulink model does not allow vector multiplication for the 2019a version of MATLAB.
4. Modify the LQTI simulation model in Figure 9.20. Add the feedforward control with LQTI control gain defined in equation 9.6.1, utilizing the lookup tables used in the real-time software. Conduct the simulation study.

This page titled 9.6: Experimental Investigation of Model-Based Control Strategy is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

9.7: Chapter References

References

1. The Absolute Guide to the Racing Tyres — Part 1: Lateral Force, Accessed: Jun. 9, 2025. [Online] Available: https://racingcardynamics.com/racing...lateral-force/
2. R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 12th ed., Englewood Cliffs, NJ, USA: Prentice Hall, 2011, ISBN-10: 978-0-13-602458-3.
3. G. Zhu, M. W. Pyclik, E. K. Bradley, and L. J. Brackney, "Closed-loop actuator control system having bumpless gain & anti-windup logic," U.S. Patent, US6424906, Jul. 2002.
4. J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," Trans. of the ASME, vol. 64, pp. 759–768, 1942.
5. Desimone Subbaram Naidu, "Optimal Control Systems," 1st ed., Boca Raton, FL, USA: CRC Press, 2003, ISBN-13: 978-1-3152-1442-9.
6. M. Athens and P. L. Flab, *Optimal Control: An Introduction to the Theory and Its Applications,* Garden City, NY, USA: Dover Publications, 2007, ISBN-10: 0-48-645328-6.
7. G. Zhu, M. A. Rote, and R. E. Skelton, "A convergent algorithm for the output covariance constraint control problem," *SIAM J. of Control and Optim.,* vol. 35, no. 1, pp. 341–361, Jan. 1997.
8. G. Zhu, K. M. Grigoriadis, and R. E. Skelton, "Covariance control design for the Hubble Space Telescope," *AIAA J. of Guid., Control, and Dyn.,* vol. 18, no.2, pp. 230–236, May 1995.
9. C. Gamache, M. Van Nieuwstadt, J. Martz, and G. Zhu, "LQTI boost pressure and EGR rate control of a diesel air charge system with eBoost assistance," *Int. J. of Engine Res.,* vol. 25, no. 9, pp. 1659–1670, Sep. 2024, doi: 10.1177/14680874241241364.
10. D. Hu, G. Li, G. Zhu, Z. Liu, and M. Tu, "Linear-quadratic tracking control of a commercial vehicle air brake system," *IEEE Access* [Online], vol. 8, pp. 149741–149750, Aug. 2020, doi: 10.1109/ACCESS.2020.3016353.
11. L. Hua and G. Zhu, "Optimal transient longitudinal trajectory tracking for electrical connected vehicles," *MPDI J. of Machines,* vol. 13, no. 163, pp. 13020263, Feb. 2025, doi: 10.3390/machines13020163.
12. G. Mastinu and M. Plöchl, *Road and Off-Road Vehicle System Dynamics Handbook*, Boca Raton, FL, USA: CRC Press, 2014, ISBN-13: 978-0-4291-2981-0.
13. Y. Huang, H. Wang, A. Khajepour, H. Ding, K. Yuan, and Y. Qin, "A novel local motion planning framework for autonomous vehicles based on resistance network and model predictive control," *IEEE Trans. on Veh. Tech.*, vol. 69, no. 1, pp. 55–66, Jan. 2020.
14. Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. on Veh. Tech.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.
15. H. B. Pacejka and R. S. Sharp, "Shear force development by pneumatic tyres in steady state conditions: a review of modelling aspects," *Veh. Syst. Dyn.,* vol. 20, no. 3–4, pp. 121–175, 1991.
16. H. Dugoff, P. Fancher, and L. Segel, "An analysis of tire traction properties and their influence on vehicle dynamic performance," *SAE Trans.*, vol. 79, pp. 1219–1243, 1970.
17. H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Veh. Syst. Dyn.*, vol. 21, no. sup1, 1992, pp. 1–18.
18. K. Guo, D. Lu, S.-K. Chen, W. C. Lin, and X.-P. Lu, "The UniTire model: A nonlinear and non-steady-state tyre model for vehicle dynamics simulation," *Veh. Syst. Dyn.*, vol. 43 (suppl. 1), pp. 341–358, 2005, doi: 10.1080/00423110500140690.
19. C. C. De Wit, H. Olsson, K. J. Astrom, and P. Lischinsky, "A new model for control of systems with friction," *IEEE Trans. on Autom. Control*, vol. 40, no. 3, pp. 419–425, Mar. 1995.
20. M. Gipser, "FTire — the tire simulation model for all applications related to vehicle dynamics," *Veh. Syst. Dyn.*, vol. 45 (suppl. 1), pp. 139–151, 2007.
21. J. Ahmadi, A. K. Sedigh, and M. Kabganian, "Adaptive vehicle lateral-plane motion control using optimal tire friction forces with saturation limits consideration," *IEEE Trans. on Autom. Control*, vol. 58, no. 8, pp. 4098–4107, Aug. 2009.
22. J. Ahmadi, A. K. Sedigh, and M. Kabganian, "Adaptive vehicle lateral-plane motion control using optimal tire friction forces with saturation limits consideration," *IEEE Trans. on Veh. Tech.*, vol. 58, no. 8, pp. 4098–4107, Aug. 2009.
23. S. Cheng, L. Li, B. Yan, C. Liu, X. Wang, and J. Fang, "Simultaneous estimation of tire side-slip angle and lateral tire force for vehicle lateral stability control," *Mech. Syst. and Signal Process.*, vol. 132, pp. 168–182, Oct. 2019, doi: 10.1016/j.ymssp.2019.06.022.
24. W. Zhao and H. Zhang, "Coupling control strategy of force and displacement for electric differential power steering system of electric vehicle with motorized wheels," *IEEE Trans. on Veh. Tech.*, vol. 67, no. 9, pp. 8118–8128, Sep. 2018.
25. X. Jin, G. Yin, and N. Chen, "Advanced estimation techniques for vehicle system dynamic state: A survey," *MPDI J. of Sensors*, vol. 19, no. 19, p. 4289, 2019.

26. G. Baffet, A. Charara, and D. Lechner, “Estimation of vehicle sideslip, tire force and wheel cornering stiffness,” *Control Eng. Pract.*, vol. 17, no. 11, pp. 1255–1264, Nov. 2009.
27. F. Cheli, E. Sabbioni, M. Pesce, and S. Melzi, “A methodology for vehicle sideslip angle identification: Comparison with experimental data,” *Veh. Syst. Dyn.*, vol. 45, no. 6, pp. 549–563, Jun. 2007.
28. G. Baffet, A. Charara, D. Lechner, and D. Thomas, “Experimental evaluation of observers for tire-road forces, sideslip angle and wheel cornering stiffness,” *Veh. Syst. Dyn.*, vol. 46, no. 6, pp. 501–520, Jun. 2008.
29. X. Zhang and D. Göhlich, “A hierarchical estimator development for estimation of tire-road friction coefficient,” *PLoS One*, vol. 12, no. 2, p. e0171085, Feb. 2017, doi: 10.1371/journal.pone.0171085.
30. N. Xu, Y. Huang, H. Askari, and Z. Tang, “Tire slip angle estimation based on the intelligent tire technology,” *IEEE Trans. on Veh. Tech.,* vol. 70, no. 3, pp. 2239–2249, Mar. 2021, doi: 10.1109/TVT.2021.3059432.
31. K. H. Lundberg and T. W. Barton, “History of inverted-pendulum systems,” *IFAC Proc. Vol.*, vol. 42, no. 24, pp. 131–133, 2010, doi: 10.3182/20091021-3-JP-2009.00025.
32. Emerson, *The laws of centripetal and centrifugal force (1769),* Whitefish, MT, USA: Kessinger Publisher, Sep. 2010, ISBN-13: 978-1-1661-5327-4.
33. T. He, G. Zhu, and S. S-M Swei, “A Sequential Design Approach for Switching H_∞ LPV Control,” *Int. J. of Control, Autom., and Syst.,* vol. 19, no. 10, pp. 3354–3367, Sep. 2021, doi: 10.1007/s12555-020-0363-3.
34. S. Qu, G. Zhu, W. Su, and S. S. Swei, “LPV-based transition flight control design for a tilt-rotor aircraft," *Proc. of IMechE, Part G: J. of Aerosp. Eng.,* vol. 236, no. 16, pp. 3354–3369, Dec. 2022, doi: 10.1177/09544100221083713.
35. C. Gamache, M. Van Nieuwstadt, J. Martz, and G. Zhu, “LQTI boost pressure and EGR rate control of a diesel air charge system with eBoost assistance,” *Int. J. of Engine Res.,* vol. 25, no. 9, pp. 1659–1670, Sep. 2024, doi: 10.1177/14680874241241364.
36. J. K. Astrom and L. Rundqwist, “Integrator windup and how to avoid it,” in *Proc. 1989 Amer. Control Conf.,* Pittsburgh, PA, USA, Jun. 21–23, 1989, pp. 1693–1698.
37. J. G. Ziegler and N. B. Nichols, “Optimum settings for automatic controllers,” *Trans. of ASME*, vol. 64**,** pp. 759–768, 1942.
38. S. Qu, T. He, and G. Zhu, “Gain-scheduled PID control optimized by iterative genetic algorithm of NSGA-II with application to a nonlinear valve system,” *MPDI J. of Appl. Sci. (Special Issue of Res. and Appl. of Intell. Control Alg.),* vol. 13, no. 5, p. 6444, May 2023, doi: 10.3390/app13116444.

Solutions to Practice Problems

1. The number of degrees of freedom is a) 2.
2. The answer is b) False. From the bottom equation of ???, we can see that when the three-wheel cart speed $\dot{x}_{cart}$ is not equal to zero, the term $\dot{x}_{cart}$ affects the turning motion $\dot{\phi}$ if β_3 is not zero (when $\theta_z \neq 0$ and $\theta_y \neq \pi/2$).
3. The answer is b) False. From the bottom equation of ???, we can see that when the three-wheel cart control inputs $u_{aL} = u_{aR}$, the forcing term $u_{aL} - u_{aR}$ is zero. However, terms $u_{aL} + u_{aR}$ and $\dot{x}_{cart}$ are not zero when $\theta_z \neq 0$ and $\theta_y \neq \pi/2$. As a result, $\dot{\phi}$ may not be damped out or goes to zero.
4. The answer is b) False. From the top and bottom equations of ???, we can see that the three-wheel cart linear motion $\dot{x}_{cart}$ in x_{cart} direction and its angular motion $\dot{\phi}$ are highly coupled when $\theta_z \neq 0$ and $\theta_y \neq \pi/2$ (slip-angle not equal to zero).
5. The main disadvantage of the speed control strategy used for the mini-Segway is its inability to compensate for the difference between left and right motors and associated drive electronics. That is, even though the same armature reference signals are applied for both left and right motors, due to the mentioned difference, the mini-Segway (or three-wheel cart) may not drive in a straight line.
6. The answer is c). For Method 1, the cart speed is controlled in closed loop by two PID controllers and the turning speed in open loop.

1. The answer is b). For Method 2, both the cart forward/backward and turning speeds are controlled in closed loop by three PID controllers.
2. The answer is a) $\theta_z = 0.2073\ (rad)$ and $\theta_y = 0.5182\ (rad)$. Both angles can be calculated using equations ??? and ???.
3. The answer is a); the order is 4.
4. False. When the driving ground is not level, the gravity force will cause wheel lateral slip that is proportional to the linear speed $\dot{x}_{cart}$. Thus, the turning speed $\dot{\phi}$ is coupled with the three-wheel cart speed.
5. These two additional error integral states x_{ce} and ϕ_{ce}, created for tracking purposes, are not controllable.
6. The poles associated with the integration states are at the origin of the complex plan and not stable. Therefore, both poles need to be moved to $-\varepsilon$ with $\varepsilon > 0$ so that a positive definite solution of the associated Riccati equation exists.

1. The answer is b).
2. The answer is c).
3. The answer is b).
4. The answer is c).

5. The answer is b). The other two choices have wrong corner frequency.
6. The answer is b). The other two choices have either wrong corner frequency or wrong application purpose.

1. The answers are a), b), and c). That is, all three could lead to a non-zero turning rate.
2. The answer is c), reducing both magnitude and mean of the three-wheel cart turning rate.
3. The answer is b), due to modeling error.
4. The answer is a), to improve system response performance.
5. The answer is b). The real-time Simulink model does not allow vector multiplication for the 2019a version.
6. The LQTI feedforward simulation model can be generated based on the Simulink model shown in Figure 9.20. Add the feedforward control based on the architecture drawing in Figure 9.12 and the Simulink model inside the blue box in Figure 9.31.

This page titled 9.7: Chapter References is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

10: Appendix A - MATLAB and Simulink Software

MATLAB, short for “MATrix LABoratory,” is both a powerful and versatile programming language and a numeric computing/simulating environment developed by MathWorks. It is widely used by engineers and research scientists for various industrial applications and academic research. These include conducting basic mathematical calculations and matrix operations, solving ordinary and partial differential equations, simulating complex physical systems, and analyzing data sets for statical purpose and visualization. The following list of MATLAB applications are widely used in mechatronic systems with associated key features and functionalities.

Numerical Calculation: MATLAB is capable of handling complex numerical computations, especially for those involving matrices and vectors. For example, with any given matrix, MATLAB can calculate its eigenvalues and eigenvectors, convert it to its Jordan form for square matrices, conduct its singular value decomposition, and calculate associated singular values and vectors. In addition, MATLAB offers a rich library of mathematical functions for linear algebra, statistics, Fourier frequency analysis (including fast Fourier transformation), and more.

Data Visualization: MATLAB provides a rich family of graphic tools for creating high-quality two-dimensional and three-dimensional plots for visualization purpose. Excel-type charts can also be created based on data provided. In addition, graphic files can be edited and linked into an animation for visualization of data to gain insights. For example, the simulated robot motion can be viewed through animation to better understand its motion trajectory. In fact, many plots in this book are created by MATLAB.

High-Level Programming Language: MATLAB utilizes high level program language with its syntax designed to express matrix and array mathematics directly, making it easier for engineers and research scientists to work with. For example, matrix eigenvalue can be calculated using a simple function call. Matrix multiplication can be expressed as a simple product operation in MATLAB without utilizing individual elemental multiplications and summations.

Interactive Development Environment: The MATLAB environment offers tools and facilities for managing variables in MATLAB workspace, importing and exporting data, developing and line-by-line debugging software, and profiling code. It includes command window, editor, and workspace windows, along with command history for a user-friendly experience. We can type a MATLAB command to get it executed in the command window or run a “*.m” file to execute a MATLAB program.

Interoperability: MATLAB can interact with other programming languages like C, C^{++}, Java, and Python, allowing users to integrate existing code in other programming languages to leverage its diverse functionalities. This feature is very important because the existing software in other programming languages can be integrated with the MATLAB program. Thus, the numerical simulation and calculation can be conducted without converting the existing software in other programming languages into the MATLAB program, leading to significant time and cost savings.

Simulink: This add-on product of MATLAB is designed to enable model-based design and simulation of dynamic systems using a graphical (block-diagram) environment. It allows users to create block diagrams representing dynamic systems, simulate their behaviors, and analyze the simulated results. Simulink is closely integrated with MATLAB, enabling users to incorporate MATLAB algorithms into Simulink models based on “S” function blocks. Users can then export simulation results to MATLAB for generating graphics and further analysis.

Real-Time Workshop: When Simulink is coupled with this MATLAB product, it can automatically generate “C” source code for real-time implementation of control algorithms in mechatronic systems. As the efficiency and flexibility of the auto-code generation improves, the real-time workshop is becoming more widely adopted for developing production control systems. In addition, the real-time workshop is now a tool for embedded system design work because of its flexibility and capacity for fast design iteration.

Simulink Real-Time (formerly known as xPC Target): An environment can be used for simulating and testing Simulink and Stateflow (another MATLAB product) models in real-time on both host computers and physical systems. Note that Stateflow is very important for a system with multiple operational states. Both the mini-Segway and the three-wheel cart utilize this feature. Real-time control software packages for mini-Segways and three-wheel carts are developed based upon Simulink Real-Time utilizing the MATLAB and Simulink Arduino supporting package.

MATLAB Toolboxes: MATLAB has a vast array of specialized toolboxes to extend the core functionalities for specific applications

such signal processing, control systems, image processing, machine learning, nonlinear system modeling, optimization, and more. Many toolboxes associated with mechatronic system applications are listed below.

1. **Control system toolbox:** MATLAB's control system toolbox provides algorithms and function calls for systematically analyzing, designing, and tuning linear time-invariant (LTI) control systems; see Chapter 3 for related topics. The system model, used for control design and simulation, can be in a form of a transfer function, state-space, zero-pole-gain, or frequency-response model. These function calls, such as step response plot and Bode plot, make it easy to analyze and visualize system behavior in both time and frequency domains. Controllers can be tuned by utilizing interactive tools such as Bode loop shaping and the root locus method. The toolbox automatically tunes both SISO (single-input and single output) and MIMO (multi-input and multi-output) compensators, including PID controllers. Gain-scheduling controllers can also be tuned with multiple tuning performance objectives, such as reference tracking, disturbance rejection, and stability margins; here the tuned controller can be verified by rise time, overshoot, settling time, gain and phase margins, and other requirements.
2. **Robust control toolbox:** This toolbox provides function calls and blocks. They are used to analyze and tune LTI (linear time invariant) control systems for performance and robustness in the presence of plant uncertainty, where uncertain models can be created by combining nominal dynamics with uncertain elements, such as uncertain parameters or unmodeled dynamics. The impact of plant model uncertainty on control system performance can be determined by identifying the worst-case combinations of uncertain elements. $\mathcal{H}_2$, $\mathcal{H}_\infty$, and μ-synthesis techniques can be used for robust controller design to optimize closed-loop system robust stability and performance. The robust tuning is also available for automated tuning the controller performance.
3. **DSP system toolbox:** The DSP (digital signal processing) system toolbox provides algorithms, function calls, and scopes for designing, simulating, and analyzing signal processing features in both MATLAB and Simulink. Here, real-time DSP functions can be designed for mechatronic systems communications, radar, audio, medical devices, IoT (internet of things), and other applications. With the help of the DSP system toolbox, many kinds of filters can be designed and analyzed. These include FIR (finite impulse response), IIR (infinity impulse response), multirate, multistage, and adaptive filters. The "Time Scope," "Spectrum Analyzer," and "Logic Analyzer" in the DSP system toolbox dynamically visualize and measure streaming signals.
4. **MPC toolbox:** The MPC (model predictive control) toolbox provides function calls, Simulink blocks, and reference examples to develop model predictive controllers for both linear and nonlinear systems. For linear systems, this toolbox designs implicit, explicit, adaptive, and gain-scheduled model-predictive controllers. For nonlinear systems, single- and multi-stage nonlinear MPC controllers can be designed and realized. Closed-loop systems performances can be evaluated in both MATLAB and Simulink by utilizing closed-loop simulations.
5. **Simscape toolbox:** Simscape toolbox makes it possible to rapidly create models of physical systems within the Simulink environment, especially for highly nonlinear systems such as electromagnetic systems and fluid dynamical systems. With the help of Simscape, these physical component models can be built based upon physical connections that directly integrate with block diagrams and other modeling paradigms. The system models (such as electric motors, H-bridge rectifiers, hydraulic actuators, and refrigeration systems) can be obtained by assembling these basic components into a schematic Simulink model. Simscape also helps develop closed-loop control system models to test closed-loop system performance. Custom component models can also be created in MATLAB using Simscape language, enabling text-based modeling components and libraries. Simscape models can also be deployed to other simulation environments, including hardware-in-the-loop (HIL) systems for real-time simulations using "C" code generation.
6. **Optimization toolbox:** The MATLAB optimization toolbox provides functions for finding optimal parameters that minimize or maximize objectives (cost functions) while satisfying given constraints. This toolbox includes solvers for LP (linear programming), MILP (mixed-integer linear programming), QP (quadratic programming), SOCP (second-order cone programming), NLP (nonlinear programming), CLLS (constrained linear least squares), NLS (nonlinear Least Squares), and nonlinear equations. The optimization problem can be defined by functions and matrices or by specifying variable expressions. This toolbox also provides numerical solvers used to find optimal solutions to continuous and discrete problems, perform tradeoff analyses, and incorporate optimization methods into algorithms and applications. Furthermore, the toolbox can be used to perform design optimization tasks such as parameter estimation, component selection, and parameter tuning.
7. **MATLAB tools for solving LMI problems:** Many control problems and design specifications can be put into LMI (linear matrix inequality) formulations. This is especially true for Lyapunov-based analysis and design, optimal LQG (linear quadratic Gaussian) control, $\mathcal{H}_\infty$ control, covariance control, gain-scheduling control for LPV (linear parameter-varying) systems, etc. Recently, LMI applications are extended to parameter and signal estimations, system identification, design optimization, structural design, matrix scaling problems, and so on. The main strength of LMI formulations is the ability to combine various design constraints or objectives in a numerically tractable manner. The basic LMI solver in MATLAB is part of the Robust Control toolbox, called LMI Lab, and other LMI tools includes free toolboxes such as YALIMP, SeDuMi, and LMITOOL.

Thumbnail: First eigenfunction of the L-shaped membrane, resembling (but not identical) to MATLAB's logo trademarked by MathWorks Inc. (Public Domain; Jarekt, LukeTriton via Wikipedia)

This page titled 10: Appendix A - MATLAB and Simulink Software is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11: APPENDIX B - MECHATRONICS LAB SESSIONS

This appendix introduces nine lab sessions for the mini-Segway mechatronic system; see Chapter 8 for detailed descriptions. These sessions can also be modified for individual teaching goals. We do not assume that students have any experience with MATLAB and Simulink. (For those who do, Lab B.0 can be eliminated.)

11.0: MATLAB/Simulink Installation

11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink

11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)

11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi

11.4: DC-Motor Control Utilizing Arduino Hardware

11.5: Simulink GUI (Graphic User Interface) Design and Programming

11.6: Sensor Fusion vs. Single Sensor

11.7: PID Tuning for Mini-Segway Simulink Model

11.8: PID Tuning for Mini-Segway Hardware System

11.9: Mini-Segway Stabilization and Drive Project

This page titled 11: Appendix B - Mechatronics Lab Sessions is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11.0: MATLAB/Simulink Installation

Lab B.0 Objectives

After completing this lab section, you will be able to:

1. Program and run MATLAB software in the MATLAB environment.
2. Program and run Simulink graphic-based software in the MATLAB/Simulink environment.

MATLAB Installation

Install MATLAB **R2019a** (or **2024b**) on your own laptop and update your academic license. Get the campus software by the link below.

https://www.mathworks.com/academia/t...igibility.html

During the installation, you will need to select the following MathWorks toolboxes. They will be used to program the mini-Segway projects or other projects.

1. MATLAB and Simulink
2. Real-Time Workshop™
3. Control System Toolbox™
4. Curve Fitting Toolbox™
5. DSP System Toolbox™
6. Image Processing Toolbox™
7. Instrument Control Toolbox™
8. Optimization Toolbox™
9. Signal Processing Toolbox™
10. Simscape™
11. Simscape Multibody™
12. Stateflow
13. Symbolic Math Toolbox™

Most US college students have a free MATLAB license. Any issues associated with your license can be addressed by your IT department.

If you already have MATLAB **R2019a** (or **2024b**) installed, check if you have the required products by typing the command ver and press “Enter” (or “return”) in the MATLAB command window.

Simulation with MATLAB

Before conducting a simulation study, we need to introduce MATLAB fundamentals. There are two ways to run a program in MATLAB. You can choose either of them for your future lab operations. The final results will be the same. The two programing methods are shown from Steps 1 to 5 below.

1. Start MATLAB 2019a (or 2024b) from either the Windows Start Menu or the desktop. A MATLAB command window will pop up (as shown in Figure B.1). The basic configuration of the command window should include “Tool bar,” “Current Folder Browser,” “Command Window,” and “Workspace.”

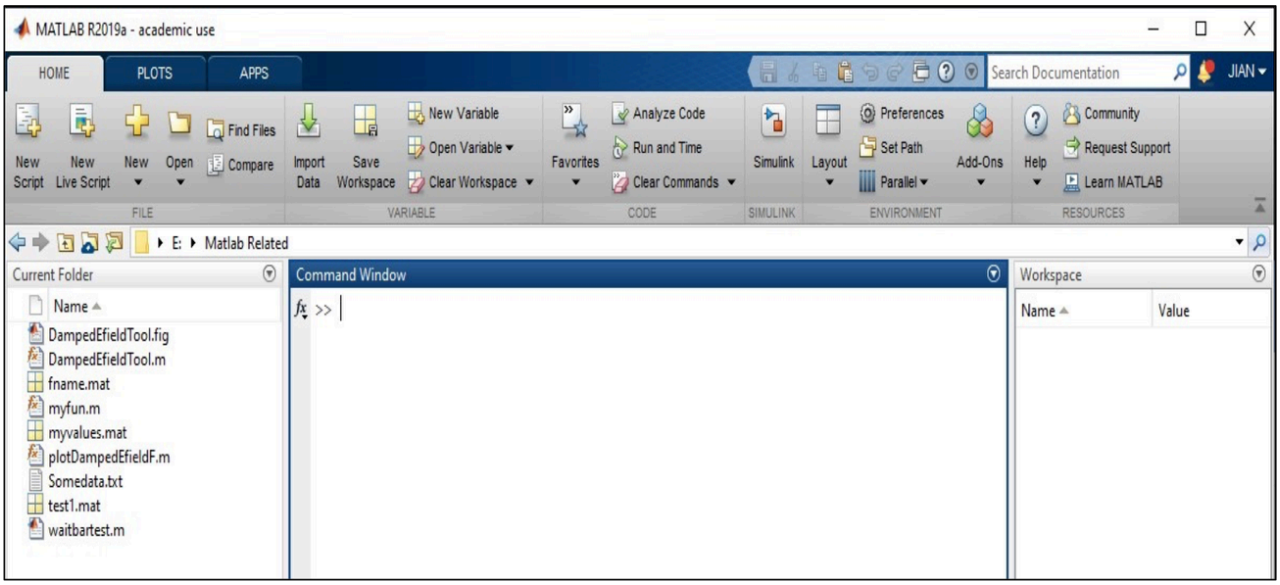


Figure B.1: MATLAB command window.

2. In the command window, type the first command in this lab section. Type the command display ('Hello, Spartan!') at the ">>" prompt and press "Enter" (or "return") on the keyboard. The result is shown in Figure B.2.

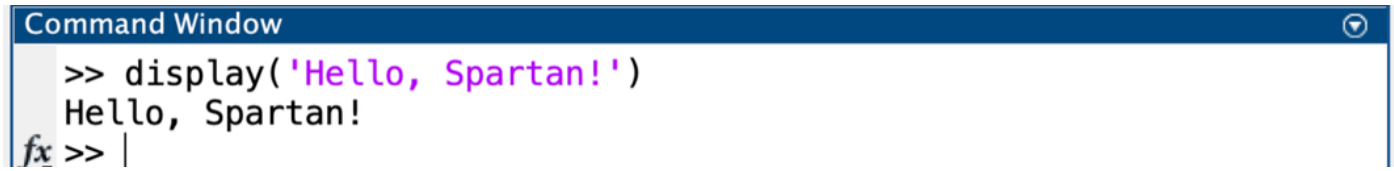


Figure B.2: First command 'Hello, Spartan.'

3. Script is also used widely in MATLAB. Left single click on the "New" icon and select "script." Then a new "Editor" window will pop up (see Figure B.3). Type the disp command in the file named "Test_1.m". Note that commands disp and display are equivalent.

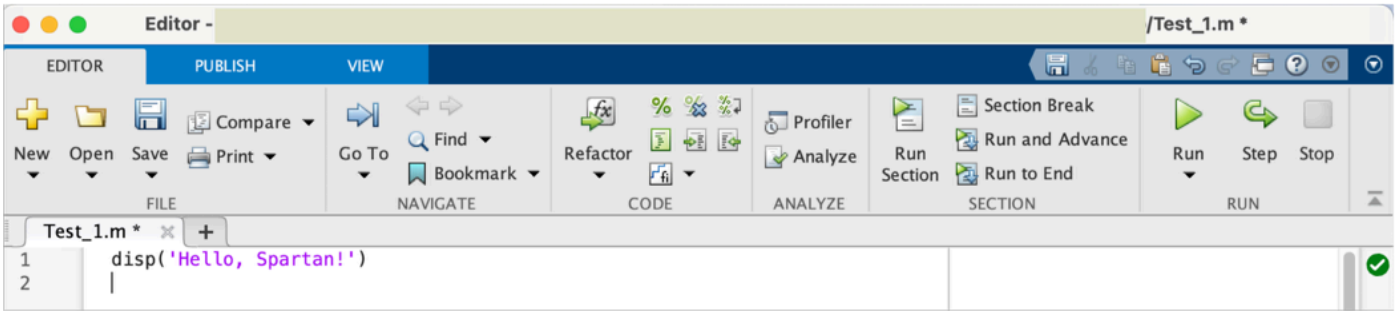


Figure B.3: Script editor window.

4. Single click the "Run" button in the tool menu. The "Save" window will pop up. Then type "Test_1" into the File name and click "Save." The file name and results will appear in the command window (see Figure B.4 below) accordingly.

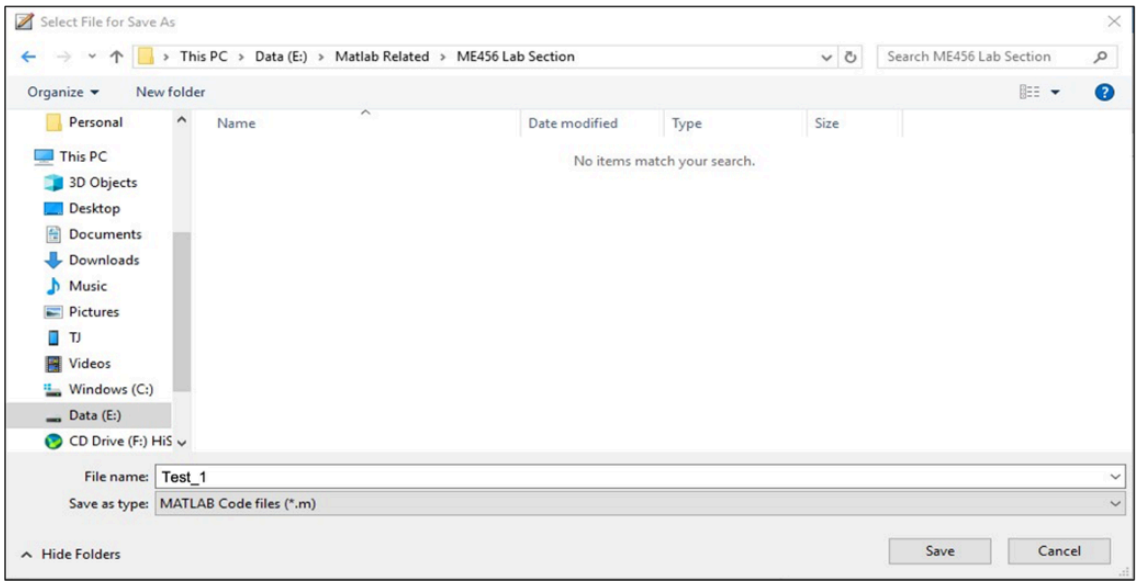


Figure B.4: File save window.

Type Test_1 in the MATLAB command window, and Hello Spartan! will be displayed, as shown in Figure B.5.

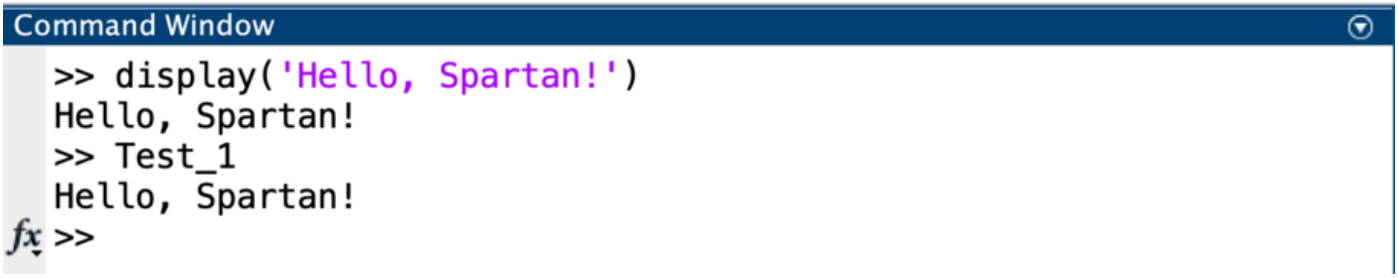


Figure B.5: Results in the Command window.

5. A variable can be defined with an assigned value using the expression, <**variable name**> = <**expression**>**;**, where the semicolon is used if you do *not* want to show the result in the MATLAB command window. Note that both a string and numerical value can be assigned to a variable. Now, try to delete the semicolon in your second command. Your assignment result should appear in the MATLAB command window. Try the command in Figure B.6. to become familiar with the variable assignment.

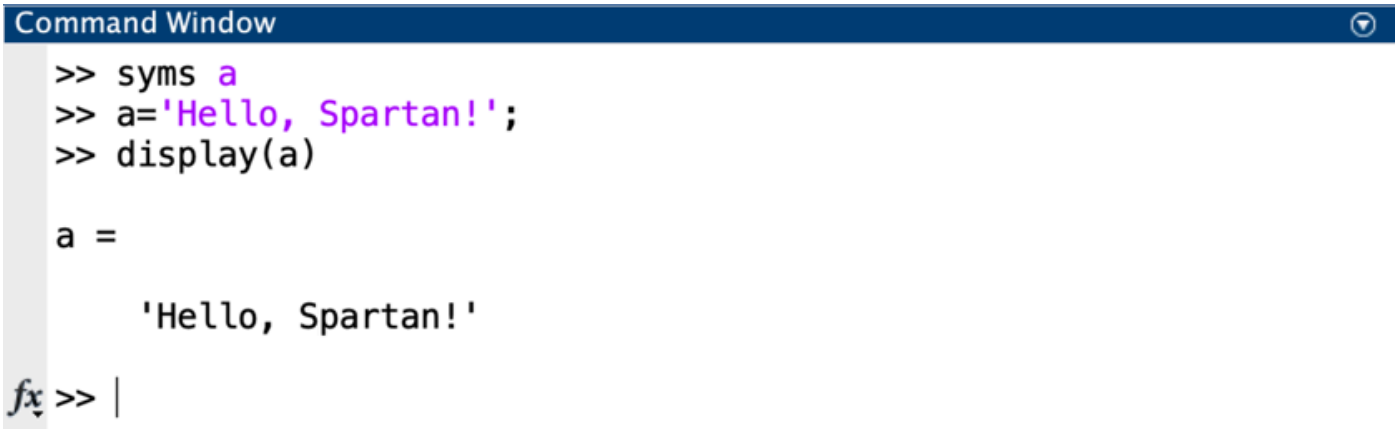


Figure B.6: Revisit “Hello Spartan!” with variable assignment.

This page titled 11.0: MATLAB/Simulink Installation is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink

Lab B.1 Objectives

After completing this lab session, you will be able to:

1. Simulate the 1st and 2nd order systems in MATLAB/Simulink.
2. Tune the PID controller gains in MATLAB/Simulink.

Simulation of the 1st and 2nd Order Systems Using Matlab

We can use MATLAB to simulate a dynamic system directly utilizing MATLAB commands. This section describes the step-by-step process of simulating a given transfer function in MATLAB including the first and second order systems.

Responses of the 1st and 2nd order LTI systems. We can calculate the system step time response using the MATLAB commands in Figure B.7 in the form of a MATLAB executable "*.m" file.

```
% step response for 1st order system
k_dc = 5;
Tc = 10;
s = tf('s');
G = k_dc/(Tc*s+1) % 1st order system
step(G)
stepinfo(G)

k_dc = 1;
w_n = 10;
zeta = 0.2;

s = tf('s');
G1 = k_dc*w_n^2/(s^2 + 2*zeta*w_n*s + w_n^2);% 2nd order system
step(G1)
axis([0 3 0 2])
stepinfo(G1)
```

Figure B.7: Code for calculating the system step response.

MATLAB response analyzer. MATLAB provides a powerful graphical user interface (GUI) for analyzing LTI (linear time-invariant) systems that can be accessed using the syntax linearSystemAnalyzer('step',G). In the shown syntax, G is the given system plant model, and step specifies the excitation type. If you right-click on the step response graph and select **Characteristics**, you can choose to have several system measures overlaid on the response, including peak response, settling time, rise time, and steady-state. See Chapter 3 for their definitions.

Exercise B1.1

Refer to the following second order systems in its standard form, introduced in Section 3.3.

$$G(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{11.1.1}$$

where K is the system DC gain, ω_n is the undamped natural frequency, and ζ is the damping ratio. Simulate the second order system for $\omega_n = 15$ and $\zeta = 0,\ 0.2,\ 1.0,$ and 1.2 separately. Then complete the following step list.

1. State the category of the second order system in terms of its damping ratio undamped, underdamped, critically damped, or overdamped).
2. Use the command linearSystemAnalyzer('step',G) in the MATLAB command window. Obtain the step response diagram for each simulated system.
3. Then complete the chart below for the settling time, rise time, and overshoot for $\zeta = 0.2$. Show both code and results.

Damping ratio	Setting Time	Rise Time	Overshoot
0.2			

Hint: An example of code is shown below for $\omega_n = 10$ and $\zeta = 0.2$.

```
% Step response for 2nd order system
w_n = 10;
zeta = 0.2; %underdamped system

s = tf('s');
G1 = w_n^2/(s^2 + 2*zeta*w_n*s + w_n^2);
linearSystemAnalyzer('step',G1)
axis([0 3 0 2])
```

Figure B.8: Sample code for Exercise B2.1

Simulation with Simulink

To start Simulink, single click on the icon in the command window's tool menu. You may also bring up this window by typing simulink in the command window. The Simulink start page will pop up (see Figure B.9). You may choose a Blank Model for your first trial, using the following steps.

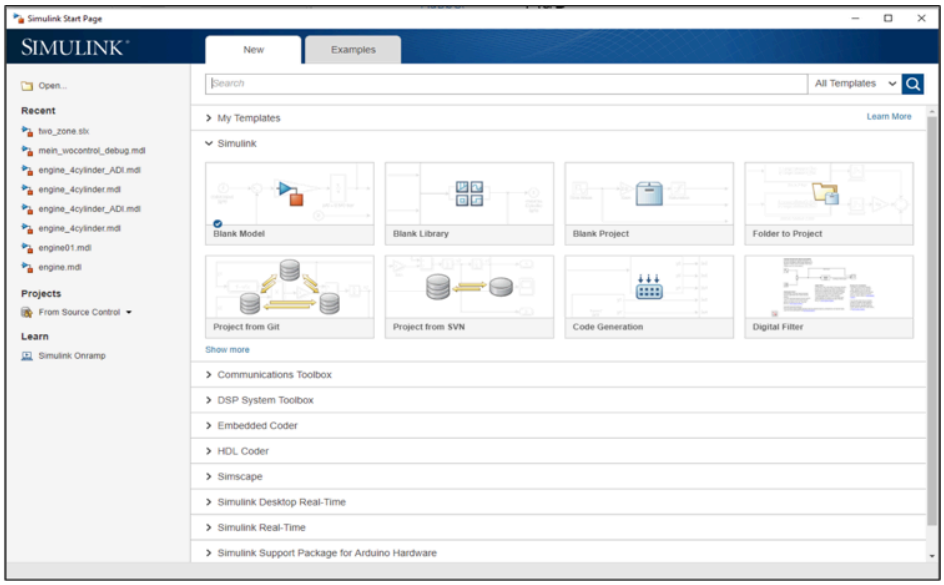
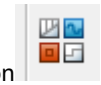


Figure B.9: Simulink start page.

1. Single click on the icon  to show the Simulink library browser (see Figure B.10). You will see a collection of different block sets used to build a Simulink model.

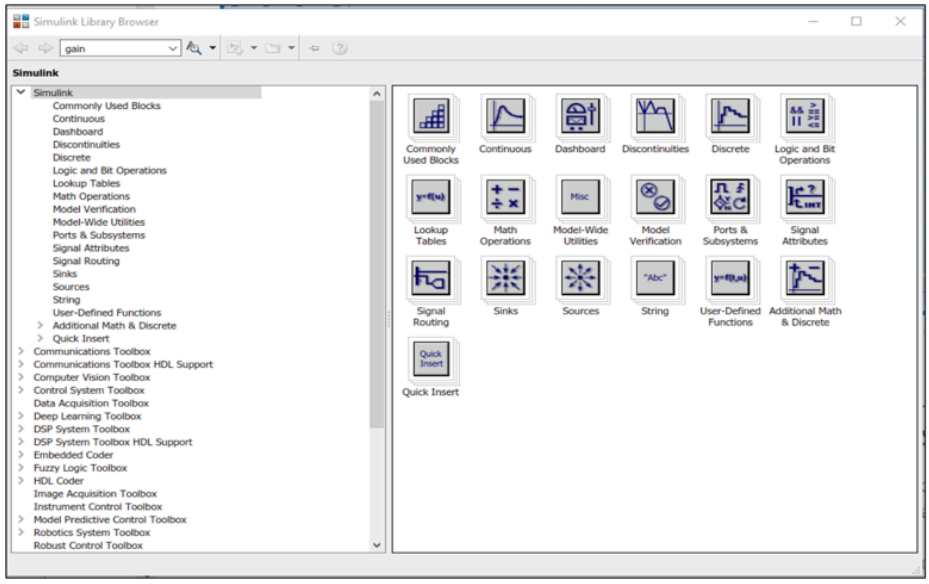


Figure B.10: Simulink library.

1. Build the first model of a second order system as shown in Figure B.11 by selecting **Step** from “Source,” **Scope** from “Sink,” and **Transfer Fcn** from “Continuous” in the library browser.

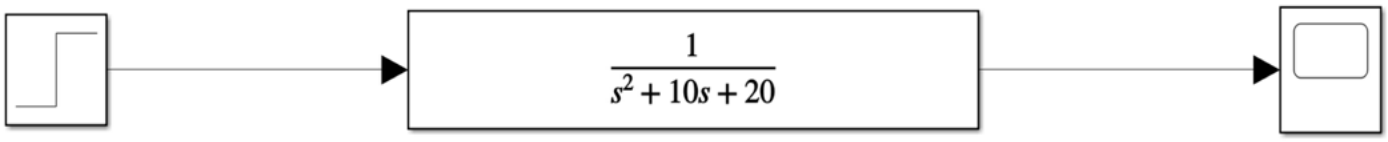


Figure B.11: The first Simulink model.

1. Double-click on the block **Transfer Fcn** to open the “Block Parameters” window. Configure the transfer function parameters shown in Figure B.12.

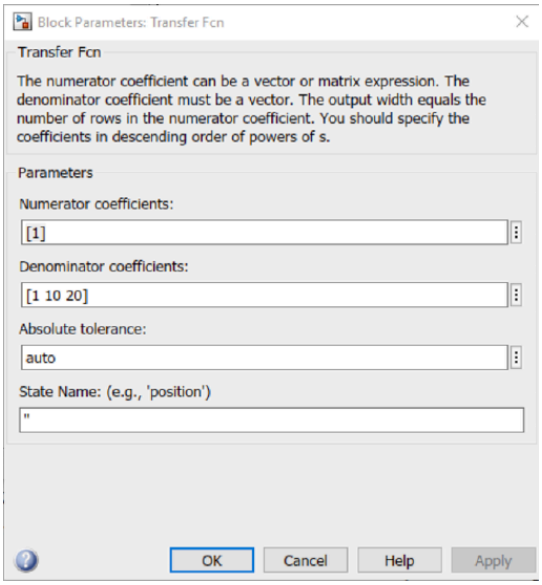


Figure B.12: Transfer function parameter configuration for the first Simulink model.

1. Connect these three blocks with the corresponding lines by dragging a line from its input port and dropping it (release the mouse button) to complete the Simulink model shown in Figure B.11.
2. Single click the button  to run the Simulink model. After the simulation completes, double-click the **Scope** block, and the chart in Figure B.13 will open.

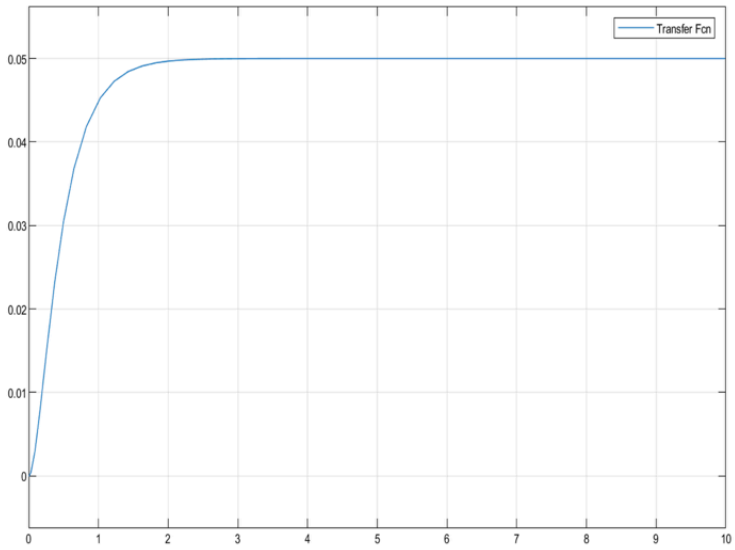


Figure B.13: Simulation result of the first second-order model.

Exercise B1.2

Use $\omega_n = 15$ and $\zeta = 0.2$ for the second order system defined in equation (B.1). Simulate the system step response. Show the code, model and results.

PID Controller Tuning

A PID (proportional-integral-derivative) controller can be tuned by either a MATLAB command or a Simulink block. The PID controller can be defined by its time-domain differential equation and Laplace transform in equations 11.1.2 and 11.1.3.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{11.1.2}$$

$$C(s) = K_p + \frac{K_i}{s} + K_d s \tag{11.1.3}$$

where $e(t)$ is the error input; $u(t)$ is the PID control output; and K_p, K_i, and K_d are proportional, integral, and derivative gains of the PID controller, respectively. In addition, $C(s)$ is the PID controller's transfer function.

MATLAB PID tuning. MATLAB provides a tool for automatically optimizing (tuning) PID control gains to reduce the trial-and-error effort. You can access the tuning algorithm through a nice graphical user interface (GUI) using the command `pidTuner`. Let's explore this automated tuning tool to generate a proportional controller by entering the command `pidTuner(P,'PID')` in the Command window. In the shown syntax, `P` is the given system plant model, and `PID` specifies that the tuner employ a proportional-integral-derivative controller. The plant model can be generated using the following commands in the MATLAB command window or using the following code file (see Figure B.14).

```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
pidTuner(P,'p')
```

Figure B.14: An example of forming a plant model and launching the PID turning GUI.

We can now interactively tune the controller parameters on the tuning GUI; see the resulting response in the GUI window shown in Figure B.15. Single click on the icon Show Parameters 

in the tool menu. The screen should be the same as Figure B.15.

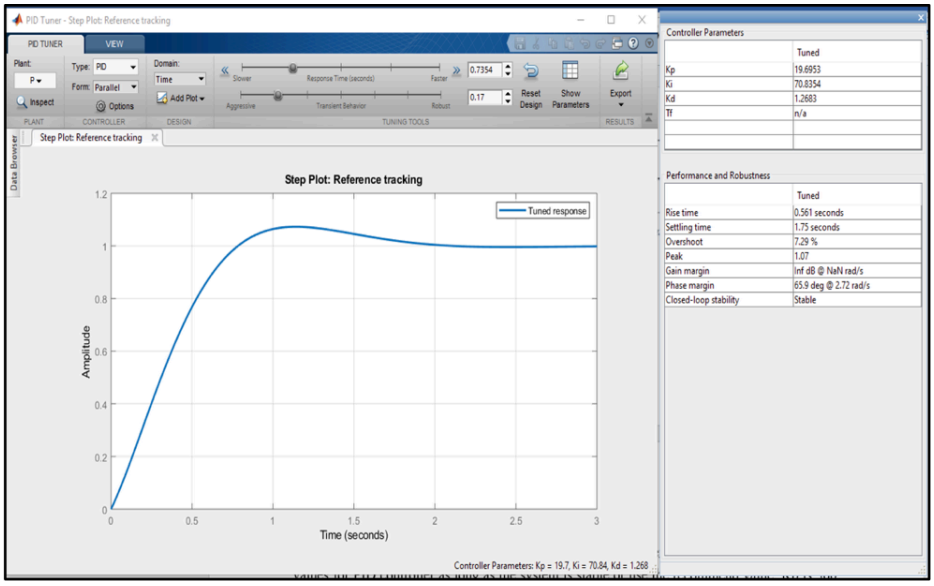


Figure B.15: An example of using MATLAB pidTuner.

Simulink PID Tuning. Based upon the Simulink model shown in Figure B.11, add the block **PID Controller** from the "Continuous" group and the block **ADD** from the "Math Operations" group in the Simulink Library browser. These will help you obtain the Simulink model shown in Figure B.16 to form a feedback control system.

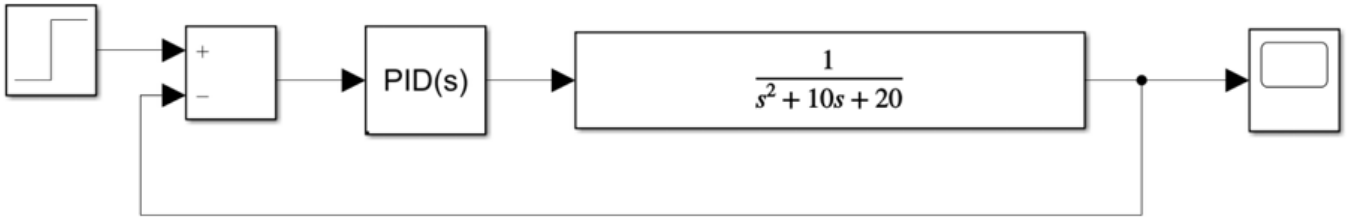


Figure B.16: Simulink model for PID control tuning.

Double-click on the **ADD** block and change its list of signs as shown in Figure B.17 to have correct parameters.

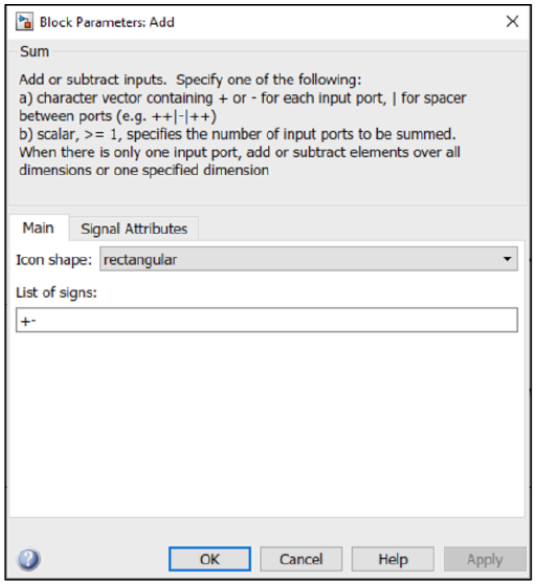


Figure B.17: Simulink "ADD" block parameter window.

Exercise B1.3

Use the pidTuner application to achieve PI control-gain tuning for the system defined in Figure B.17. Find K_P and K_i values for the closed-loop system with closed-loop system rise time close to 0.1 second. Then tune the PID control gains in the Simulink model for the system defined in Figure B.18. Using the five-step tuning process introduced in Section 3.4 to show the responses when you tuned K_p, K_i, and K_d in Steps 2 through 4 of the tuning process and the corresponding PID control parameters. Show the code/model and results.

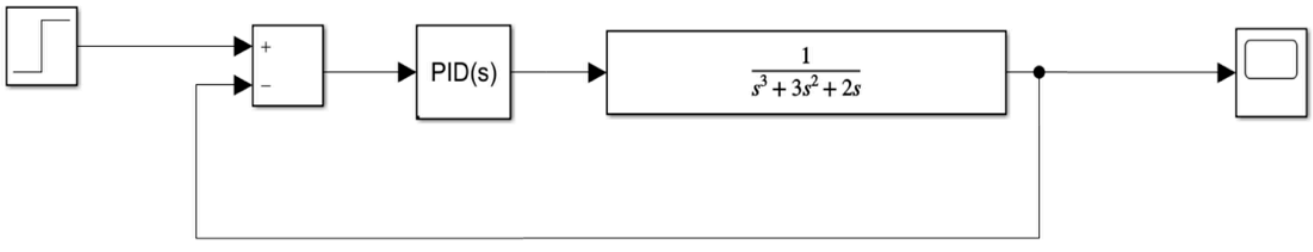


Figure B.18: Simulink model for Exercise B1.3.

This page titled 11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)

Lab B.2 Objectives

After completing this lab section, you will be able to:

1. Understand the analog system and the associated digital system through sample and hold discretization.
2. Understand the principle of the FFT (fast Fourier transform).

Zero-Order Hold

In Chapter 4, the graph in Figure B.19 was introduced to provide a basic concept of D/A (digital-to-analog) and A/D (analog-to-digital) conversions. In this lab section, the Simulink is used to simulate the A/D conversion process.

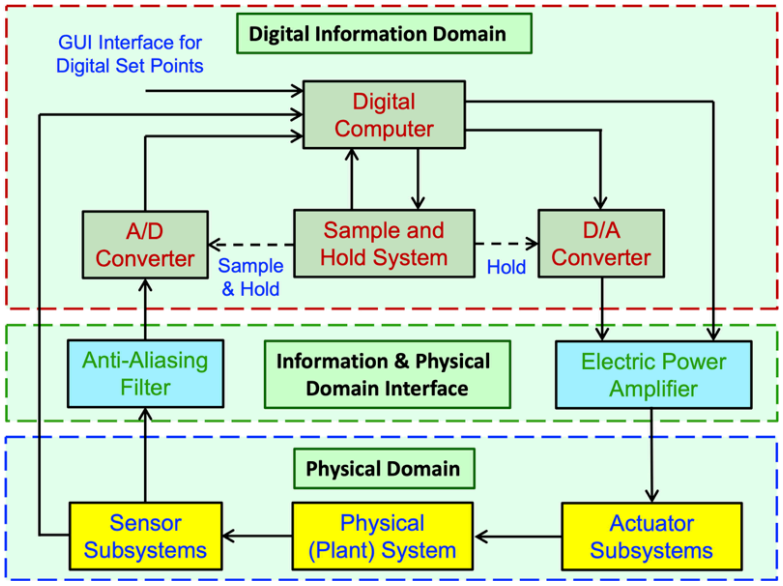


Figure B.19: Mechatronic system with A/D and D/A converters.

The A/D data sampling with the zero-order hold process can be realized in the Simulink with block **Zero order hold** and followed with the **Quantizer** block for quantization. Follow the two steps described below to generate a Simulink model, shown in Figure B.20; these results are required for your report.

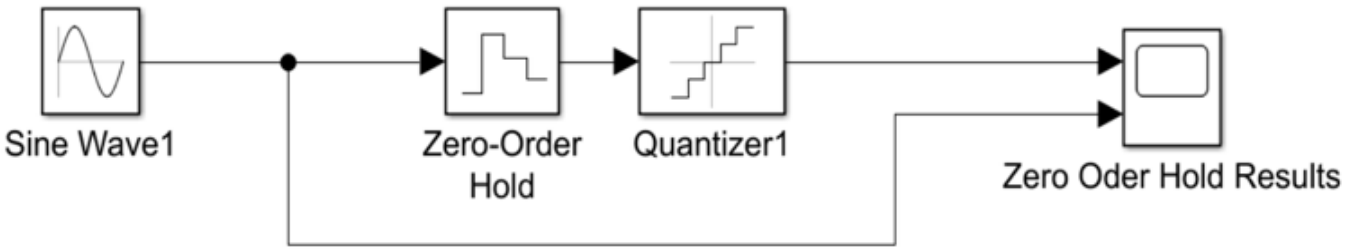


Figure B.20: Zero-order sample and holder with quantization.

1. Choose the blocks in Figure B.20 from the Simulink library browser, build the model indicated in Figure B.20. Set the zero-order hold sample period to 0.4 second.
2. Edit the parameters in the **Sine Wave** block as below to generate a sinusoidal signal, $2\sin(40\pi t)$; see Figure B.21. Record the frequency of this signal in your short form (lab report).

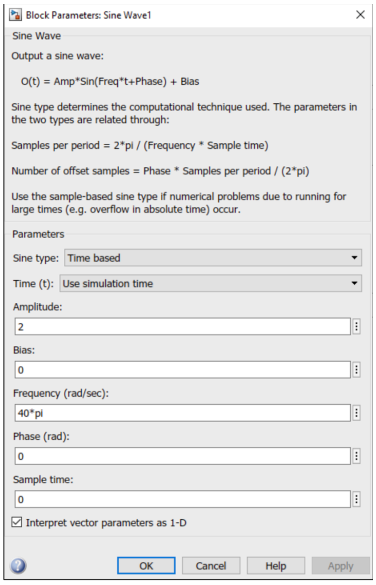


Figure B.21: Calibration window of the “Sine Wave” block.

Note that Lab B.2 has three main tasks to be completed, shown below, to be completed, utilizing the model developed in Figure B.20.

Task #1:

1. Edit the parameters in the **Quantizer** block, as shown in Figure B.22. Here, 4 is the magnitude of the input signal and an 8-bit quantizer is used to represent 2^8 steps with a maximum digital number of 2^8-1.
2. Edit the parameters in the **Zero-Order Hold** block and set sample time to 0.04 second. Write down the sample frequency (in unit Hz) in the short form and show the simulation results.
3. Edit the parameters in the **Zero-Order Hold** block and set sample time to 0.033 second. Write down the sample frequency (in unit Hz) in the short form and show the simulation results.
4. Compare simulation results from Steps 2 and 3. Then describe how the sample period influenced sampled digital signals.

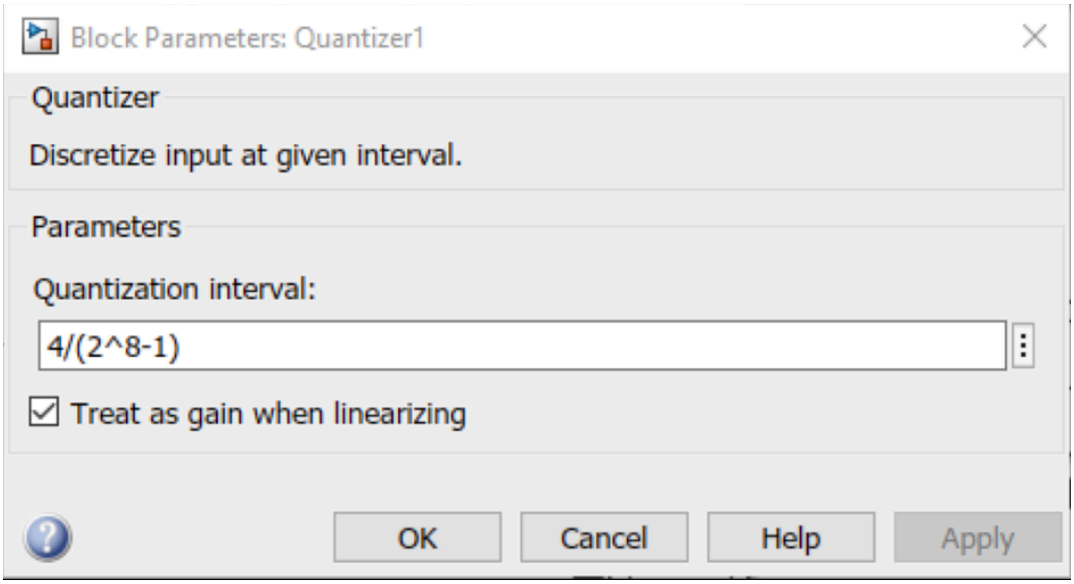


Figure B.22: Calibration window of "Quantizer" block.

Task #2:

1. Keep the configuration for the quantizer unchanged.
2. Edit parameters in the **Zero-Order Hold** block and set sample time to 0.025 second. Write down the sample frequency (in unit Hz) in the short form and show the simulation results.
3. Edit parameters in the **Zero-Order Hold** block with sample time of 0.025 second. Add $\pi/6\ (rad)$ phase delay to the original sinusoidal wave and keep its frequency unchanged at 20 Hz. Show the final simulation results. Can the frequency and amplitude of the signal be recovered based on Nyquist-Shannon theory?
4. Edit parameters in the **Zero-Order Hold** block and set sample time to 0.005 second. Write down the sample frequency (in unit Hz) in the short form and show the simulation results.
5. Edit parameters in the **Zero-Order Hold** block and set sample time to 0.0025 second. Write down the sample frequency (in unit Hz) in the short form and show the simulation results
6. Compare simulation results from Steps 2, 4, and 5. Describe how sample time influences sampled results.

Task #3:

1. Set the sample time to 0.001 second and keep it for the following two steps.
2. Set the quantization bit to 4 in the **Quantizer** block and show the simulation results.
3. Set the quantization bit to 16 in the **Quantizer** block and show the simulation results.
4. Compare simulation results from Steps 2 and 3. Describe how the quantizer number of bits influences sampled results.

FFT of Digital Signals

FFT Code for Single Sided Spectrum. Use Fourier transforms to find the frequency components of a signal. Specify the signal parameters with a sampling frequency of 1 kHz and a period of 1.5 seconds. Generate a summation of two sinusoidal signals, one with a frequency of 50 Hz and amplitude of 0.7 and other with a frequency of 120 Hz and amplitude 1. That is, the generated signal is, $S = 0.7 \times sin(2\pi \times 50t) + sin(2\pi \times 120t)$. It is difficult to identify the frequency components by directly looking at the generated signal in the time domain. Compute the two-sided spectrum of $P2$ using FFT. Then compute the single-sided spectrum $P1$ based on $P2$ by trimming the negative frequency part off. Define the frequency vector f and plot the single-sided amplitude spectrum $P1$. On average, longer signals produce better frequency approximations using FFT. The code is shown in Figure B.23.

```
Fs = 1000;            % Sampling frequency
T = 1/Fs;             % Sampling period
L = 1500;             % Length of signal
t = (0:L-1)*T;        % Time vector
S = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
f = Fs*(0:(L/2))/L;
Y = fft(S);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);

plot(f,P1)
title('Single-Sided Amplitude Spectrum of S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

Figure B.23: FFT Code.

Exercise B2.1

Obtain the FFT response of the following time-domain function

$$y = 0.7\sin\left(2\pi \times 50t\right) \ + \sin\left(2\pi \times 100t\right) \ + 2\sin(2\pi \times 100400t) \tag{11.2.1}$$

with other parameters the same as those in Figure B.23. Find the frequency components of the original signal before FFT. Next, write down the three frequencies from the FFT plot Then compare your results. Are they the same and, if so, why?

This page titled 11.2: Analog and Digital Systems and Fast Fourier Transform (FFT) is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi

Lab B.3 Objectives

After completing this lab session, you will be able to:

1. Explain both low- and high-pass filter design in both continuous- and discrete-time domains.
2. Describe the architecture and Simulink implementation of duel-PID control of the mini-Segway.

Low-Pass and High-Pass Filter Design

This lab section introduces continuous- and discrete-time low- and high-pass filters. Consider the following signal.

$$f(t) = 0.05\sin(2\pi \times 5000t) + 2\sin(2\pi \times 50t) \tag{11.3.1}$$

including two frequency components of 50 and 5000 Hz with different amplitudes. Its wave form in time domain is shown in Figure B.24.

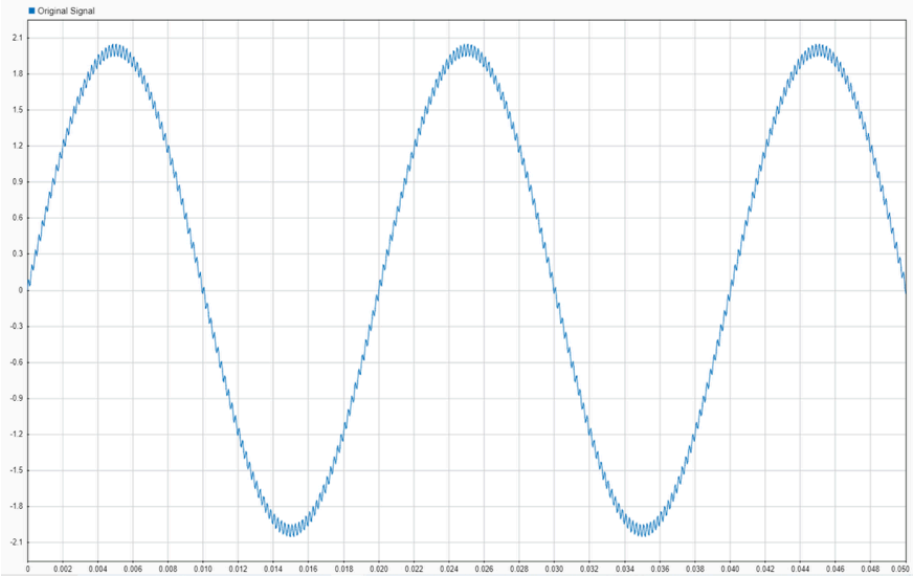


Figure B.24: Wave form of the continuous-time signal in equation 11.3.1.

To develop a simulation model in Simulink, **Clock**, **MATLAB Function**, and **Transfer Fcn** blocks need to be dragged into the Simulink model from the Simulink library, respectively. Figure B.25 shows the constructed model.

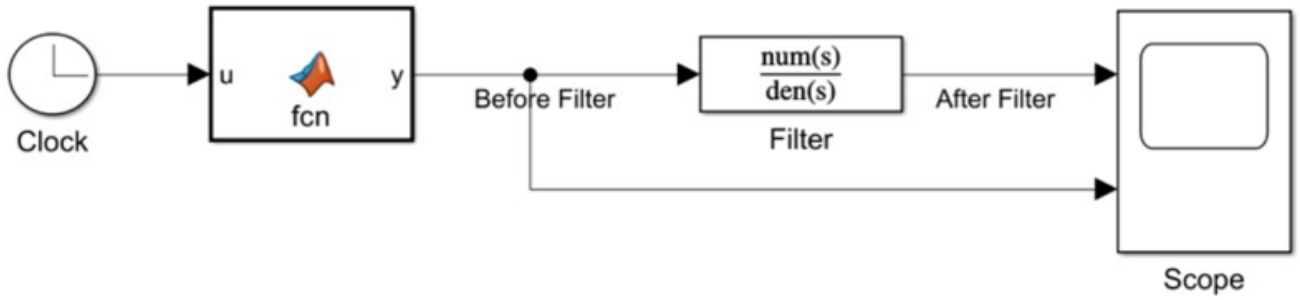


Figure B.25: Filter design validation model.

Continuous-Time Low-Pass Filter Design

The low-pass filter can help to attenuate the high frequency signal components and pass the low frequency ones. Note that the continuous-time low-pass filter was introduced in Section 4.4; see Figure B.26.

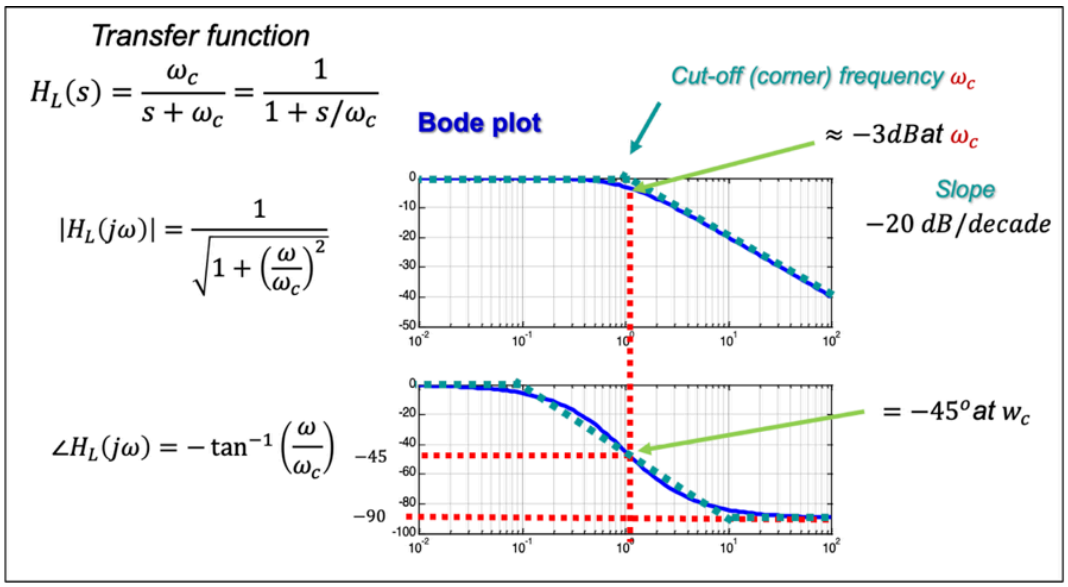


Figure B.26: Frequency responses of a first-order low-pass filter.

The cutoff frequency ω_c (in rad/s) needs to be higher than the expected frequency and lower than the unexpected. In this example and equation 11.3.1, select $f_c = 200$ Hz that is between 50 and 5000 Hz. Hence, the parameters of the **Transfer Fcn** block need to be set as below in the filter calibration window (see Figure B.27).

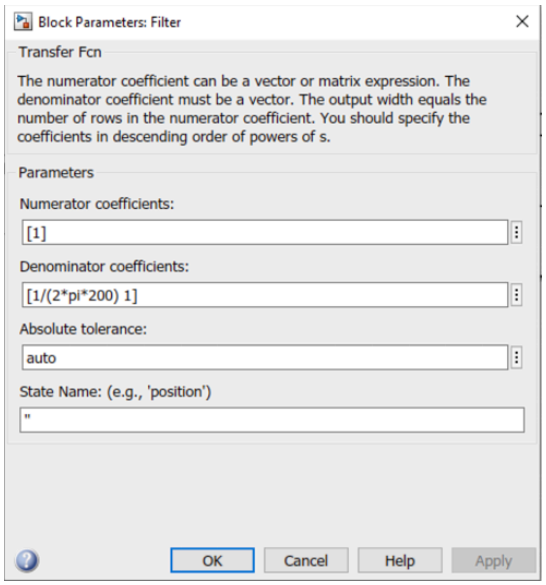


Figure B.27: Low-pass filter parameter selection.

After simulation, the comparison results are shown in Figure B.28. Here, the high frequency component (5000 Hz) has been filtered, but the low frequency part (50 Hz) remains with a phase delay. If $f_c = \omega_c/(2\pi) = 500$ Hz were selected, what would be expected?

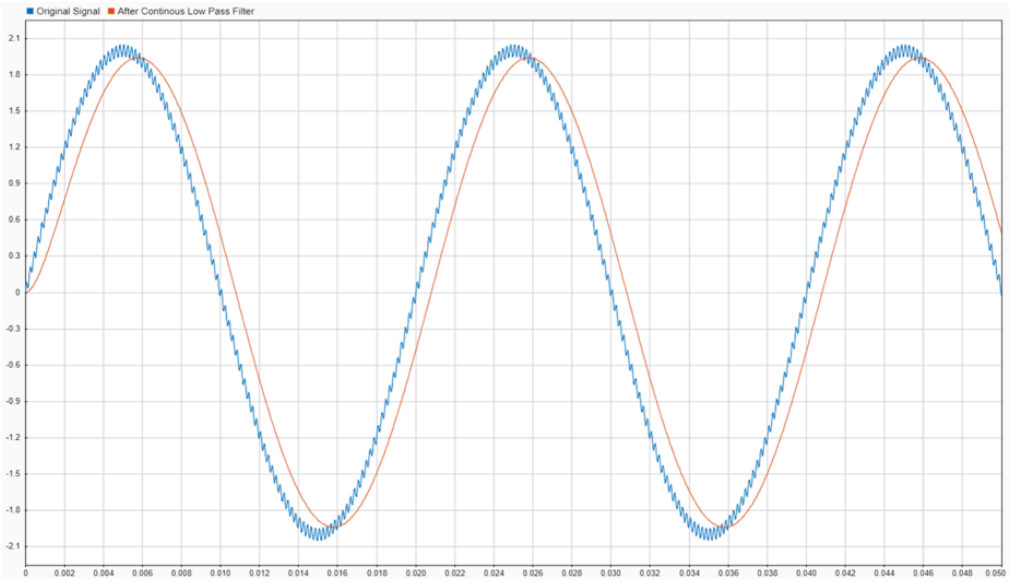


Figure B.28: Comparison of low-pass filtered and unfiltered signals.

Continuous-Time High-Pass Filter Design

The high-pass filter can help to attenuate the low frequency signal components and pass the high frequency ones. The continuous-time high-pass filter was introduced in Section 4.4 and is shown in Figure B.29.

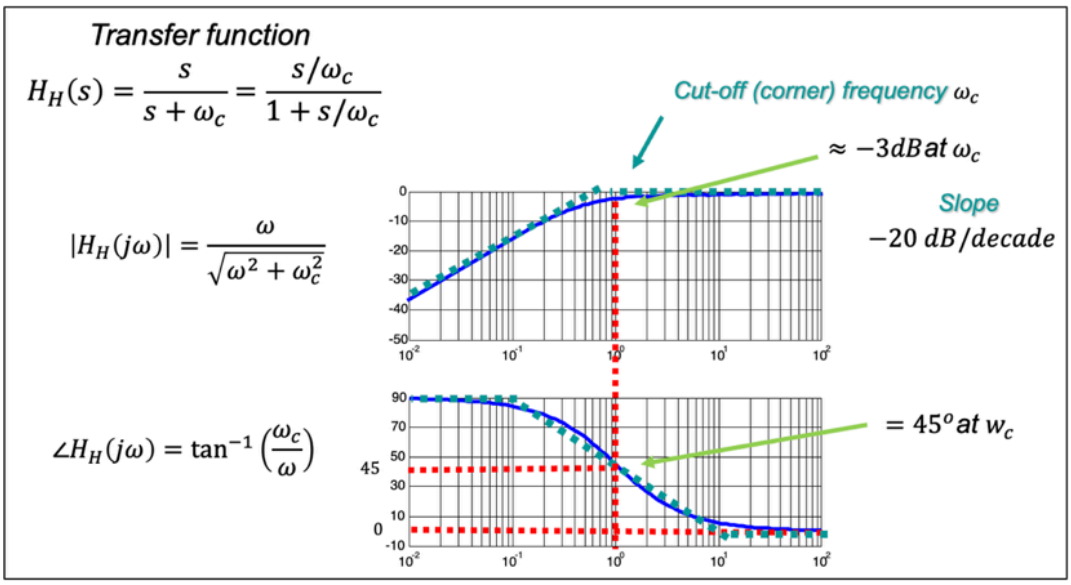


Figure B.29: Frequency responses of a first-order high-pass filter.

Similarly, the cutoff frequency $\omega_c = 2\pi f_c$ (in rad/s) needs to be lower than the expected frequency and higher than the unexpected. In this example, $f_c = 500$ and 1000 Hz are selected, separately, between 50 and 5000 Hz. Parameters of **Transfer Fcn** block need to be set as below, respectively, in Figure B.30.

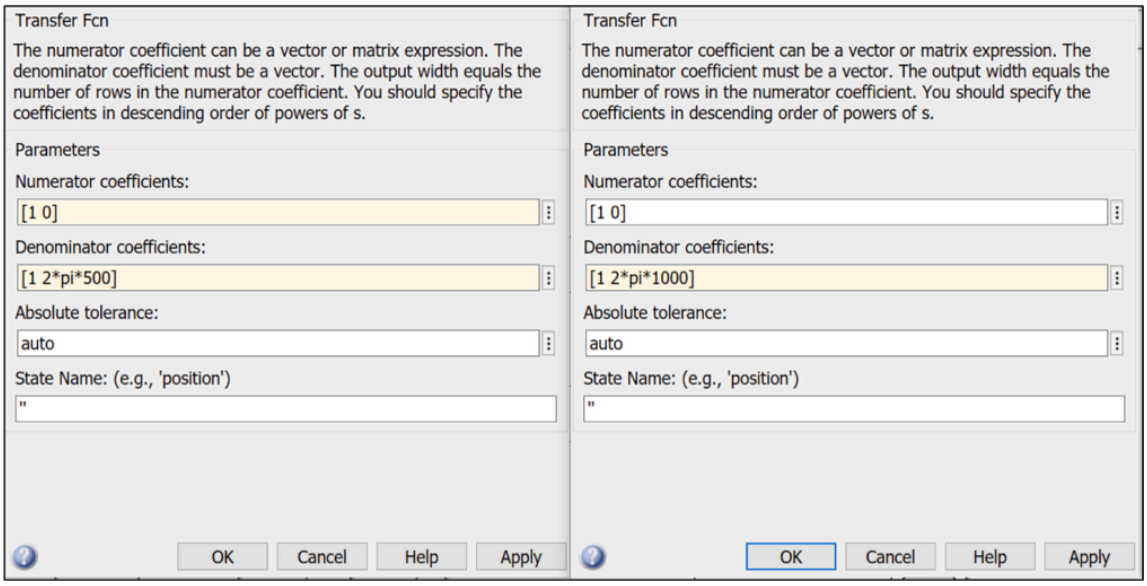


Figure B.30: High-pass filter parameter configuration.

After simulations, the comparison results are shown in Figure B.31, where the low frequency portion has been mostly filtered with suppressed magnitude. However, the amplitude and phase shift are different with different cutoff frequencies. Note that the red line with low cutoff frequency of 500 Hz is preferred.

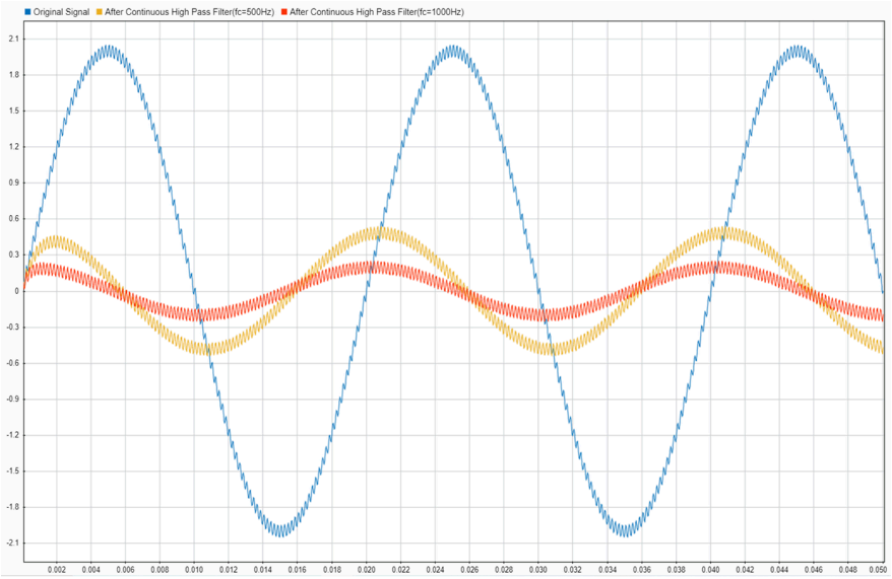


Figure B.31: Comparison of high pass filtered and unfiltered signals.

Discrete-Time Low-Pass Filter Design

The input of a discrete-time low-pass filter needs to be discretized. Hence, the signal is sampled and held using the **Zero Order Hold** block discussed in Lab B.2; it will be implemented here. Add the **Zero Order Hold** and **Discrete Filter** blocks into the Simulink model and connect them, as shown in Figure B.32. The sample period is set to $T = 5.0 \times 10^{-5}$ second (10 time higher than 5000 Hz) for both of these blocks.

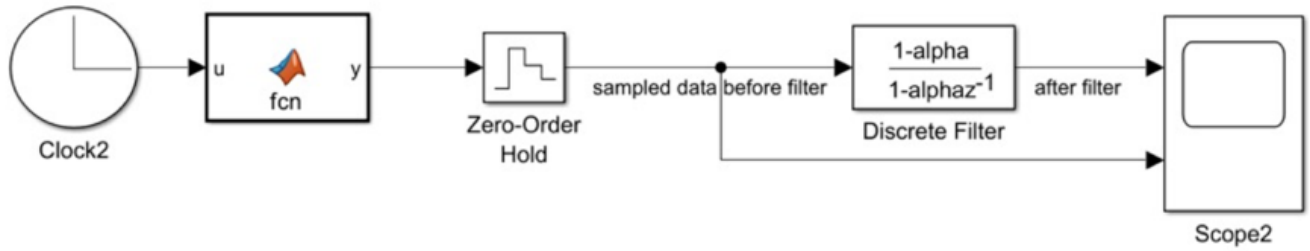


Figure B.32: Digital low-pass filter simulation model.

The transfer function of a discrete-time first-order low-pass filter was introduced in Chapter 4 and again in equation 11.3.2.

$$\frac{Y(z)}{X(z)} = \frac{1-\alpha}{1-\alpha z^{-1}} \tag{11.3.2}$$

The continuous-time cutoff frequency $\omega_c = 2\pi f_c$ (in rad/s) needs to be higher than the expected frequency and lower than the unexpected in the continuous-time domain. In this example, $f_c = 200$ Hz between 50 and 5000 Hz is selected. With the given continuous-time cutoff frequency, parameter α can be calculated using the following relationship between continuous- and discrete-time domains.

$$\alpha = e^{-\omega_c T} \tag{11.3.3}$$

where $\omega_c = 2\pi \times 200 = 400\pi$ and $\alpha = e^{-400\pi \times 5.0 \times 10^{-5}} = 0.939$ with $T = 1/200$ second. The associated command in MATLAB is alpha=exp(-400*pi*5.0*10^(-5)). Using the same method, parameter α for $f_c = 500$ Hz can also be calculated. Figure B.33 shows the comparison results after conducting model simulations, with blue for $f_c = 500$ Hz and red for $f_c = 200$ Hz, where the high frequency component has been filtered.

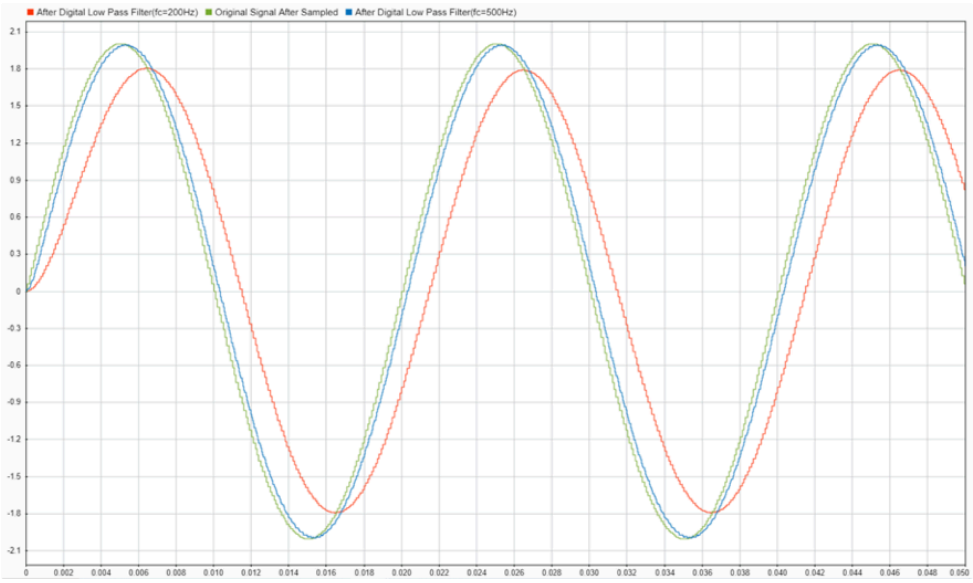


Figure B.33: Comparison of discrete-time low-pass filter simulation results.

Discrete-Time High-Pass Filter Design

For the high-pass filter, the cutoff frequency is selected to be $f_c = 1000$ Hz while the discrete transfer function of the high-pass filter needs to be modified, as shown in equation 11.3.4. Thus, the high-pass filter simulation model shown in Figure B.34 can be adopted as the simulation model.

$$\frac{Y(z)}{X(z)} = 1 - \frac{1-\alpha}{1-\alpha z^{-1}} = \frac{\alpha(1-z^{-1})}{1-\alpha z^{-1}} \tag{11.3.4}$$

where α is determined by equation 11.3.5.

$$\alpha = e^{-\omega_c T} = e^{-2\times\pi\times1000\times5\times10^{-5}} = 0.7304 \tag{11.3.5}$$

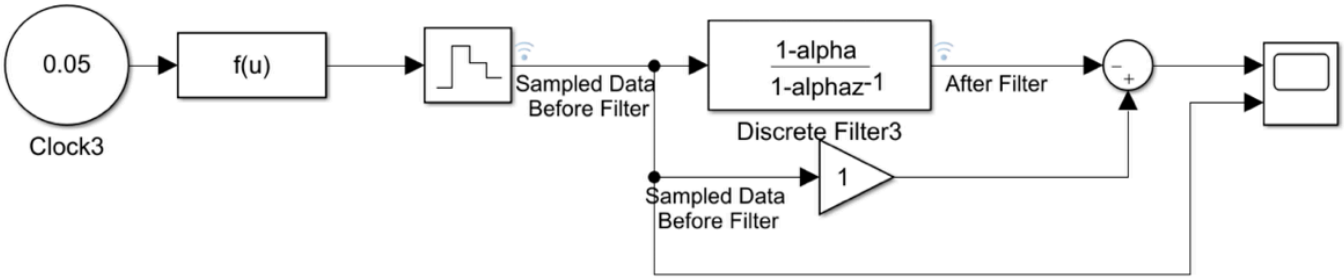


Figure B.34: Discrete-time high-pass filter simulation model.

After simulation, Figure B.35 shows the comparison results after simulation, where the low frequency component has been filtered.

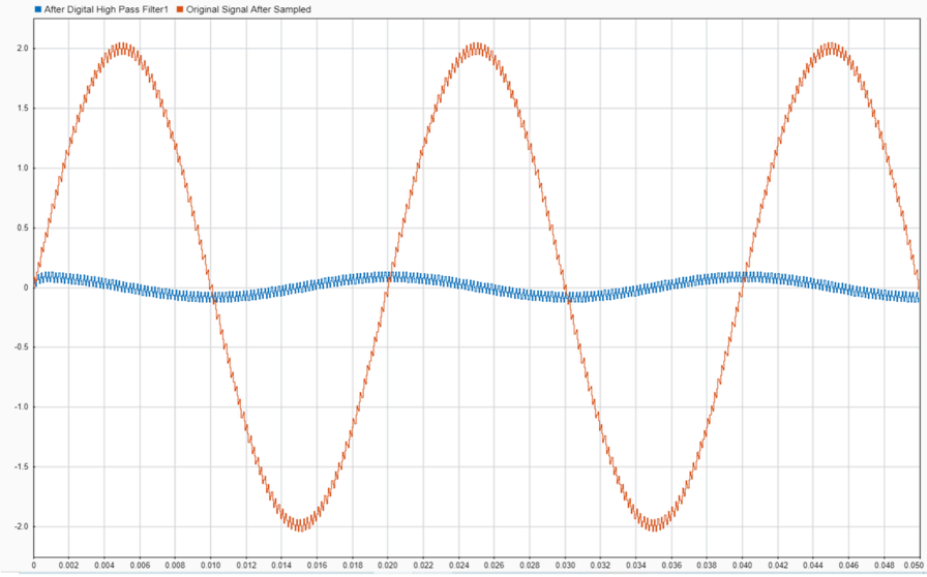


Figure B.35: Comparison of discrete-time high-pass filter simulation results.

Exercise 11.3.1

This exercise is for low- and high pass filter design. Consider the following time-domain signal

$$y = 5\sin(2\pi \times 100t) + \sin(2\pi \times 2000t) + 0.5\sin(2\pi \times 8000t) \tag{11.3.6}$$

and complete the following tasks step-by-step.

1. Design a continuous-time low-pass filter to allow passing the lowest frequency portion (100 Hz) of signal. Show the simulation results and describe the cutoff frequency selected.
2. Design a continuous-time high-pass filter to allow passing the high frequency components (2000 and 8000 Hz) of signal. Show the simulation results and describe the cutoff frequency selected.
3. Design a continuous-time band-pass filter based on contents in Section 4.4. Combine low- and high-pass filters to allow passing a middle frequency component (2000 Hz) of the signal. Show the simulation results and provide both cutoff frequencies selected.
4. Design a discrete-time low-pass filter to allow passing the lowest frequency portion (200 Hz) of the signal. Show the simulation results and provide the cutoff frequency and sample period selected.

Introduction to Arduino MKR 1000 Wi-Fi

Over the decades, Arduino has developed a variety of microprocessor boards with different capabilities and functionalities. For instance, different MKR boards are designed for different communication protocols, such as Wi-Fi and Bluetooth. The most popular board for beginners is the Arduino UNO based on the ATmega328P processor from ATMEL. UNO is the most used and robust board for the entry-level Arduino family; it is a good candidate for getting started with the Arduino platform.

Arduino processor boards released after 2017 use more advanced microprocessors with more memory and enhanced input/output features. For the most part, however, they use the same pinout arrangements and work with existing add-on boards (called shield or carrier) and various add-on components such as sensors, communication chip sets, and actuators.

In 2017, Arduino released the MKR 1000 board, an entirely new design with a new standard pinout, shape, and operating voltage (3.3V). Since then, several boards have been released following this new standard, creating a new subset of Arduino boards called the MKR family. This family of boards are made for those who need a specific digital communication protocol (Wi-Fi, Lora, SigFox, etc.) while keeping the classic Arduino board features.

Arduino MKR 1000 Wi-Fi Board

The MKR 1000 Wi-Fi board is used in Chapters 8 and 9 of this book (see Figure B.36). The board includes a LiPo charging circuit that allows it to run on batteries. It has a Wi-Fi module with a crypto chip to ensure secure communications, and it can be configured to constrain its power consumption to keep it running for long period. The MKR 1000 Wi-Fi board has 28 pins. Some of them have fixed functionalities and others are configurable. Fixed functionality pins include GND and 5V to provide access to the board ground and 5-volt reference voltage lines, respectively. The pins A0 to A6 are analog pins. They can transmit or receive voltage values between 0 and 3.3 volts, relative to GND. Finally, the pins 0 to 14 are digital pins. When these pins transmit data, they can only transmit voltage values of 0 or 3.3 volts, relative to GND. When they receive data, the voltage is interpreted as HIGH or LOW, relative to some threshold between 0 and 3.3 volts. Note that you must never apply more than 3.3 volts to the board's digital and analog pins. Care must be taken when connecting sensors and actuators to assure that the 3.3-volt limit is never exceeded. Connecting higher voltage signals, like the 5V commonly used with the other Arduino boards, will damage the MKR 1000 Wi-Fi board.

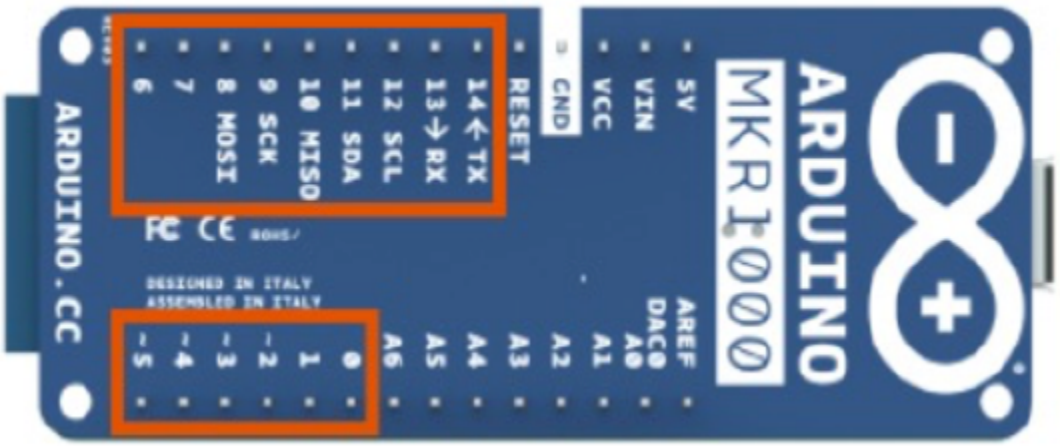


Figure B.36: MKR 1000 Wi-Fi board.

The pins on the Arduino boards can be used to control and communicate with external components, such as LEDs, motors, or sensors. They can be configured either as input or output ones. Input pins enable the board to read values from external components attached to them, such as the state of a button. When pins are configured as output, the board can control and communicate with actuators, e.g., lighting up an LED.

Output pins are said to be in a low-impedance state. In this configuration, they can provide current to other circuits. In the MKR 1000 Wi-Fi board, output pins can source up to 7 mA (milliamps) of current per I/O pin to other devices and circuits. This means, for example, that we cannot drive a motor directly from an output pin because the current will not be sufficient. We need to consider the use of drivers of different types to achieve different tasks like lighting up high power LEDs, driving motors, or activating relays.

Digital pins have only two states: on and off. In the Arduino programming language, these are labeled high and low. We can use the `digitalWrite()` function to set a pin to the desired state, indicating between these brackets which pin we want to act upon and the state we want it to be in.

Installing the Arduino IDE

To test the MKR 1000 Wi-Fi board included in the kit, you need to install the Arduino IDE. The classic Arduino IDE can be downloaded from https://www.arduino.cc/en/Main/Software. Choose the version corresponding to the operating system in your computer, then follow the instructions. More details about installation in different OSs (operating systems) can be found in the link: https://www.arduino.cc/en/Guide/HomePage.

By installing the necessary files to compile programs for the MKR 1000 Wi-Fi, you will also install the drivers needed for your computer to identify the board and communicate with it.

The following steps provide the installation instruction to be used for your own PC:

1. Install Board Driver. Choose **Board** in the “Tools Menu” and then choose **Board Manager** on the top of the list of boards. The window shown in Figure B.37 will pop up. In the drop-down list of **Type**, choose **Arduino**. Further filtering your search by typing **MKR1000** into the search blank will point to the available driver for the MKR 1000 family. Install this driver in your personal computer.

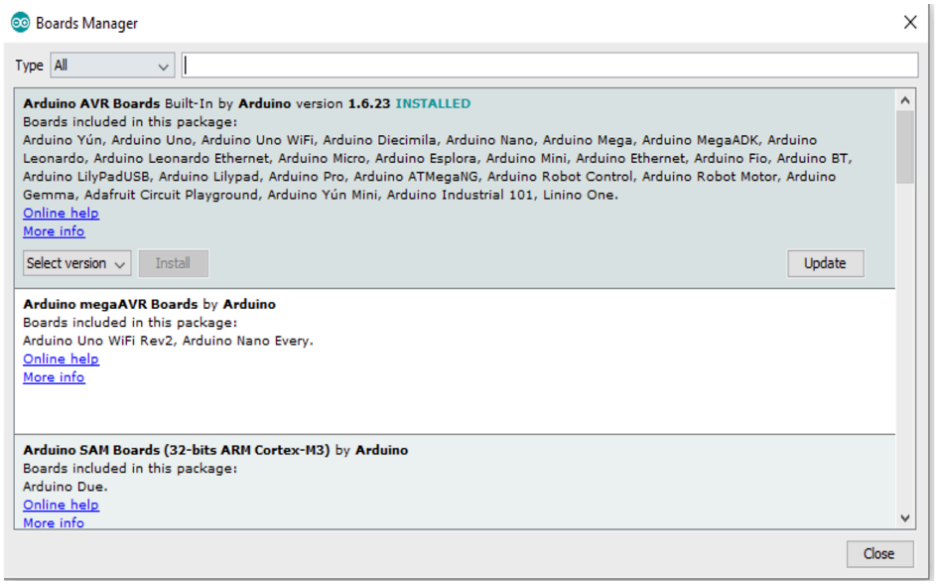


Figure B.37: Board IDE manager.

2. Install the Library. Similarly, choose **Library Manager** in the **Sketch** menu; see **Library Manager** in Figure B.38**.**

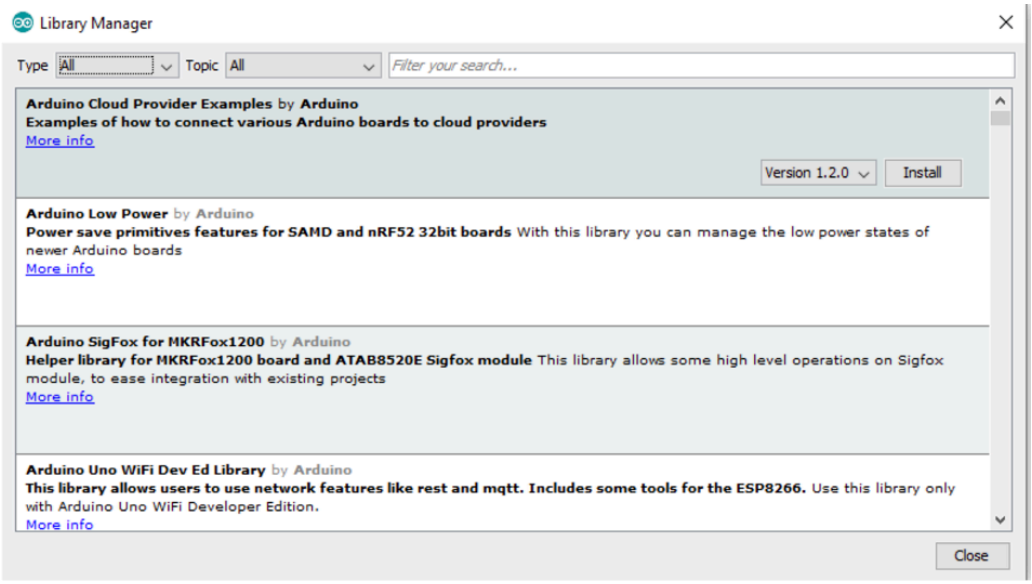


Figure B.38: Library manager.

Two libraries are necessary for this lab session: **MKRMotorCarrier** and **WIFI101.** Select and install them.

Three main computer operating systems (Windows, Linux, and Mac OS) handle the connection of new USB devices differently. For example, the term COM followed by a number in Windows is different from that in Linus or Mac OS.

Programing by IDE

(This task should be completed during the lab session and used to verify if the software is installed successfully.) Choose **Examples** from the drop-down menu and then single-click **Blink** from the item list of “Basics.” The Blink program should be as

shown in Figure B.39.

```
Blink §
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, HIGH);    // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

Figure B.39: Blink program.

The mode of pin 6 is set to output so that pin 6 can be set to the desired state (on or off) for lighting an LED. Try to tune the value in the command delay(*)to observe the changes.

Communicate between Arduino and MATLAB/Simulink.

The remaining lab sessions, including this one, use the MATLAB/Simulink as the programming tools for real-time control of the Arduino MKR 1000 Wi-Fi board. The following discussion details how to use this tool for the MKR 1000 Wi-Fi.

MATLAB and Simulink Package Installation

These instructions for installing MATLAB and Simulink can be used for your own PC. You will need the MathWorks products (toolboxes) listed in Lab B.0 to program the projects with the Arduino Engineering Kit. If MATLAB is installed, you can check if you have the required products by typing ver in the command window; then press “Enter” (or “return”) in the MATLAB command window. After completing the installation, install several add-ons that will be used in future lab sessions. First reopen MATLAB with administrator access. Then select **Add-Ons** from the MATLAB tool bar, as shown in Figure B.40. Click on **Get Add-Ons**.

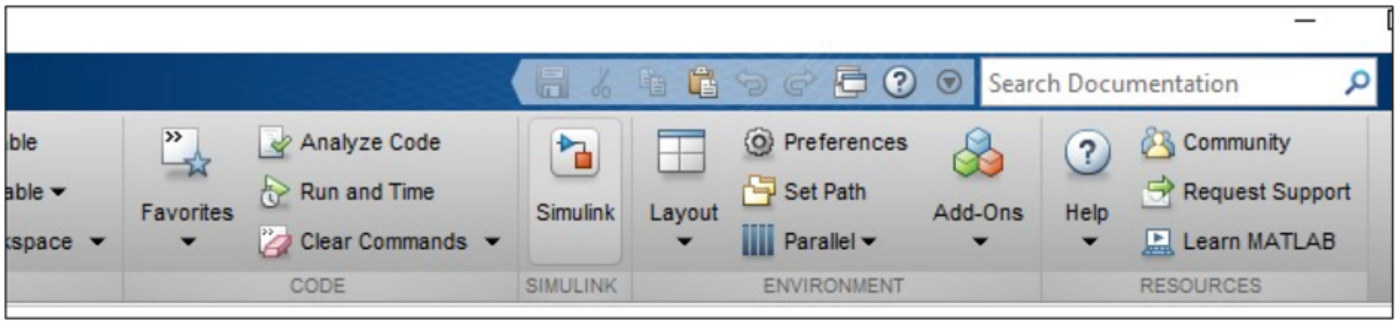


Figure B.40: Matlab tool bar.

Install these add-ons in the exact order provided below

1. MATLAB Support Package for Arduino Hardware: Acquire inputs and send outputs to Arduino boards and connected devices.

2. Simulink Support Package for Arduino Hardware: Run Simulink models on Arduino boards.
3. MATLAB Support Package for USB Webcams: Acquire images and video from USB webcams.
4. Arduino Engineering Kit Hardware Support: Select the right version depending on your MATLAB release (Arduino Engineering Kit Hardware Support R2019a or later, e.g., 2024b) and install MATLAB and Simulink Support Package for the Arduino MKR Motor Carrier and other components used in the Arduino Engineering Kit.

Connection Setup for Matlab and Arduino

You can set up the MKR 1000 Wi-Fi board in MATLAB. Using this method, you can check the connection status when you get any errors during programing between MATLAB/Simulink and Arduino. Type the following command in the MATLAB Command window to start:

```
>> arduinosetup
```

This command will open the Hardware Setup window shown in Figure B.41. Select **USB** from the available options and click “Next.”

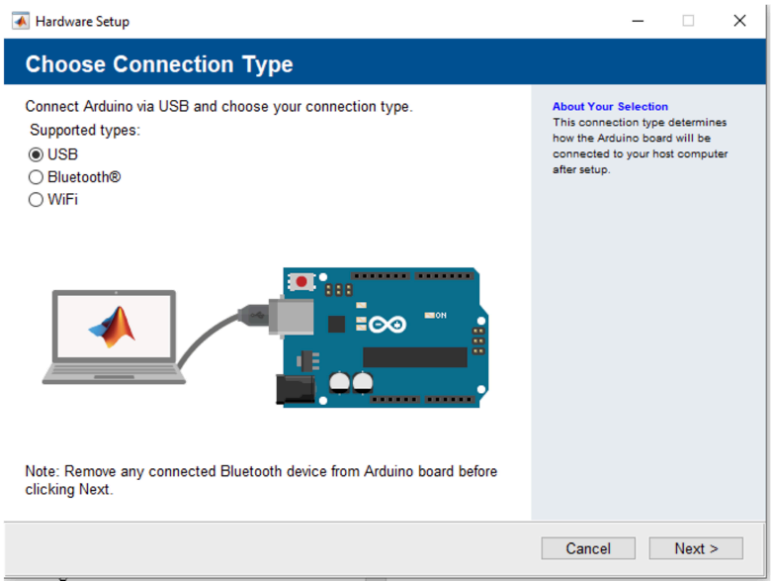


Figure B.41: Hardware setup window.

Next, select your Arduino board type, the port through which you will communicate with it, and the external devices you will need for access. You will need to access the MKR motor carrier board (see Section 7.4). This board interfaces among the MKR 1000 Wi-Fi, DC motors, encoders, and servo motors. This example uses one DC motor and its associated rotary encoder. Set your board to MKR 1000 Wi-Fi, set the communication port to the COMx, and select **Arduino/MKRMotorCarrier** and **RotaryEncoder**, as shown in Figure B.42. Click **Program**.

This step may take several minutes to complete. After the libraries have successfully loaded to the MKR 1000 Wi-Fi board, click **Next** and complete the remaining dialog. The firmware you are uploading will enable the Wi-Fi board to communicate back to your computer, where MATLAB/Simulink executes commands with the help of the information that is captured by sensors and sends it back to actuators via the same communication channel.

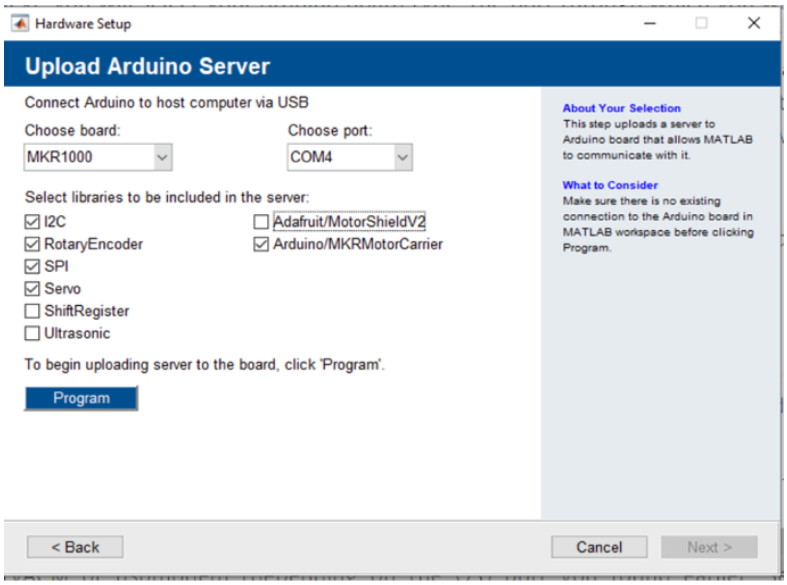


Figure B.42: Upload server window.

Exercise B3.2

Setup communication between Simulink and Arduino for Arduino MKR 1000 Wi-Fi board. Follow these six steps to complete the exercise.

1. Choose the **Pulse Generator** block from the Simulink Sources library and the **Digital Output** block from the "Simulink Support Package" within the Arduino hardware library. Connect these blocks as shown in Figure B.43.

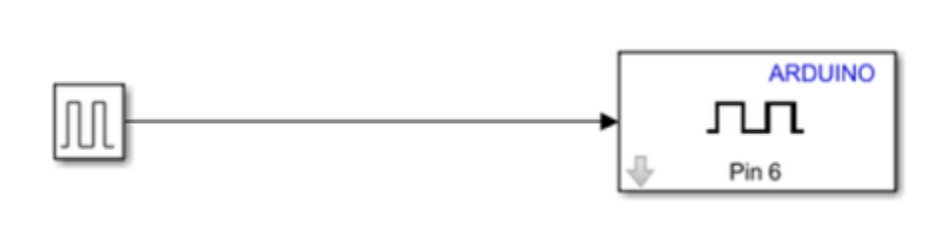


Figure B.43: Simulink model for lighting an LED.

2. Edit these parameters in the **Pulse Generator** block as shown in Figure B.44, with amplitude equal to 1, period 2 seconds, and 50% pulse duty-cycle.

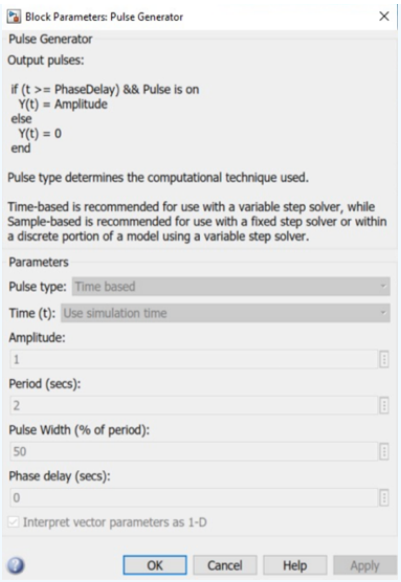


Figure B.44: Pulse generation block.

4. Change Pin number 6 in the **Digital Output** block for illuminating LED lights.
5. Choose the External mode in the Simulink tool bar and type in inf for the simulation time; see Figure B.45.



Figure B.45: External mode setup.

5. Single-click on the icon  in the Simulink tool bar to configure the model's parameters. In the **Configuration window** block, configure the following items:
   - In the Hardware Implementation section, choose Arduino MKR1000 as the Hardware board, as shown in Figure B.46.

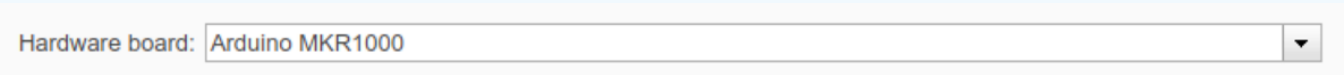


Figure B.46: Hardware configuration.

   - In the Hardware board settings, choose **Serial** for External mode.
   - In the Solver configuration, set up the **Fixed step** size as 0.1 second.

6. Now the basic model has been created. Click the **Run with IO** button in "Hardware" from the tool bar to launch the model.

If the build process completed successfully, the LED light on the Arduino MKR1000 Wi-Fi board should blink. Note that the LED light will not light if the Arduino MKR1000 Wi-Fi board is inserted into the MKR motor carrier board.

This page titled 11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11.4: DC-Motor Control Utilizing Arduino Hardware

Lab B.4 Objectives

After completing this lab section, you will be able to:

1. Describe the feedforward control of a DC motor.
2. Interpret the encoder reading for motor speed measurement.

Feedforward DC Motor Control

Lab B.3 developed the connection between Arduino and Simulink. In Lab B.4, the motor speed will be controlled by a PWM signal as a feedforward control. First, let's look at the **Arduino Device Driver** blocks in Simulink. In the "Simulink Library" browser, navigate to "Simulink Support Package for Arduino Hardware/MKR Motor Carrier," and examine the blocks inside as shown in Figure B.47.

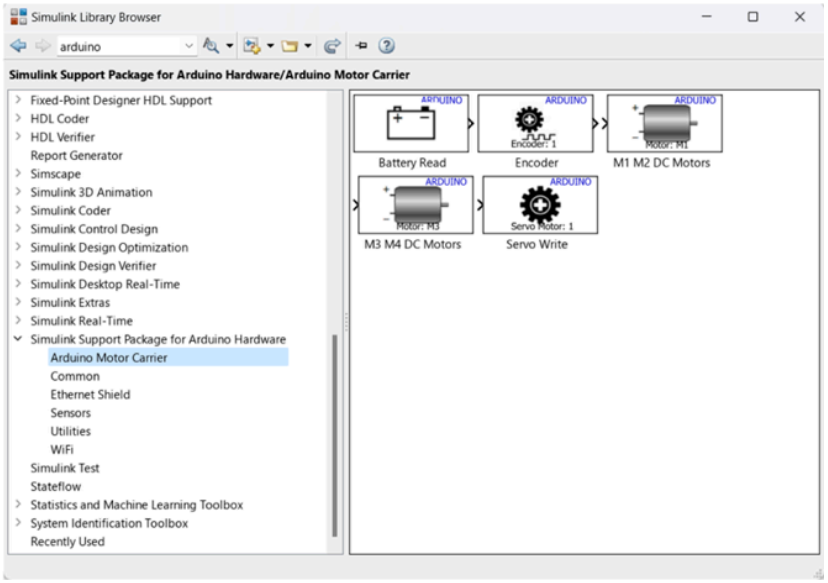


Figure B.47: Simulink support for Arduino.

Driving a DC Motor at Constant Speed. The following six-step process runs the DC motor in a constant speed using feedforward control.

1. Drag the **M3M4DCMotors** block in the "Simulink Support for Arduino MKR Motor Carrier" library into your model.
   The motor port property maps the block to one of two labeled DC-motor ports (M3 or M4) on the Motor Carrier board. The **M3M4DCMotors** block requires one input signal, where the drive command is expressed as a value between -255 and 255 counts. The input controls the duty-cycle of the PWM waveform. An input value of 0 produces a 0 percent duty-cycle and an input value of 255 produces a positive 100 percent duty cycle.
2. Double-click the **M3M4DCMotors** block and configure it as shown in Figure B.48.

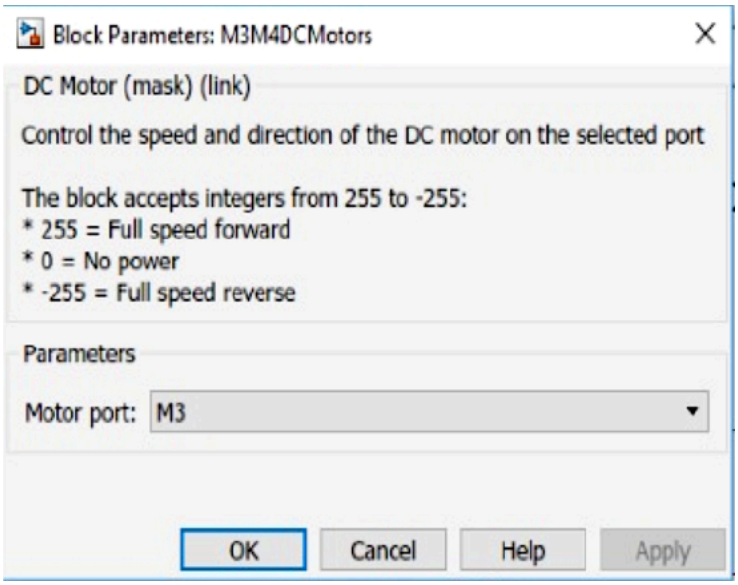


Figure B.48 Parameter of "M3M4Motors" block.

3. Choose the **Constant** and **Saturation** blocks from the **Simulink Commonly Used** blocks. Connect the three blocks in sequence as shown in Figure B.49. Set the **Constant** block value to 1.

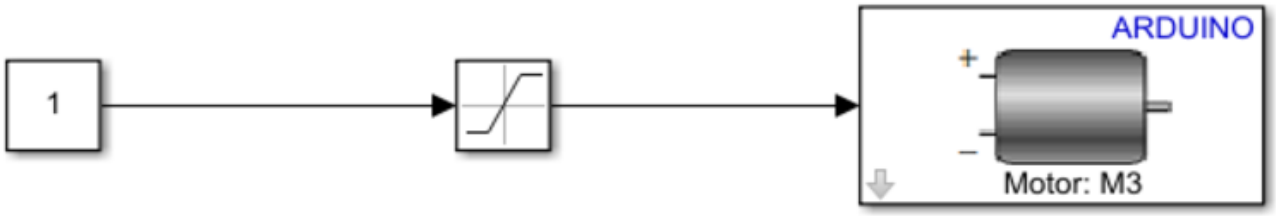


Figure B.49: Simulink model for DC-motor speed control.

4. Edit parameters in the **Saturation** block to protect the motor and set the limit to ± 255, as shown in Figure B.50.

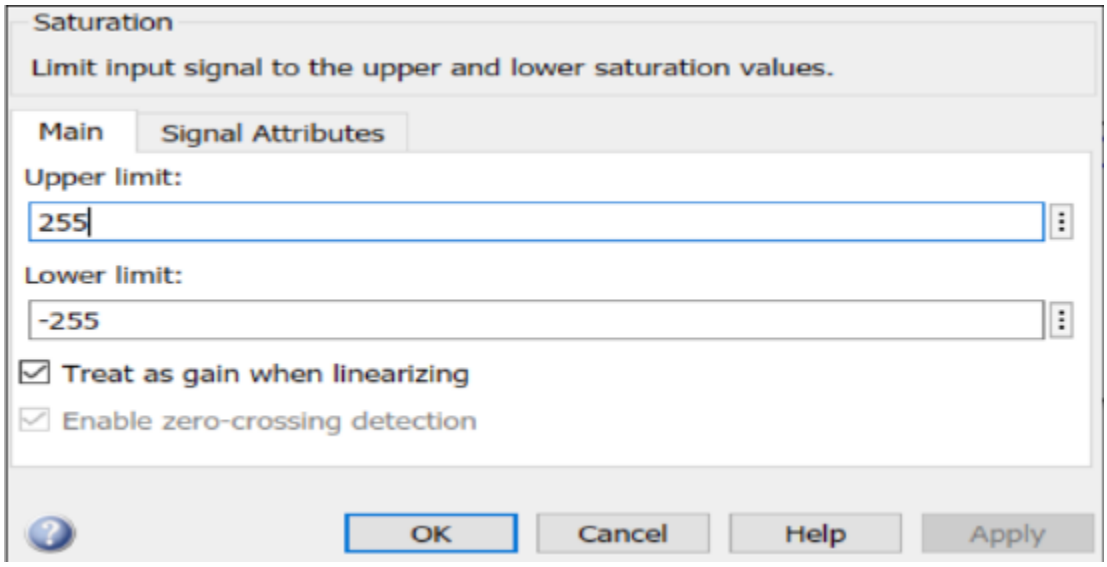


Figure B.50: Saturation block parameters.

5. Recall the configuration about the running mode in Lab B.3. The external mode and **Hardware** board need to be updated every time when you build a new model. Now change the simulation mode to external with correct hardware configuration.

6. Set up the simulation time (Stop Time) to infinity, as shown in Figure B.51.

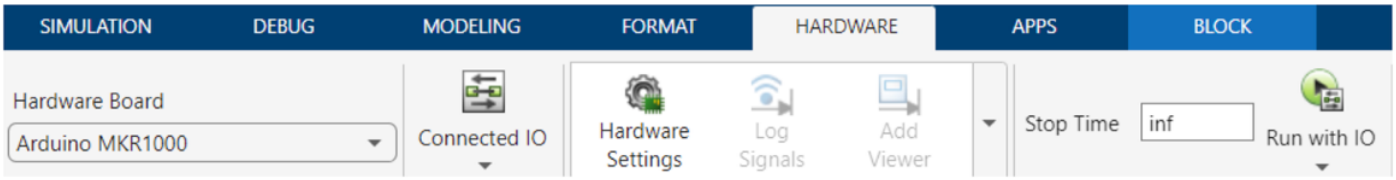


Figure B.51: Simulation time setup.

You now have a functional system that theoretically drives the motor based on the desired PWM signal. The next section addresses the model to read the hall-effect encoder count and calculate the motor rotational speed.

Encoder Reading for Speed Calculation

The following six steps are related to reading the hall-effect encoder signal as pulse counting and converting the encoder counting to the rotational speed based upon the number of teeth per revolution.

1. Locate the Arduino **Encoder** block in the "Simulink Support for Arduino MKR Motor Carrier" library (see Figure B.52). Drag it into the Simulink model previously built.

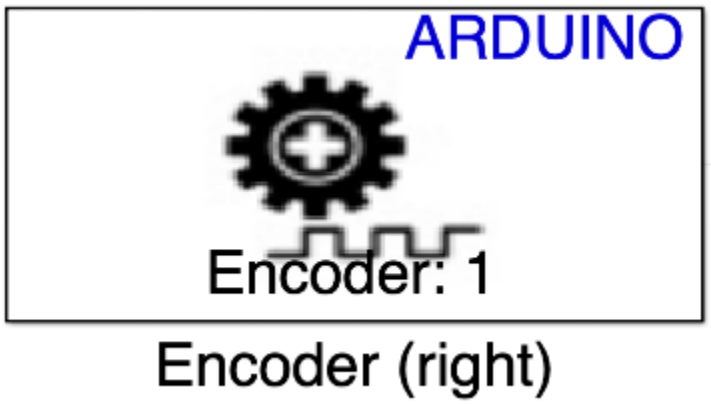


Figure B.52: Arduino "Encoder" block.

2. Open the block dialog window; set encoder number to 1 and sample time to 0.01 second. In this case, the encoder needs to be connected to encoder channel 1. Next, test the **Encoder** block in isolation by attaching a temporary **Display** block (from the Simulink "Sinks" group) to the output, as shown in Figure B.53.



Figure B.53: Encoder display.

3. Now turn on the "Motor Carrier" board power switch and run the Simulink model. Watch the encoder count in the **Scope** block.
4. Next, calculate the angular speed of the motor axis speed. In the Simulink model, insert four blocks: a **Sum** block in the Simulink "Math Operations" group along with a **List of Signs** block calibrated as '+-', a **Unit Delay** block in the Simulink "Discrete" group with sample period of $T_s = 0.01$ second, and a **Gain** block with a gain of $1/T_s$. Connect these new blocks as shown in Figure B.54 to calculate the change in counts divided by the sample time. Note that the **double** block is used to

convert fixed-point number to float-point one.

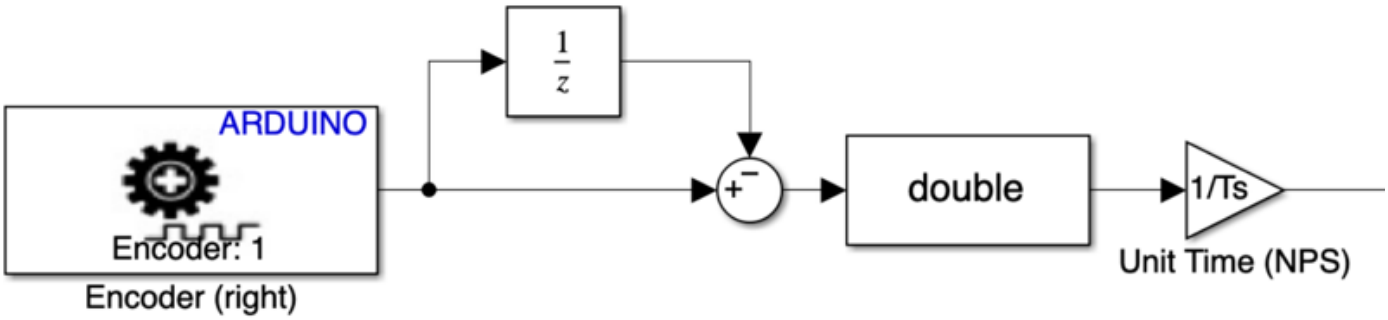


Figure B.54: Encoder count difference for speed calculation.

5. For the encoder used, one revolution will generate 1320 counts (see Section 6.2). Information in Section 6.2 can be used to convert speed counts per second to RPM (revolution per minute) with a gain block; see Figure B.55.

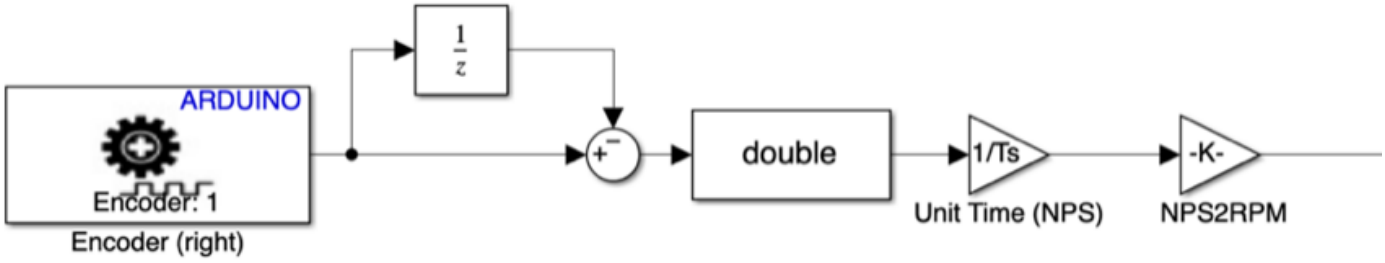


Figure B.55: Rotational speed unit conversion.

6. The radius of the wheel is 0.034 meter. Use the same method to convert the speed unit into meter per second utilizing information in Section 6.2; see Figure B.56.

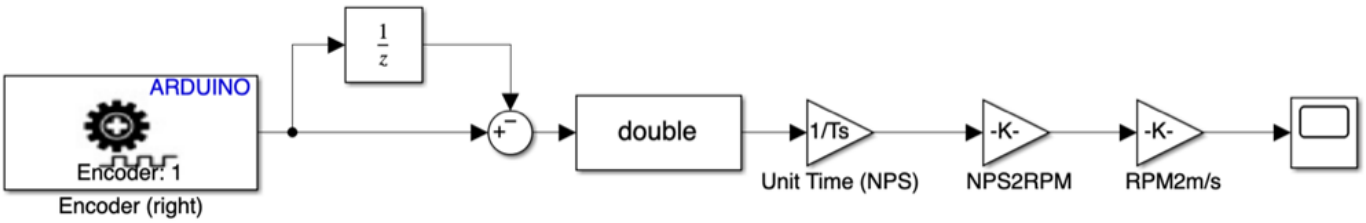


Figure B.56: Speed unit conversion from RPM to meter per second.

Exercise B4.1

Complete the following five tasks step-by-step.

1. Run the Simulink model developed. After the software development process is completed successfully, double-click the **Constant** block in Figure B.49 and change the parameters step-by-step to -100, -50, 0, 50, and 100. Calculate the corresponding PWM duty-cycle associated with these parameters and plot their speed trajectories.
2. Simulate acceleration from 0 to full speed by using the **Ramp** block to replace the constant block. Set slope as 10 with initial output as 0 and use the **Saturation** block to protect the DC motor. Plot the resulting speed trajectory.
3. Plot the relationship between the PWM duty-cycle and motor speed.
   a. Use the model with "Ramp" input with slope of 5 and initial output of -255. Plot the graph with the x-axis as PWM duty-cycle and the y-axis as motor speed. (*Hint: A **Divide** block is required to convert ramp input to PWM duty-cycle. A **Scope** block can be used for plotting*).

b. In the $x - y$ graph, circle the PWM duty-cycle range with zero motor speed. Find out if the zero duty-cycle is single point or a region and why?

4. Drive two DC motors, respectively, with 50% and -50% PWM command (i.e., in opposite directions). Predict the movement of the mini-Segway. Share your results with the lab TA (teaching assistant).

This page titled 11.4: DC-Motor Control Utilizing Arduino Hardware is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11.5: Simulink GUI (Graphic User Interface) Design and Programming

Lab B.5 Objectives

After completing this lab section, you will be able to:

1. Create a Simulink subsystem for encoder reading.
2. Develop a GUI (graphic user interface) in Simulink for real-time applications.
3. Develop a GUI for tuning a PID controller.

Lab B.4 addressed how to build a Simulink model to read sensor information from hardware such as an encoder. For the mini-Segway project, at least five subsystems are required, including DC motor, encoder, MPU 6050, closed-loop controllers, and response display. The GUI (graphic user interface) design will be helpful for developing a concise and clear model that is easy to understand. The partially completed subsystem models shown below in Figure B.57 will be provided before the lab section. Then, follow the steps below to complete three sections: an encoder subsystem model, a GUI design for displaying mini-Segway motion and control, and a GUI design for PID speed control of the mini-Segway, marked as Sections B5.1, B5.2, and B5.3, respectively. The rest of this lab section discusses these three tasks in detail.

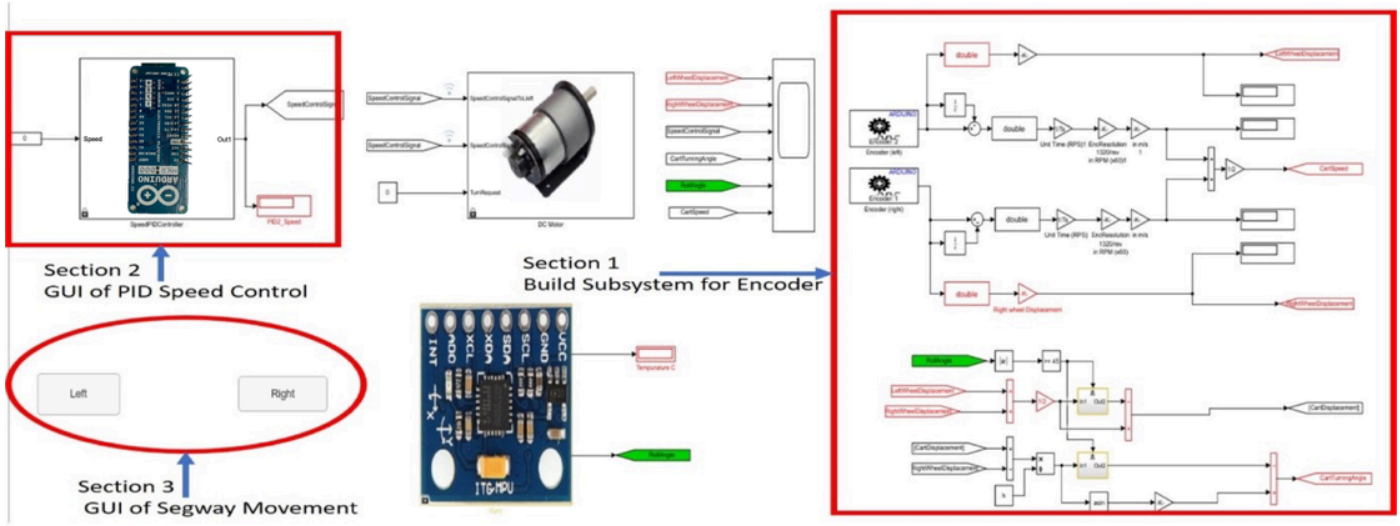


Figure B.57: Subsystem model architecture.

Build Subsystem for Encoder

In the “Encoder” subsystem, several **Goto** and **From** blocks are used. All **Goto** blocks are set as “Global” variables so that they must have a unique variable name for its tag. If interested, double-click any block to view its parameter.

Exercise 11.5.1

This lab exercise focuses mainly on the subsystem design for reading the encoder signals from the left and right motor axles, where three **Goto** blocks are defined for outputs of the Encoder subsystem. They are **LeftWheelDisplacement**, **RightWheelDisplacement**, and **CarSpeed**. The following steps help develop the subsystem.

1. Highlight a box around the blocks with associated connecting lines to be included in the subsystem. That is, select all the components in the red rectangle on the upper-right part of Figure B.57.

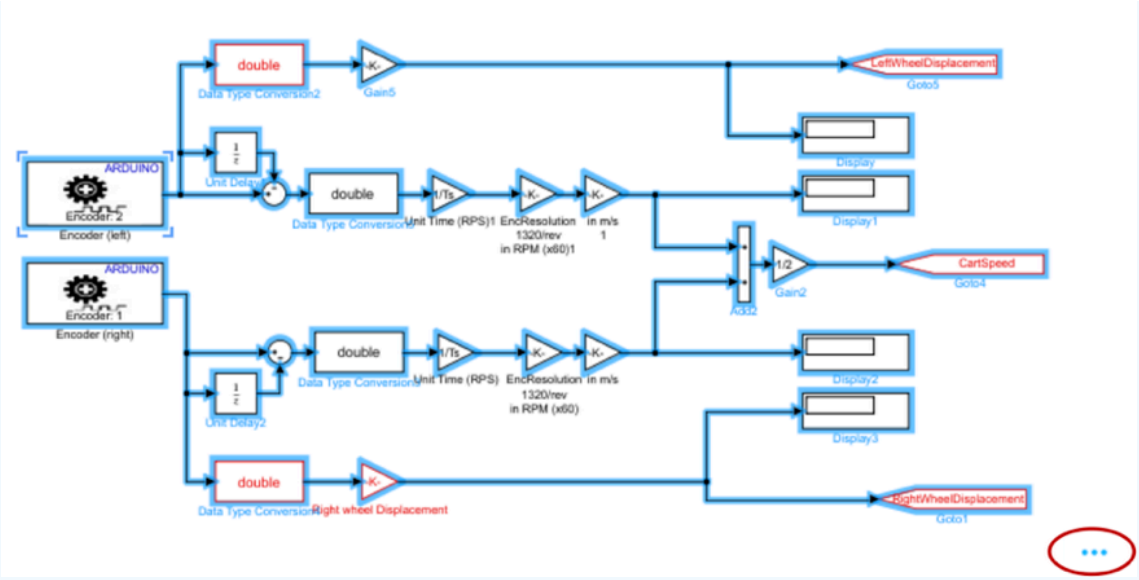


Figure B.58: Highlighted encoder subsystem.

2. After these blocks and connecting lines are selected, there will be a symbol shown in the lower right corner highlighted by the red circle in Figure B.58. Moving the mouse over it extends the action bar, as shown in the lower right corner of Figure B.59.

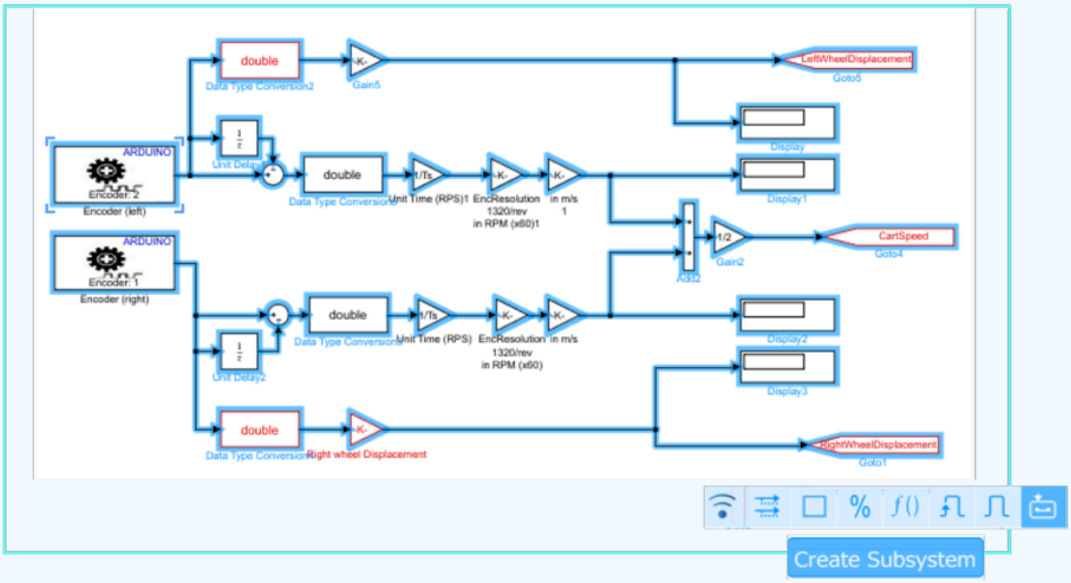


Figure B.59: Displayed action bar.

3. Single click on the item to the far right in the extended action bar (highlighted Figure B.59) to create the subsystem. Details of this created subsystem can be viewed by double-clicking the subsystem block shown in Figure B.60.

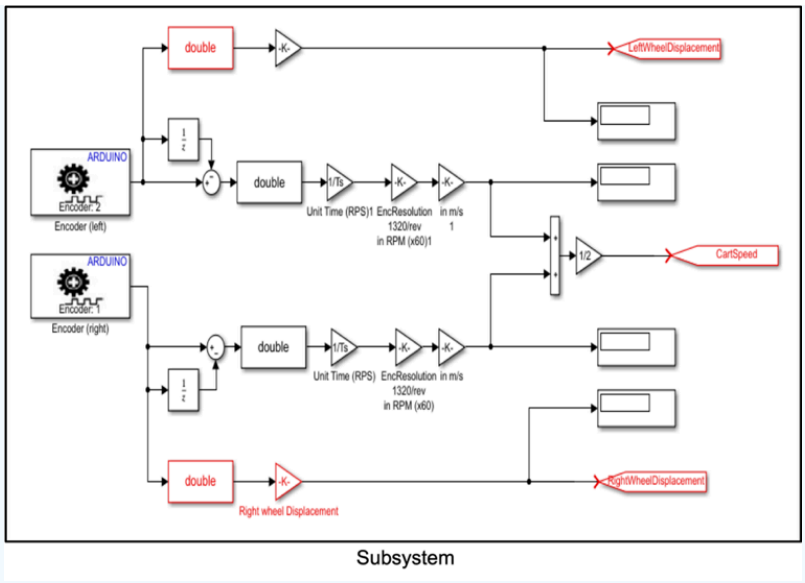


Figure B.60: Created subsystem block.

4. Add three output blocks from the "Ports & Subsystems" library and edit their labels, respectively, into **LeftWheelDisplacemet**, **RightWheelDisplacement**, and **CarSpeed**. Now the **Subsystem** block should have three outputs as shown in Figure B.61.
5. Disconnect three **Goto** blocks labeled as **LeftWheelDisplacemet**, **RightWheelDisplacement**, and **CarSpeed**.

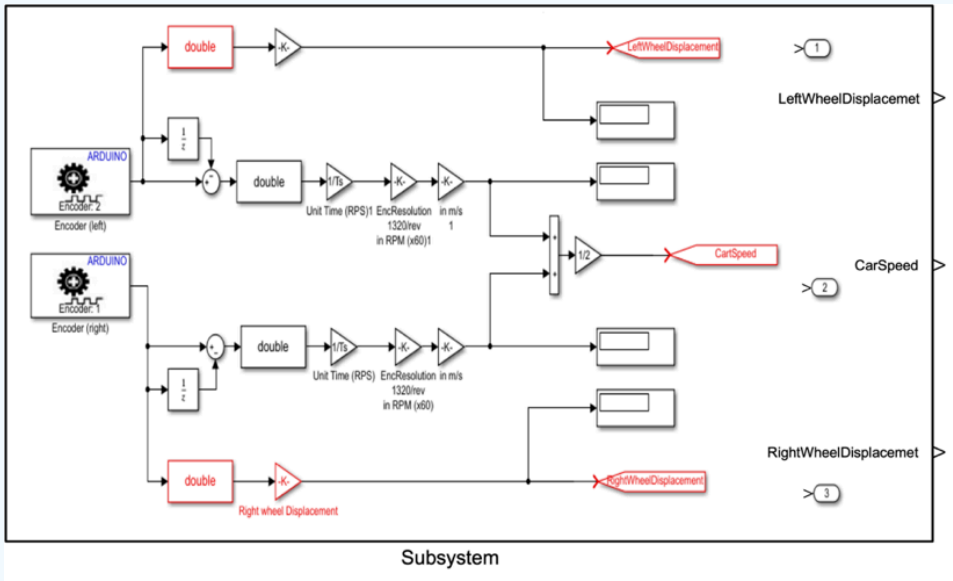


Figure B.61: The subsystem block with three outputs.

6. Connect three **Output** blocks in sequence to the positions where the **Goto** blocks were connected previously. Now the inside of the model should look like Figure B.62.

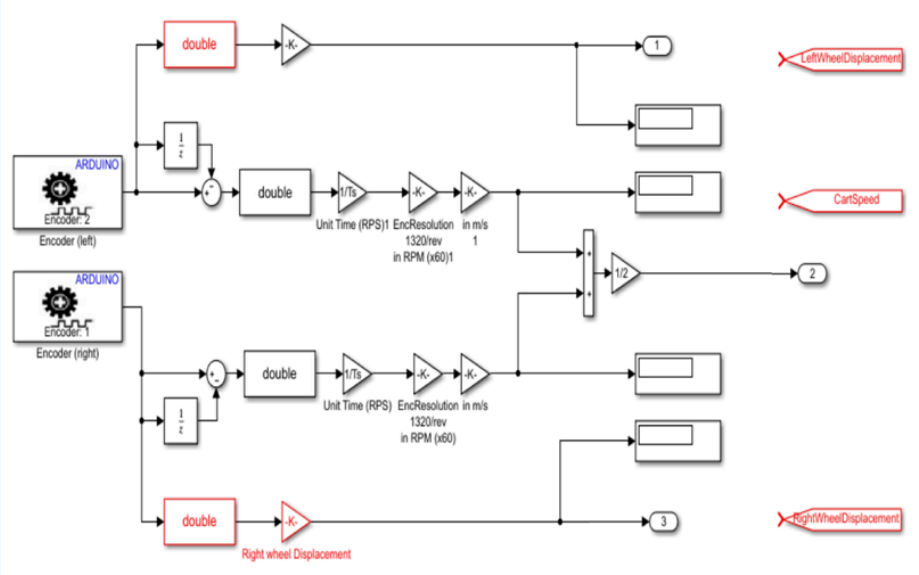


Figure B.62: Inside of subsystem with output blocks connected.

7. Delete the three **Goto** blocks and paste them to the outside of the subsystem (upper level).
8. Connect these three **Goto** blocks to the outputs highlighted in Figure B.61 by matching the associated signals inside of the Subsystem block.
9. Right click the Subsystem block, choose “Mask” and edit the icon image. The dialog window of the “Edit Mask” icon image will pop up; see Figure B.63.
10. Browse to the folder where you stored the picture of the encoder. Select the desired encoder picture from the folder.
11. Right click on the encoder Subsystem block, choose “Mask and Edit Mask”. The Mask Editor dialog will pop up. Change the icon transparency to opaque with ports, as shown in Figure B.63. Then click “OK.”
12. Now your encoder subsystem should be similar to Figure B.64. Make sure the names of ports are exactly the same as these of the **Goto** Blocks.

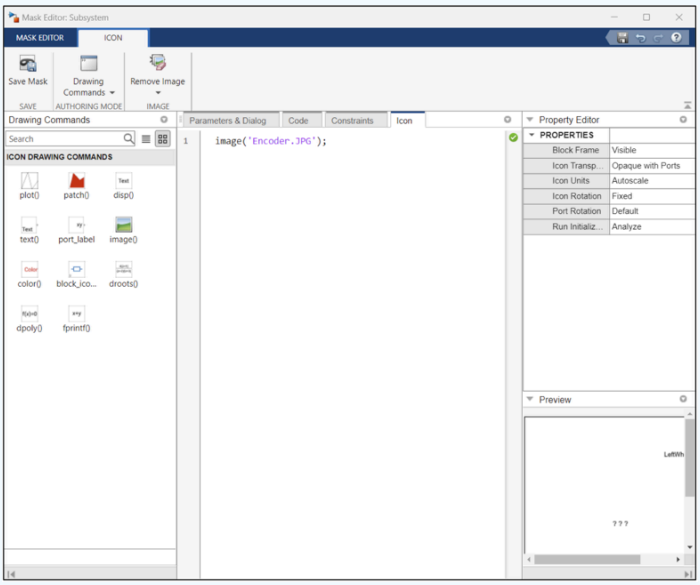


Figure B.63: Dialog window of “Mask Editor”.

All the other subsystems in this mini-Segway model were built by the same method. Notice that this model provided only left and right movement control. However, we also need forward and backward movement control for the mini-Segway. Section B5.2 introduces the process of creating a Left button. After that, you will need to complete the GUI design for Forward and Backward buttons.

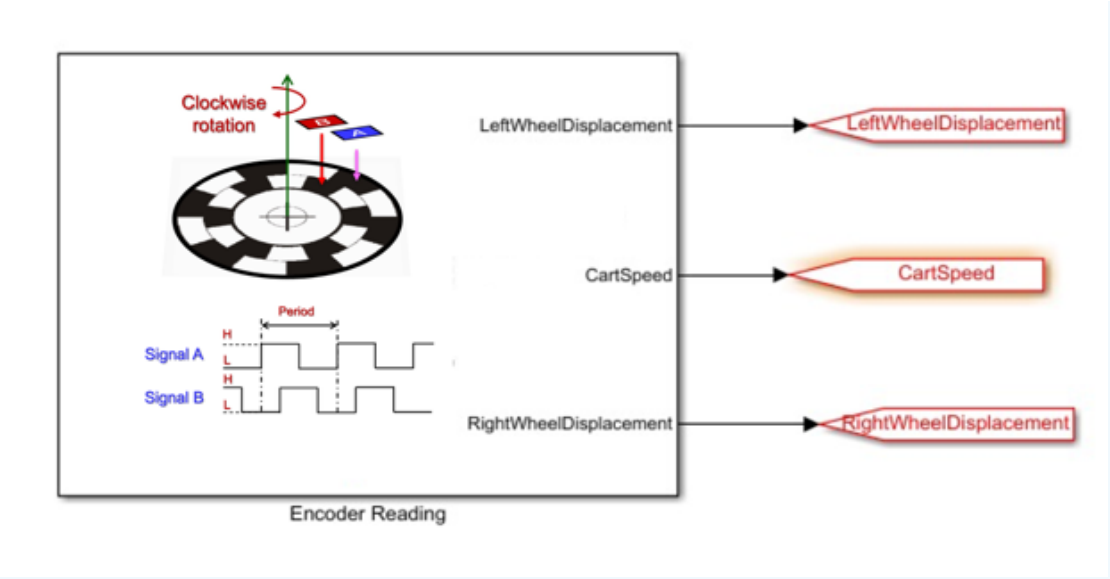


Figure B.64: Finished encoder subsystem block.

GUI Design for Mini-Segway Motion Control

This section discusses how to develop a graphic user interface (GUI), utilizing the push button in Simulink, to control the mini-Segway motion while the software is running.

Exercise B5.2

Follow this step-by-step list to develop a GUI.

1. Select the **Push Button** block from the Simulink Dashboard library. Move the block to the available location in your model.
2. Single click the connect symbol above the **Push Button** block, as shown in Figure B.65.



Figure B.65: “Push Button” block.

3. Single click on the **Constant** block connected to the “TurnRequest” input port of the DC-motor subsystem, as shown on the left of Figure B.66. The Connect dialog will then pop up, as shown on the right of Figure B.66.

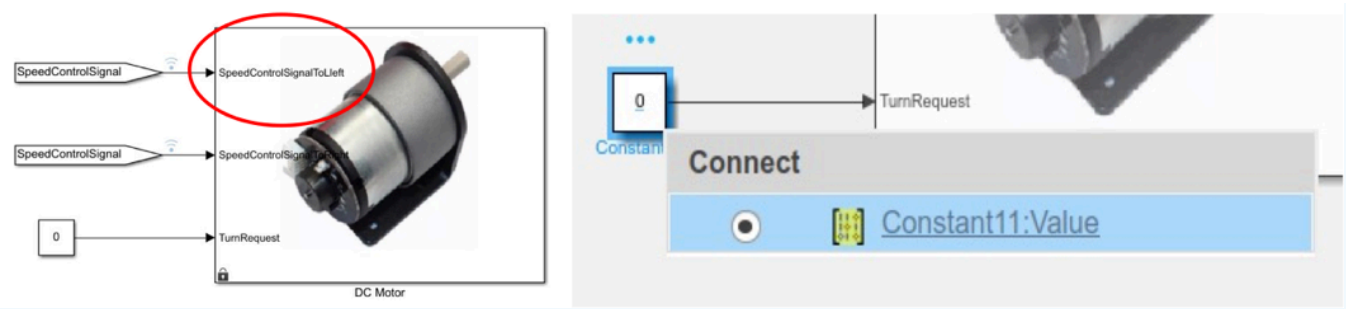


Figure B.66: "TurnRequest" constant input to DC motor and associated connection dialog.

4. Single click the circled **Constant** block in the Connect dialog to link this constant value to your button. After this click, press "ESC" to escape the link operation. The appearance of the **Button** block will then be changed, as shown in Figure B.67.



Figure B.67: Linked button block.

5. Now the connection is tested to validate its operation. If the connection has been established successfully, both connected blocks will be highlighted when you single click and hold on the connected **Button** block.
6. Double-click the **Button** block to change the set value, as shown in Figure B.68. Then, click "OK."

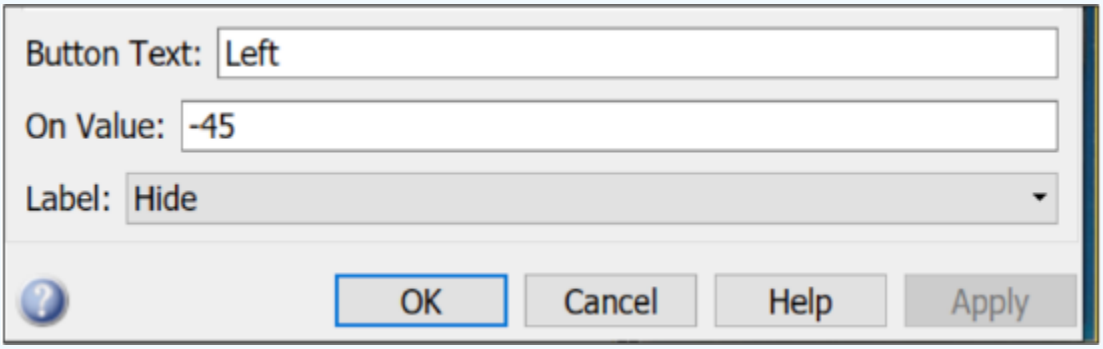


Figure B.68: "Button" block parameter window.

7. Inspect the right **Button** block and compare its "On Value" with the left one. Why is the right one positive?

Referring to these steps above, complete the GUI design for forward and backward motions. *Hint: The **Forward** and **Backward** control input constant blocks are next to the "SpeedPIDController" subsystem, as shown in* Figure B.69. Set the "On" value to 0.2 for forward motion.

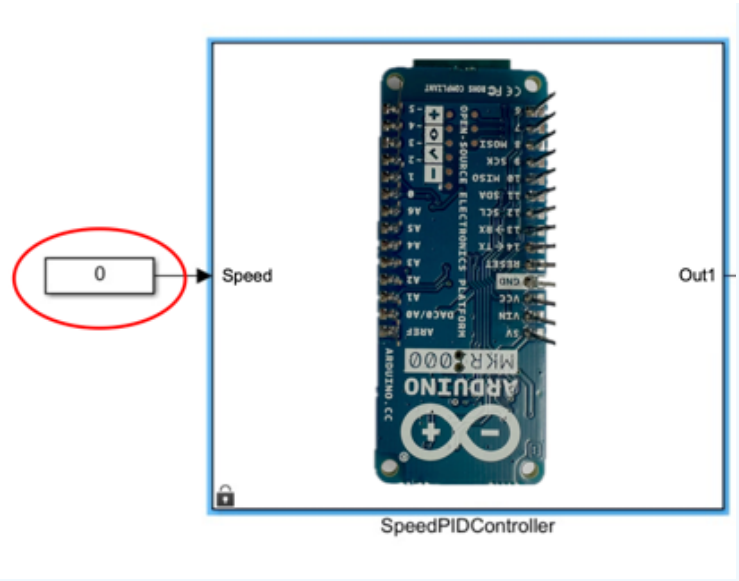


Figure B.69: Forward and backward input control block.

GUI Design for PID Speed Control

The last section of this lab discusses how to develop a GUI for PID speed control. The following steps provide a detailed approach.

1. Right click on the **SpeedPIDController** block, choose “Mask”, and then select “Edit Mask”.
2. Inside the dialog box, select the “Parameters%Dialog” tab.
3. Single click on the row of “Parameters” in the “Dialog” box to highlight it as shown in Figure B.70.
4. Single click “Slider” in the “Control” column” of the “Mask” editor (see the square-dashed box in Figure B.70). The #1 “Slider” will be added (see the rectangular-dashed box in Figure B.70).

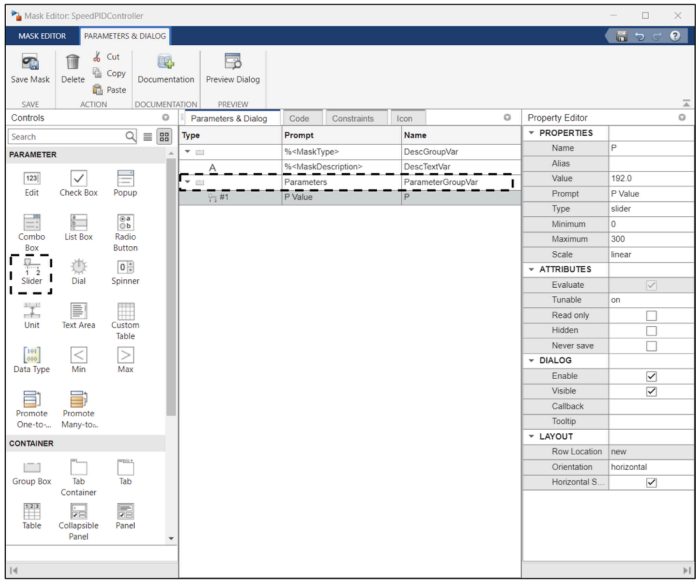


Figure B.70: Mask editor of “SpeedPIDController” subsystem.

5. Edit parameters for the #1 slider as shown in Figure B.71, including the parameters for “Prompt,” “Name,” “Minimum,” and “Maximum.” Then click “Apply.”

Parameters & Dialog | Code | Constraints | Icon

Type	Prompt	Name
	%<MaskType>	DescGroupVar
A	%<MaskDescription>	DescTextVar
	Parameters	ParameterGroupVar
#1	P Value	P

Property Editor

PROPERTIES	
Name	P
Alias	
Value	192.0
Prompt	P Value
Type	slider
Minimum	0
Maximum	300
Scale	linear

Figure B.71: Slider parameter window.

6. After the configuration is applied, double-click on “SpeedPIDController.” The calibrated dialog with “Slider” will appear, as shown in Figure B.72.

Referring to the steps above, complete the GUI design for both “**I Value”** and **“D Value”** of the PID controller. The range of “**I Value**” should be between 0 and 30; the range for “**D Value**” should be between 0 and 2. Use capital letter **I** and **D** for the two slider names.

After completing all three sections, connect your mini-Segway with either the lab Surface computer or your own laptop with the cable provided. Run the model. Try to tune your PID controller to have a stable speed response.

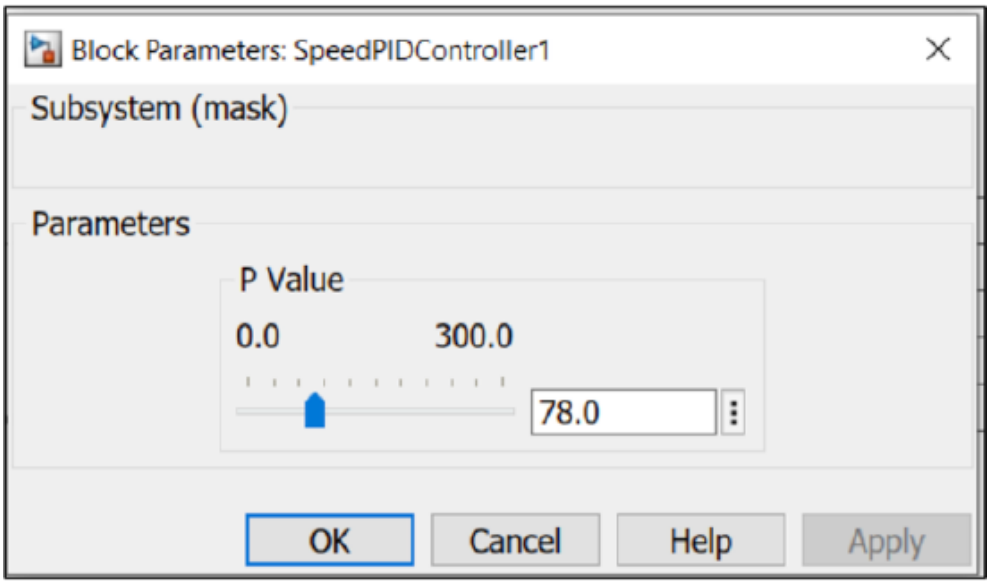


Figure B.72: PID controller tuning dialog.

This page titled 11.5: Simulink GUI (Graphic User Interface) Design and Programming is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11.6: Sensor Fusion vs. Single Sensor

Lab B.6 Objectives

After completing this lab section, you will be able to:

1. Utilize MATLAB State Flow software.
2. Explain the sensor fusion technology used for calculating mini-Segway pitch angle.

State Flow and Initialization

For the mini-Segway, an MPU 6050 sensor board is connected to the I2C pins of the Arduino MKR 1000 Wi-Fi board. The MPU 6050 sensor set is a 6-DOF (degree of freedom) inertial measurement unit used to detect sensor acceleration and angular velocity in all three dimensions. It incorporates both a three-axis gyroscope and a three-axis accelerometer. Its measured signals can be accessed through the I2C communication bus. We can read the sensor data in MATLAB/Simulink by proper configuration of the MPU 6050.

In addition, the MPU 6050 needs to be initialized once powering up to have repeatable operation. For initialization, the I2C address of the slave with the write command needs to be sent, followed by the internal register number to write to. Note that MATLAB/Simulink provides the I2C **Master Write** block capable of writing data to an I2C slave device or a device register as shown in Figure B.73. A slave device is a passive component in a communication system that operates under the direction of a single master device (e.g., Arduino MKR 1000). The slave device address of MPU 6050 can be found either from its data sheet or by typing the command as shown in Figure B.74 in MATLAB after the host computer is connected to the MPU 6050 through mini-Segway control hardware.

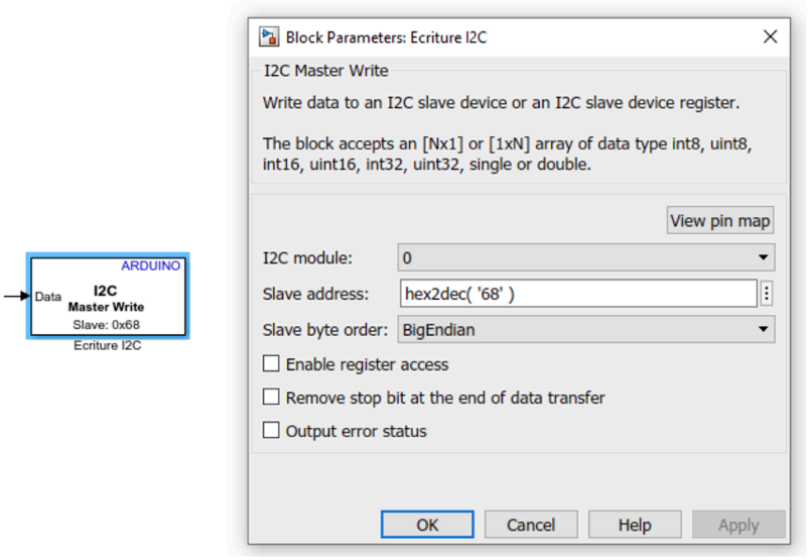


Figure B.73: Simulink I2C write block and its parameter window.

```
>> a=arduino;
>> a

a =

  arduino with properties:

                Port: 'COM5'
               Board: 'MKR1000'
      AvailablePins: {'D0-D14', 'A0-A6'}
  AvailableDigitalPins: {'D0-D14', 'A0-A6'}
     AvailablePWMPins: {'D0-D8', 'D10', 'A3-A4'}
   AvailableAnalogPins: {'A0-A6'}
    AvailableI2CBusIDs: [0]
          Libraries: {'Arduino/MKRMotorCarrier', 'I2C', 'RotaryEncoder', 'SPI', 'Servo'}

>> imu=mpu6050(a)

imu =

  mpu6050 with properties:

                I2CAddress: 104      ("0x68")
                    SCLPin: "D12"
                    SDAPin: "D11"

            AvailableMeasurements: ["Acceleration", "AngularVelocity"]
```

Figure B.74: Detecting MPU 6050 I2C address.

The MPU 6050 provides a programmable full-scale range of ± 2, ± 4, ± 8, and ± 16 g for acceleration measurements and full-scale range of ± 250, ± 500, ± 1000, and ± 2000 degree/second for x-, y-, and z-axis angular rate sensors for these rate gyros. During the initialization process, the ranges of ± 4 g and ± 1000 degree/second, respectively, were selected by writing commands to these associated registers, as shown in Figure B.75.

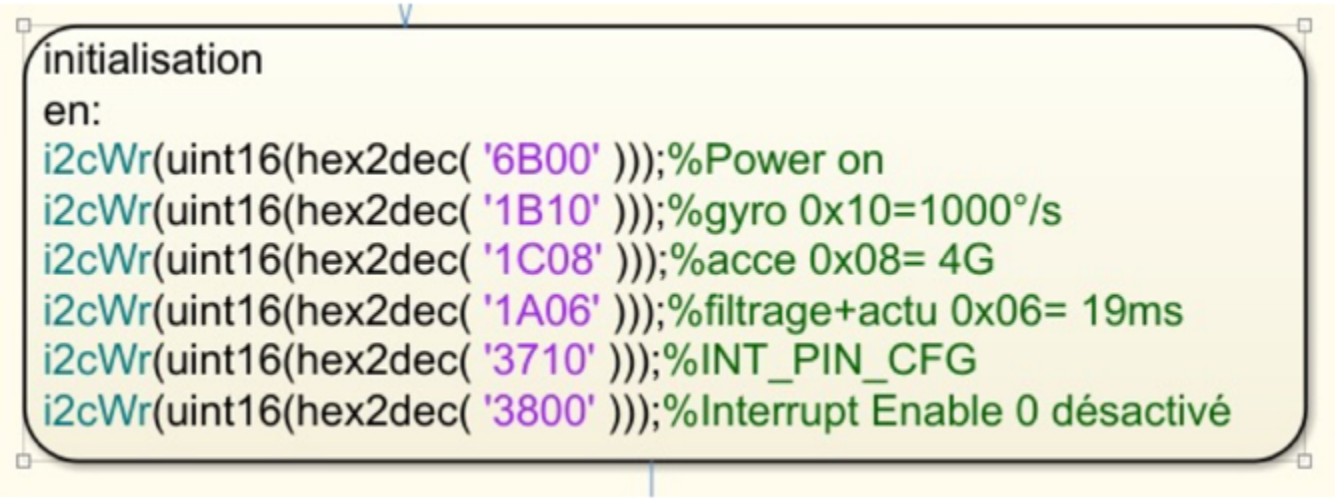


Figure B.75: MPU 6050 initialization.

The specific register to read and write is in the data sheets. However, once the MPU 6050 is configured and running, you can read the associated data. The gyro data starts at register 0x3b.

This model uses a State Flow chart, a software package extending MATLAB and Simulink for modeling logic and event-driven systems. When active, the contents of the selected state are executed sequentially. The entry action is executed immediately upon entering this state. After the entry phase, and for as long as this state is active, the commands in the “During” section of the State Flow model in Figure B.76 are executed. The code for “During” can be thought as the body of a “for” loop in software. Then, as the state becomes inactive, the exit commands are executed. The Simulink function is called from within State Flow charts. Note that a Simulink function is a Simulink subsystem defined to write and read data for MPU 6050 in the mini-Segway Simulink model.

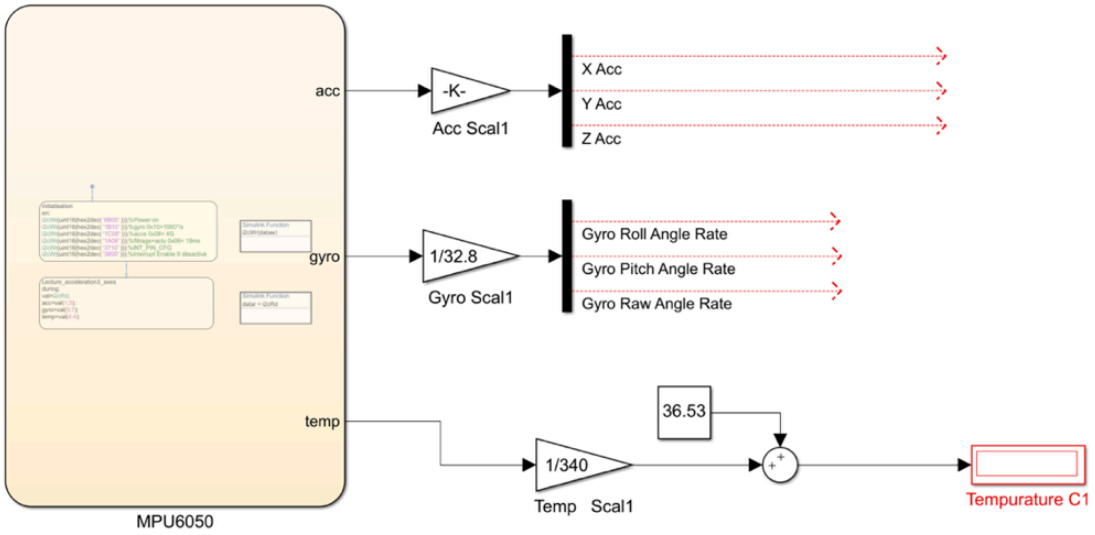


Figure B.76: Calculation based on scaling factor.

The developed MPU 6050 Simulink model is shown in Figure B.76. One can study the model based on the descriptions above. Please do not change any parameters in this block. In addition, the register map and data sheet of MPU 6050 can be found in reference [8] of Chapter 6.

With model outputs defined, from the datasheet, the sensitivity scaling factor can be found for the gyroscope, the accelerometer, and the sensor temperature based on the selected full sensing range. An example for the gyroscope scaling factor is shown in Figure B.77.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	

Figure B.77: Gyro sensitivity scaling factor.

Exercise B6.1

Describe the selected scaling factor of an accelerometer (full scaling factor of ± 4 g) in the short form.

Acceleration and angular velocity can be calculated by dividing sensor raw data by their associated sensitivity scaling factor. One example for rate gyro is shown in Figure B.76. A Simulink **Demux** block was used to split signals to associated three axes, respectively.

As introduced in Section 6.5, mini-Segway pitch angle can be calculated from either accelerometer signals (a_x and a_z) or the gyro signal (ω_g). Using the accelerometer signals in x and z directions, the mini-Segway pitch angle can be calculated by

$$\theta_a = \text{atan}\left(\frac{a_x}{a_z}\right) \tag{11.6.1}$$

Using the gyro signal, the angle can be calculated from angular rate by a discrete integration transfer function, as below.

$$\theta_g = \frac{T_s z}{z-1}\omega_g \tag{11.6.2}$$

where T_s is the sample period.

Exercise B6.2

Connect the rate gyro outputs to the scope and rotate the mini-Segway along each axis manually so that you can associate the sensor output with those of mini-Segway. Describe your findings about roll, pitch, and yaw angles in the report.The direction of the three-orthogonal coordinate system is shown in Figure B.78. When you rotate or move your mini-Segway, keep the arrow of the x-axis point to the forward position.

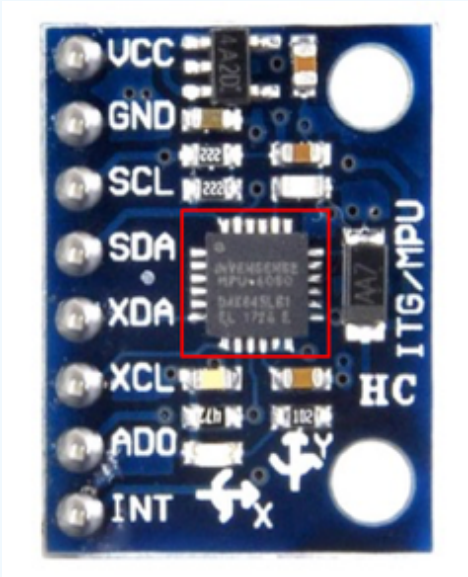


Figure B.78: Definition of three orthogonal axes.

The reference model for completing Exercise B6.2 is shown in Figure B.79.

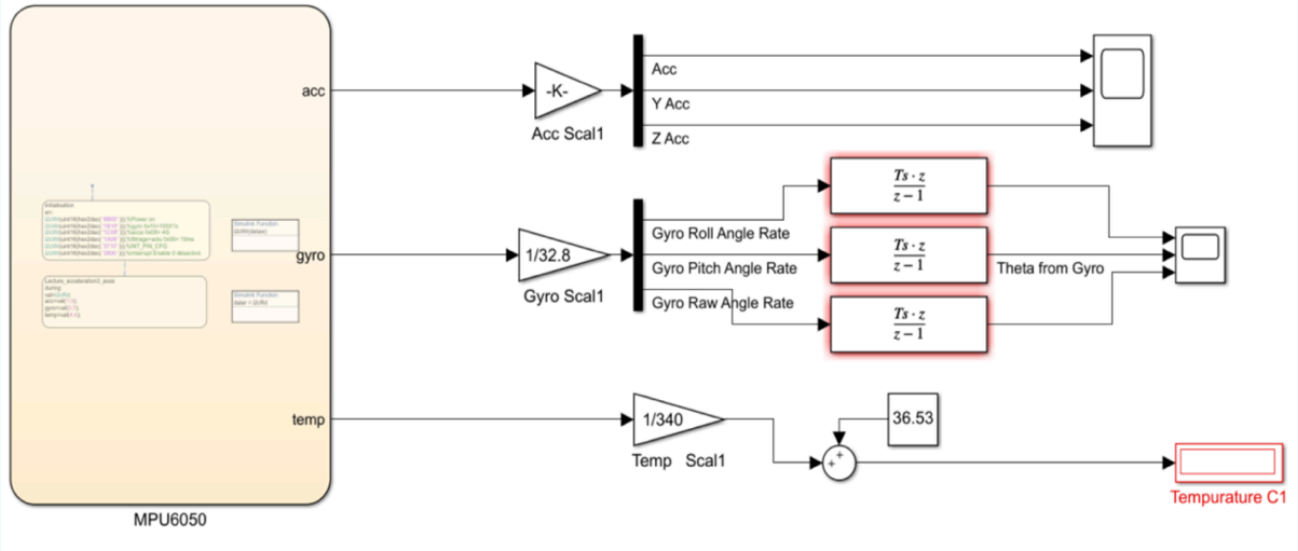


Figure B.79: Reference model for Exercise B6.2.

Sensor Fusion for Calculating Pitch Angle

The pitch angle can be calculated using the formulae in equations (B.11) and (B.12). However, the pitch angle calculated based upon accelerometer signals is accurate only at low frequency because the calculated pitch angle becomes very noisy at high frequency. The pitch angle based on the gyroscope signal is accurate only at high frequency. This is because the integration operation leads to the calculated value drift at low frequency due to rate gyro signal bias. In this lab section, sensor fusion technology will be introduced in Simulink by the seven steps below. For this sensor fusion application, we will adopt the complementary filter technique (see Section 6.5). The basic idea behind this technology is to combine the two calculated outputs from both the accelerometer signals and the gyroscope signal, utilizing the high- and low-pass filters, to obtain a good estimate of the mini-Segway pitch angle.

1. Choose **Trigonometric Function** from the Simulink "Math" library and select **atan2** for function in the parameter dialog window.
2. Connect this block to **X acc** and **Z acc**, resulting in the calculated pitch angle based on accelerometer signals.
3. Add a gain block to convert the unit of output, obtained in Step 2, from radian to degree so that the unit is consistent with the output from gyro.
4. Run the model in the external mode with infinite time. Describe the difference of pitch angle from two different sensors.
5. Stop the simulation.
6. Add a complementary filter into the model to verify the sensor fusion technology described in Section 6.5.
7. Run the model with a complementary filter. Compare the pitch angle signal before and after sensor fusion.

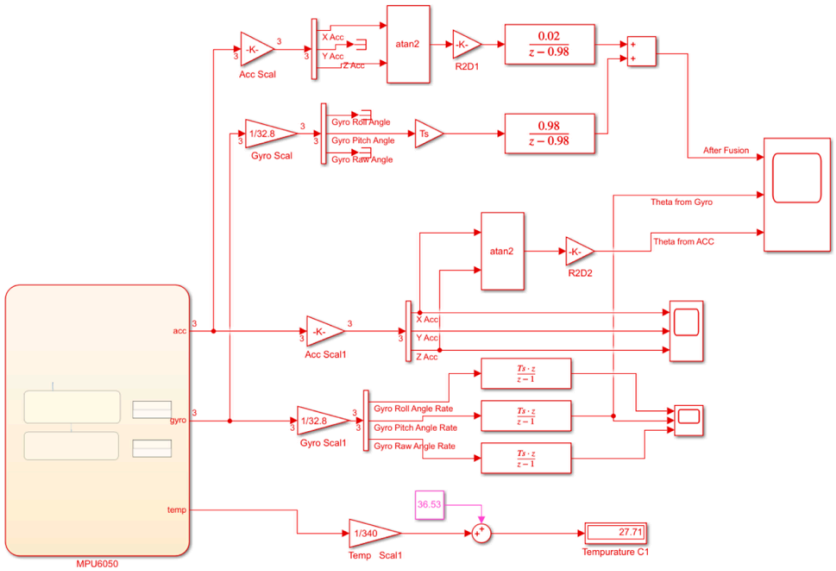


Figure B.80: Reference model for sensor fusion.

Exercise 11.6.1

Show your model and results in the lab report. Figures B.80 and B.81 show the reference model and results, respectively.

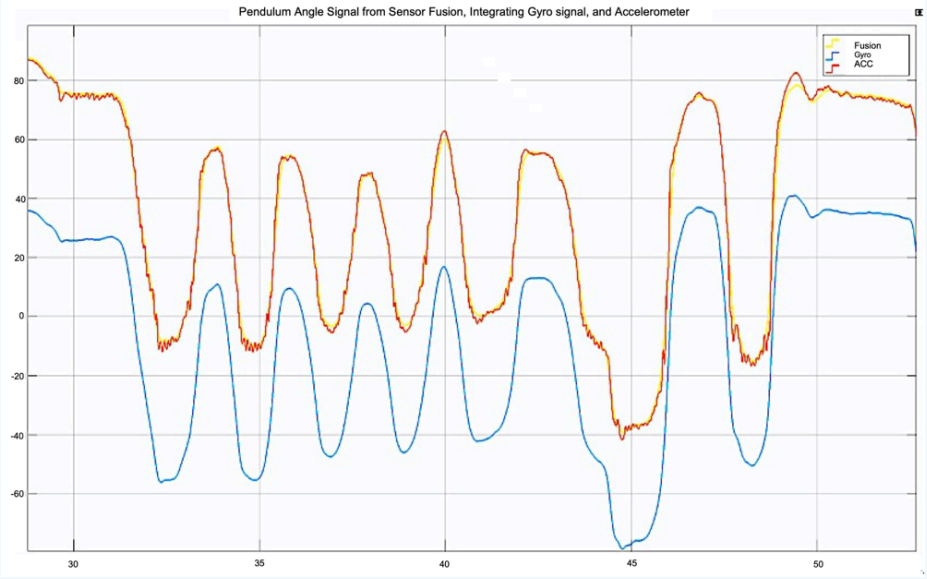


Figure B.81: Pitch angle calculated from both sensor fusion and gyro signal.

This page titled 11.6: Sensor Fusion vs. Single Sensor is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11.7: PID Tuning for Mini-Segway Simulink Model

Lab B.7 Objectives

After completing this section, you will be able to:

1. Create a PID controller in Simulink.
2. Tune the pitch angle and cart speed PID controller gains.

Creating Customized PID Controller

This lab session looks at the mini-Segway Simulink model discussed in Section 8.1. We want to complete tuning gains of both stabilization (pitch angle) and driving (cart speed) PID controllers for the mini-Segway model shown in Figure B.82 as **SEGWAY PLANT**.

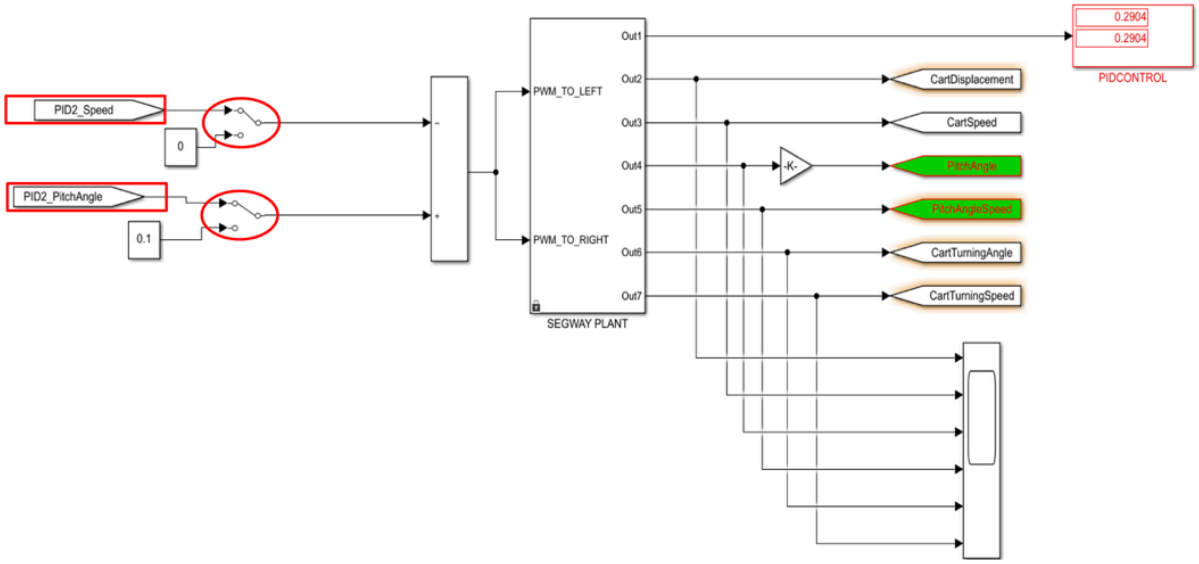


Figure B.82: Simulink model of the mini-Segway.

This linear model provides seven output ports. They are the control input signal, cart displacement (CarDisplacement), cart speed (CarSpeed), pitch angle (PitchAngle), pitch angular speed (PitchAngleSpeed), cart turning angle (CartTuningAngle), and cart turning speed (CartTuningSpeed). The pitch angle and cart speed are considered as system outputs for feedback control purpose. The control inputs (armature voltages of left u_{aL} and right u_{aR} motors) are realized through PWM control (see Section 5.3), where percentages of duty-cycle are actual control inputs.

As discussed in Section 3.4, PID control can deal only with single-input and single-output (SISO) systems. For the mini-Segway system, two PID controllers are used to control mini-Segway pitch angle and cart speed, respectively. A closed-loop PID control system block diagram (see Figure B.83) needs to be designed in Simulink. Here, the output of the PID controller is fed to the mini-Segway system plant as input. The PID controller input is the error between the reference and actual plant output (reference pitch and actual angles).

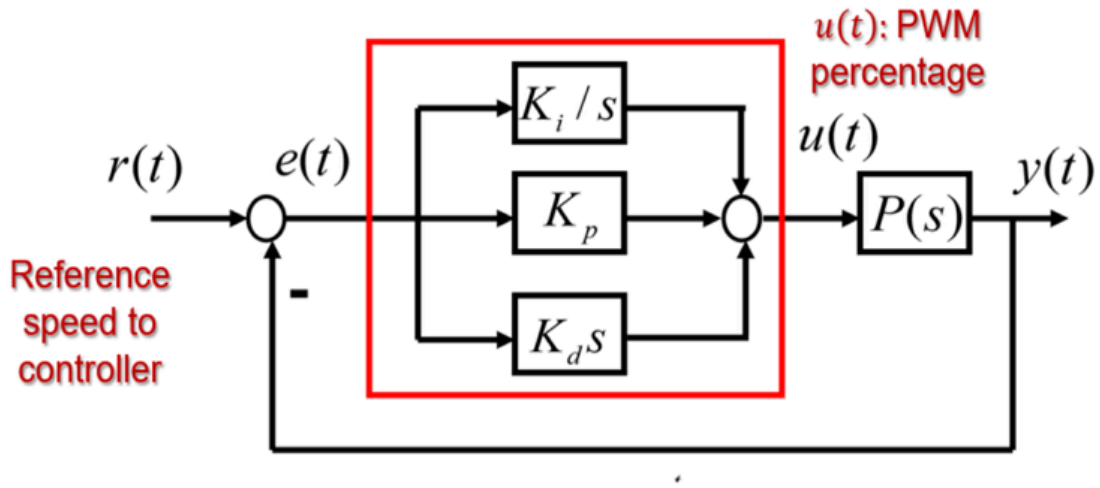


Figure B.83: A closed-loop PID control system.

Exercise B7.1

Refer to the following steps to create two customized PID controllers for pitch angle and cart speed control, respectively. Present both in the lab report.

1. Select the correct Simulink blocks from the corresponding Simulink library to realize the PID controller with its time-domain dynamic equation (B.13). For practical applications, a low-pass filter in the form of $a/(s+a)$ needs to be added ahead of the derivative operation. This reduces its sensitivity to high frequency noise in the error signal $e(t)$ (*Hint: You will need the* ***Sum***, ***Constant***, ***Product***, ***Integrator***, *and* ***Derivative*** *blocks.*)

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{11.7.1}$$

2. The error input of the pitch angle PID controller is the negative pitch angle output of the mini-Segway plant. Because the reference input is zero, pitch angle PID control is to keep the mini-Segway inverted pendulum standing vertically at its neutral position. Once the pitch angle is deviated from zero, both DC motors are energized to turn both wheels in the same direction to pull the mini-Segway pendulum back. This is the pitch angle stabilizing process of the mini-Segway.
3. Recall the mini-Segway hardware protection process adopted for both motors and their Motor carrier board in Lab B.4. For each PID controller, a **Saturation** block is used to limit the control output magnitude to be within ± 255 counts.
4. Connect the pitch angle error to the input port of the PID controller.
5. Connect the PID controller output to a **Goto** block. Its name should be consistent with what is shown in the red box of Figure B.83.
6. Figure B.84 shows the reference PID controller model. However, a low-pass filter needs to be added after the block $\Delta u/\Delta t$ highlighted by the red box (see Step 1). Creating a subsystem makes the PID controller model look simple and clean. (*Hint: For the pitch angle controller,* a *can be set to* 200).

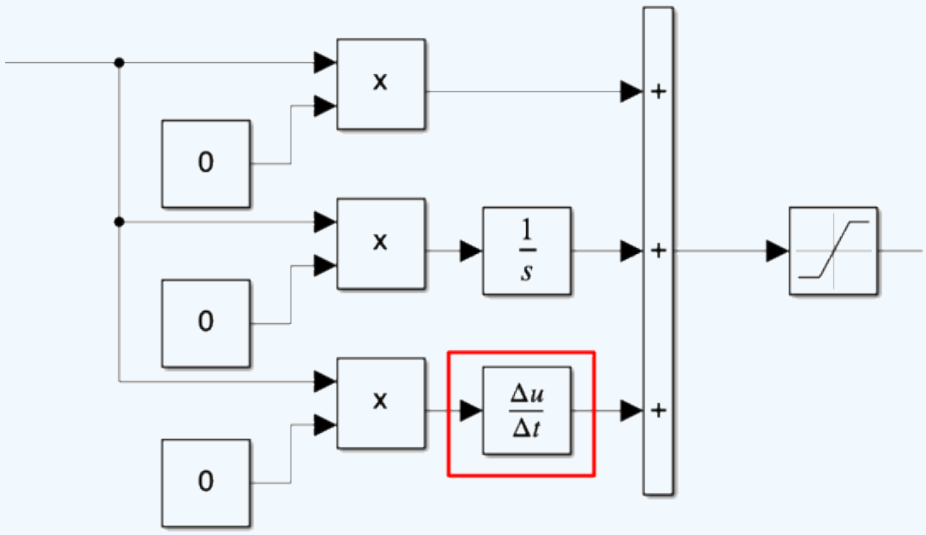


Figure B.84: Reference PID controller model for pitch angle control.

7. Use the same logic to build the cart speed PID controller with $a = 100$ for the low-pass filter; the cart speed has much slower dynamics than that of pitch angle. Use the step function (**Step** block in Simulink) as the input reference cart speed. The associated parameter configuration window is shown in Figure B.85.

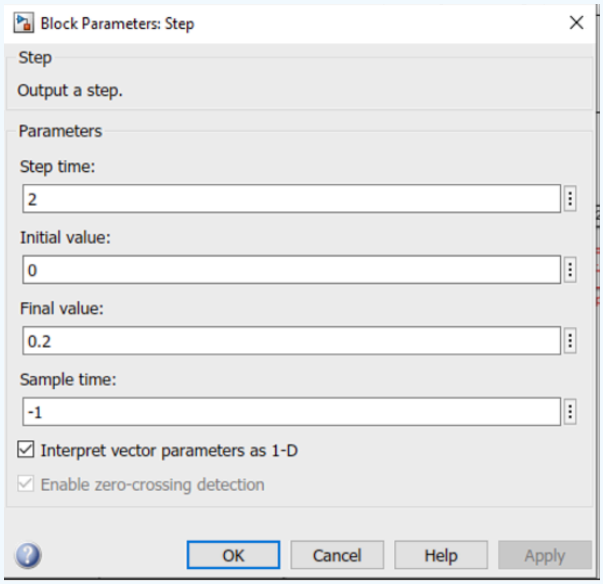


Figure B.85: Step function parameter window.

8. Before calculating the error between the plant output (cart speed) and the reference input, we must add a low-pass filter with $a = 5$ in order to filter out the high frequency noise of cart speed signal. The error input of controller is calculated as shown in Figure B.86.

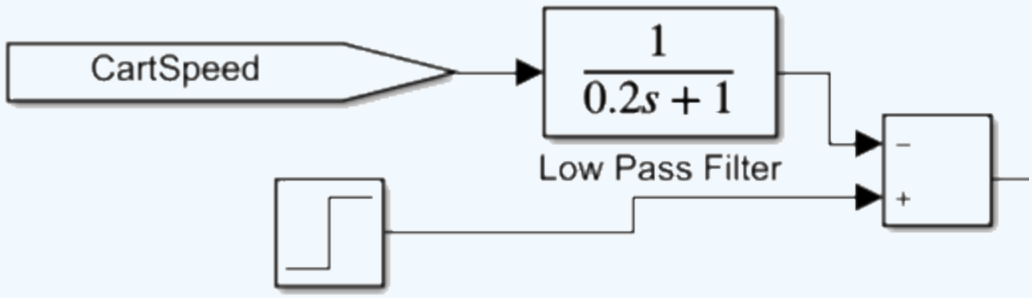


Figure B.86: Cart speed error input calculation.

PID Control-Gain Tuning for Pitch Angle Regulation

The pitch angle PID control gains need to be tuned so that the mini-Segway pendulum is stabilized at its upright position. The pitch angle control-gain tuning process follows steps below.

1. Double-click the **Manual Switch** blocks as shown in any red circle in Figure B.82. The switch will then move when you double-click it. Configure the manual switch as shown in Figure B.87 so that the pitch angle PID control gains can be tuned.

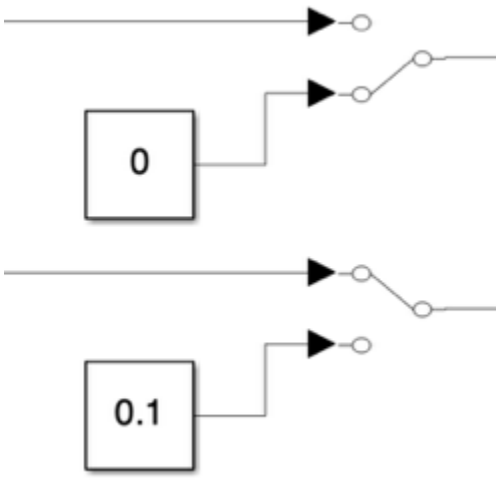


Figure B.87: Manual switch of pitch angle controller.

2. Tune the **P** gain firstly and keep both the **I** and **D** gains at zero. If the **P** gain is too small, the pitch angle cannot be stabilized because the motor output torque is too small to stabilize. Start from the **P** gain of 6; increase it until the pitch angle becomes stable. What do you find when you increase the **P** gain? (*Hint: compare the maximum amplitude and oscillation of control signal with a different **P** gain*). Note that the **P** gain needs to be small but large enough to stabilize the pitch angle motion.
3. As discussed in Chapter 8, the pitch angle and mini-Segway speed dynamics are NOT independent. To avoid double integration, the integration (**I**) gain of the pitch angle PID controller must be set to zero due to its fast dynamics.
4. Although the pitch angle becomes stable as the **P** gain increases, it leads to an increased number of oscillations before it is stabilized. Tuning the **D** gain can reduce the oscillation level. Start from a **D** gain of 0.03; increase it till that the pitch angle can be stabilized within 0.1 second.

Exercise B7.2

Document the pitch angle PID control-gain tuning process and the associated resulting gains.

PID Controller for Cart Speed Regulation

Cart speed PID control gains can be tuned for smooth and stable cart speed tracking by using the following steps.

1. Double-click the manual switch blocks as shown in the red circle in Figure B.82. The switch will move once you double-click it. Configure the manual switch as shown in Figure B.88 so that the cart speed PID control gains can be tuned under the influence of the pitch angle PID control. Because of the coupling effect of the pitch angle and cart speed motions, tuning the cart speed PID control gains can be challenging.

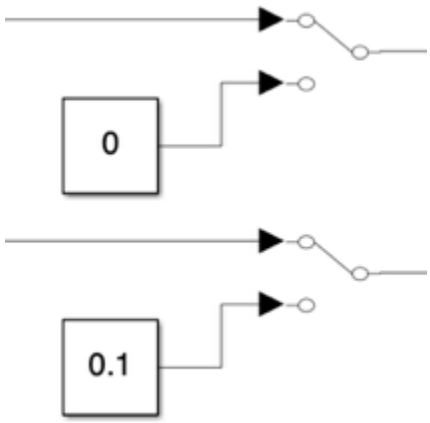


Figure B.88: Manual switch configuration of the dual PID controllers.

2. Referring to the same logic shown in Section B7.2, tune one PID gain and fix the others. Then fix the tuned gain and tune the others using the PID gain tuning method discussed in Section 3.4.
3. For the **P** gain, start from 100 and increase it till the following two goals are achieved. First, the steady-state error is less than 0.1. Second, the rise time (see Section 3.2) is less than 1 second, with the final **P** gain as small as possible.
4. For the **I** gain, start from 8. Then increase it until the steady-state error is less than 0.002.
5. The **D** gain is used to predict the tendency of change. An actual physical system has measurement noise. Because a large **D** gain could lead to instability, select a small **D** gain for this model. In addition, to help pitch angle stabilization (as discussed early in Section 8.5), a negative derivative gain helps improve performance.

Exercise B7.3

Document both the cart speed PID control-gain tuning process and the resulting gains.

This page titled 11.7: PID Tuning for Mini-Segway Simulink Model is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11.8: PID Tuning for Mini-Segway Hardware System

Lab B.8 Objectives

After completing this lab section, you will be able to:

1. Tune the pitch angle PID controller gains for the mini-Segway hardware.
2. Tune the cart speed PID controller gains for the mini-Segway hardware.

This lab session provides the real-time Simulink mini-Segway control model so that the PID control gains can be tuned with the mini-Segway hardware in feedback loops. The locations of the parameters for two PID controllers in the real-time Simulink control model are marked with two red boxes in Figure B.89. These two PID controller parameters were tuned for the mathematical Simulink model in Lab B.7. Now they can be used as the initial gains for the PID gain tuning process for the mini-Segway hardware.

Before beginning the tuning process, visually inspect your mini-Segway to check if the following hardware connections and battery location are correct:

1. All pins and cables of the boards are connected properly without any loose or distort connections.
2. The battery is connected to the Arduino through the interconnector of the mini-Segway.
3. The battery was placed close to the mass center of the mini-Segway without obvious deflection.

Each time, when you tune the parameters, you need to complete the following two steps:

1. Compile and upload the Simulink model software to the mini-Segway hardware (MKR 1000 Wi-Fi board) in the Wi-Fi mode. Select the same router for the host computer with the USB cable connected from the host computer to the mini-Segway. While compiling and uploading the software, the mini-Segway should be laid down to ensure that the wheels not spinning during the process (the mini-Segway will be active when pitch angle is less than 45 degrees).
2. After successfully uploading the Simulink model into the mini-Segway, disconnect the USB cable. Then lift and hold the mini-Segway in the vertical position, with zero pitch angle, to activate software.

Pitch Angle PID Controller Gain Tuning

Use the following steps to tune pitch angle PID gains (**P** and **D** gains). The **I** gain of the pitch angle PID controller is set to zero to avoid integration value windup.

1. At this point, when tuning the pitch angle PID controller gains, all cart speed PID gains are set to zero.
2. Set both **I** and **D** gains to zero and tune the **P** gain first. Use the tuned **P** gain based upon the mini-Segway model from Lab B.7 as the initial **P** gain.
   a. If the mini-Segway is unstable with large oscillations, reduce the **P** gain.
   b. If the mini-Segway inclines to the forward (or backward) direction and falls down directly, the **P** gain is too small. Thus, increase the **P** gain.

Tune the **P** gain until the mini-Segway has a backward tendency–before the mini-Segway begins to move forward.

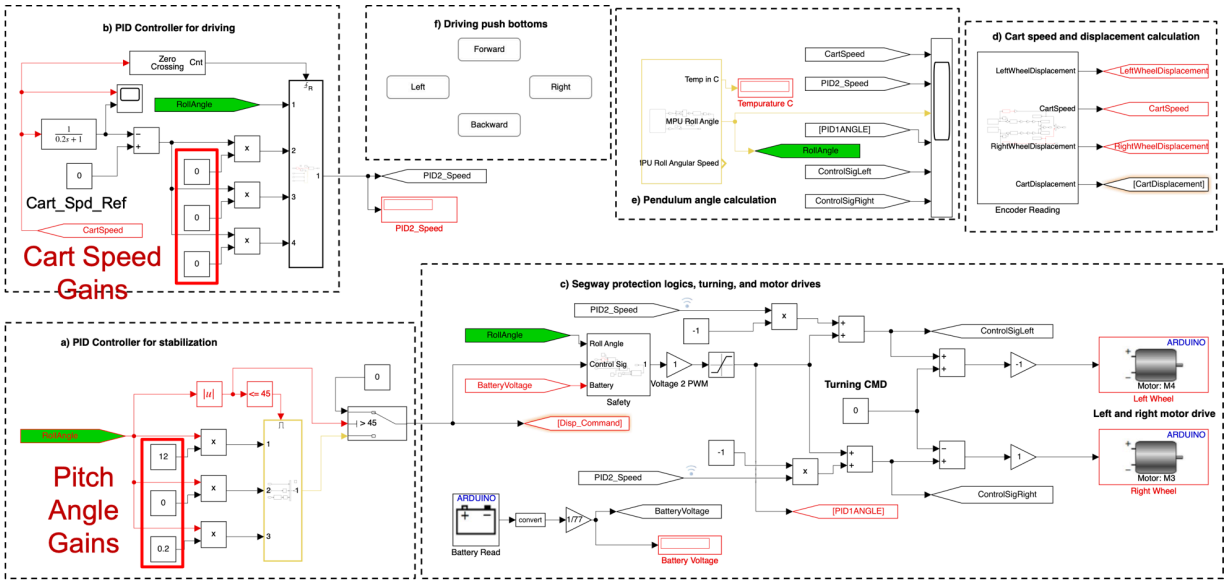


Figure B.89: Real-time Simulink control model for the mini-Segway.

1. Set the **I** gain to 1.5 temporarily and keep it unchanged until the end of this section. From Lab B.7, the integration gain of the pitch angle PID controller is set to zero so that the **I** gain is used only for the cart speed control. However, to make the **D** gain tuning easy in the next step, temporarily set the **I** gain to a nonzero value.
2. Tune the **D** gain.
3. If the mini-Segway still moves forward, the **D** gain is still too small; therefore, increase it.
4. If the mini-Segway becomes unstable with large oscillations, the **D** gain is too large; therefore, decrease it.
5. Tune the **D** gain until the mini-Segway can stand stably.
6. Reset **I** gain back to zero.

Cart Speed PID Controller Gain Tuning

Use the following steps to tune cart speed PID gains (**P**, **I**, and **D** gains).

1. The reference speed inputs from the push buttons “Forward” and “Backward” need to be set to 0.1 and -0.1 (m/s).
2. Tune the **P** gain first
   a. Press the forward button. If the cart does not move, the **P** gain is too small, and increase it.
   b. Press the forward button. If the cart moves too fast, the **P** gain is too large, and decrease it.
   c. Find a **P** gain so that the cart can move forward steadily.
   d. Try the backward button.
3. Tune the **I** gain

The **I** gain for the cart motion control will also affect the performance of pitch angle motion. Choose an “I” gain so that the mini-Segway can stand and move stably.

1. Tune the **D** gain
2. The **D** gain will not affect the performance much when you press the forward button. However, releasing the button will affect the stability of the mini-Segway.
3. If the **D** gain is too large, the mini-Segway will also oscillate significantly.
4. Consider choosing a small **D** gain while maintaining stability. Also, try to use a negative **D** gain for improved pitch angle stability during the beginning and ending of driving.

After you finish tuning the PID gains for both pitch angle and cart speed PID controllers, try to make your model as clean and simple as you can so that you can always find any block interested in the model. Because both pitch angle and cart speed motions are coupled, tune both controller gains iteratively for improved performance.

Exercise B8.1

In your report, document the tuning processes for both pitch angle and cart speed PID control-gain tuning processes, along with the suggested iterative tuning process. In addition, present the tuned gains for both PID controllers as well as the associated time responses in the report.

Exercise B8.2 (Bonus Task)

These tuned PID gains can be affected by the ground surface condition (or friction coefficient between wheel and ground) and the set point of cart speed from the “Forward” and “Backward” push buttons. After these PID controller gains are successfully tuned, you can increase the reference speed input from 0.1 (m/s) to 0.2 (m/s). Find out if both PID controller gains need to be retuned so that the mini-Segway moves stably. In addition, during Lab B.9 show whether these PID gains need to be retuned for different floors (carpet, hard wood, plastic, etc.)

This page titled 11.8: PID Tuning for Mini-Segway Hardware System is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

11.9: Mini-Segway Stabilization and Drive Project

Lab B.9 Objectives

After completing this lab section, you will be able to:

1. Retune control gains of both PID controllers under a real-world driving condition.
2. Drive through the final project course.

The final project is to drive the mini-Segway through the testing track (3 by 8 feet) shown in Figure B.90 with the shortest time possible. The red bars represent obstacles that the mini-Segway cannot go through. The mini-Segway needs to navigate the entire course by entering from the right side and exiting from the left side while keeping its upright position (no falling down). A ramp can also be positioned as indicated by the red-dashed box. With a peak height of 2.5 in, the ramp can be placed so that the mini-Segway can go through the three right bars without make any turn. However, going through the ramp is challenging because more driving force is required to go up the ramp and reduced force to go down the ramp. In this case, certain feedforward control logic may be needed.

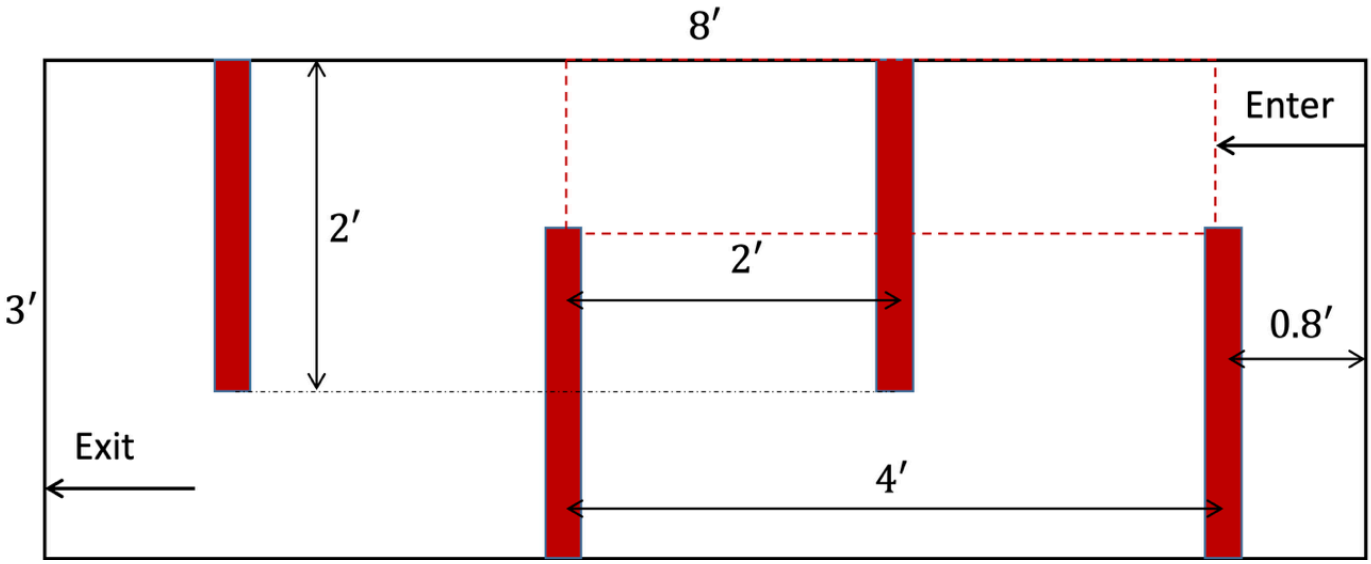


Figure B.90: Final project driving course layout.

Let's first discuss how to drive through the regular course. Before driving, both PID gains for pitch angle and car speed controllers need to be retuned, aimed to improve the controllability while driving through the entire course. For example, the reference speed magnitudes for forward and backward driving, left and right tuning, need to be carefully adjusted so that the driver feels comfortable operating the mini-Segway. After these reference speed signals are calibrated, the PID gains for pitch angle and car speed controllers need to be retuned.

Large reference speeds may require increased derivative gain and proportional gain for the pitch angle PID controller to stabilize the mini-Segway, especially when starting and stopping it. On the other hand, extremely large gains could also destabilize the mini-Segway. Therefore, we must find a sweet point for the tuned gains.

Figure B.91 illustrates a strategy for driving through the course that would minimize the number of turns required to complete the course. In this case, only four turns are required. Note that these push button calibrations make it impossible to turn when the mini-Segway moves forward or backward.

The other method for completing the course quickly is to develop a Simulink program that get the mini-Segway through the course autonomously. This method uses a logic similar to the one described in Section 9.5 for the three wheel-cart. Even with manual driving, the mini-Segway can complete the course within 10 seconds. The key for completing the course quickly is to avoid

any overshoot at any turning point so that the mini-Segway does not need to be back out.

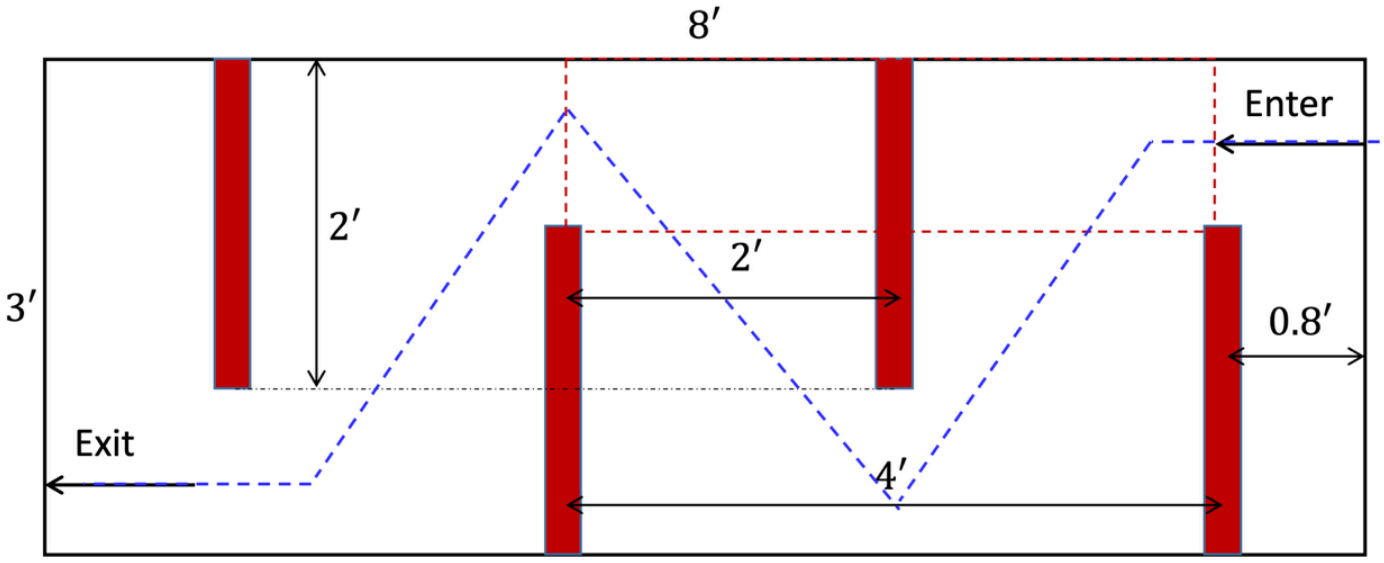


Figure B.91: Final project driving over the regular course.

Consider the case of driving through the course with the ramp (represented by the red-dashed box) with a trajectory represented in the blue dot-dashed line in Figure B.92. The challenge is stabilizing the pitch angle when the mini-Segway climbs the ramp and then moves down the ramp. We also need to factor in the associated driving torque required to climb and drive down the ramp. We can try several options. One uses large gains, especially derivative gain, for the pitch angle PID controller. A second considers the feedforward control used for climbing and moving down the ramp with different push buttons. Last, there is the option of autonomous driving.

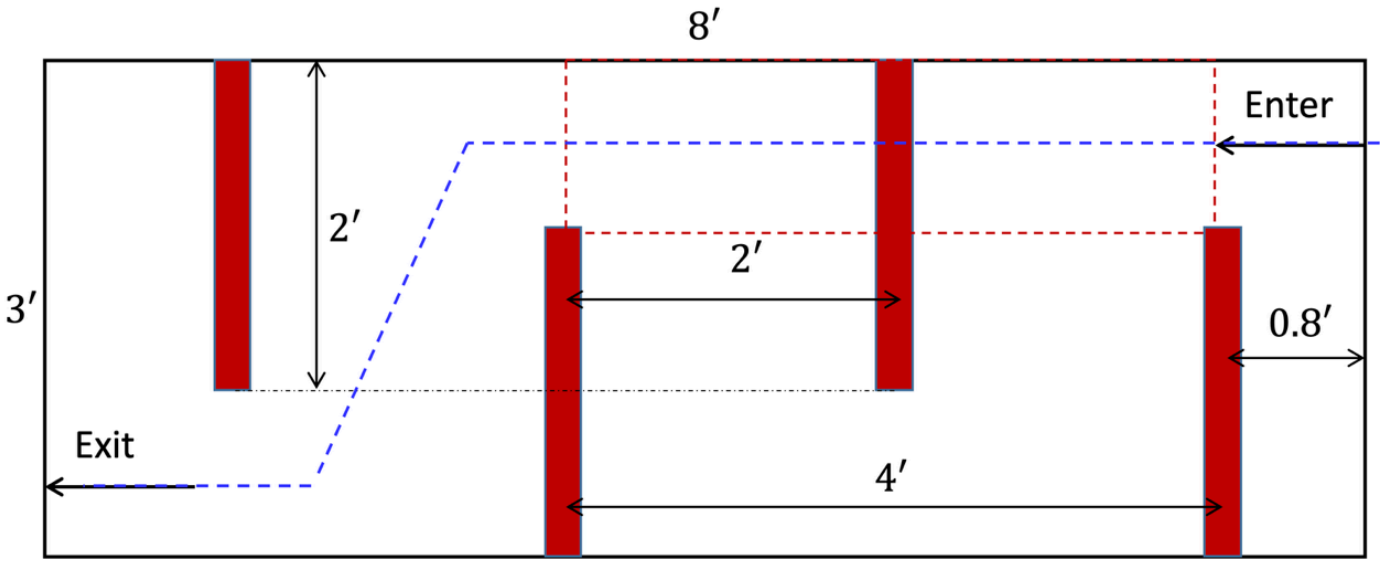


Figure B.92: Final project driving over the course with a ramp.

Exercise B9.1

The project report for this lab session should include an introduction section to describe the mini-Segway system (both hardware and software); the mini-Segway stabilization (pitch angle) and driving (cart speed) controller gain tuning process for the target driving course, along with associated time responses; and discussions about driving through the project course with its trajectory and time, along with adjustment to reduce the driving time. The report should also have a summary describing what has been learned from this project.

This page titled 11.9: Mini-Segway Stabilization and Drive Project is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

INDEX

A

A/D conversion
- 11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)
- 4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)
- 6.0: Prelude

A/D converter
- 6.2: Rotational Speed Sensors and Their Speed Calculation

AC motor
- 5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control

accelerometer
- 11.6: Sensor Fusion vs. Single Sensor

accuracy
- 6.5: Sensor Fusion

actuation
- 5: Mechatronic System Component- Actuators
- 5.4: Discrete Electromagnetic Valve Actuation (EMVA)

actuation subsystems
- 5.0: Prelude

actuator
- 5.5: Discrete Electrohydraulic Valve Actuation (EHVA)

actuator design
- 1.4: Chapter References

actuator selection
- 7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process

actuators
- 1.1: Introduction of Mechatronic Systems
- 5: Mechatronic System Component- Actuators
- 8.0: Prelude

adaptive control
- 5.7: Chapter References

adaptive cruise control
- 1.4: Chapter References

aerospace engineering
- 9.7: Chapter References

agricultural spray system
- 6.8: Vision-Based Sensing

aliasing
- 4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters

analog
- 4.1: Analog Signals and Their Frequency Analysis

analog filter design
- 4.7: Advanced Analog and Digital Filter Design and Implementation

analog filtering
- 4.4: Analog and Digital Filter Design

analog sensor
- 6.2: Rotational Speed Sensors and Their Speed Calculation

analog sensors
- 6.0: Prelude

analog signal
- 4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters

analog signals
- 4: Signal Processing
- 4.0: Prelude
- 4.6: Pulse-Width and Frequency Modulated Signals
- 5.0: Prelude

analog system
- 11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)

analog systems
- 11: Appendix B - Mechatronics Lab Sessions

analog to digital conversion
- 4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters

analysis
- 2.2: Mechanical System Modeling
- 7.1: Mechatronic System Architecture

angle calculation
- 9.2: Dual- and Triple-PID Control of a Three-Wheel Cart

angular rate measurement
- 6.4: Angular Velocity Sensors (Rate Gyro)

angular velocity control
- 5.3: DC-Motor Drive Circuits

angular velocity sensors
- 6.4: Angular Velocity Sensors (Rate Gyro)

appendix
- 11: Appendix B - Mechatronics Lab Sessions

applications
- 5: Mechatronic System Component- Actuators

architecture
- 7.2: Microcontrollers Used for Mechatronic Systems

Arduino
- 11.4: DC-Motor Control Utilizing Arduino Hardware
- 7: Mechatronic Control Systems
- 7.0: Prelude
- 7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi
- 7.4: Mechatronic Control System Architecture- Speed Control Example

Arduino IDE
- 7.6: Chapter References

Arduino MKR 1000
- 11.6: Sensor Fusion vs. Single Sensor
- 8.7: Experimental Study of the Mini-Segway Model-Free Control

armature voltage
- 5.2: Mechatronic Actuator Background

automation
- 9.7: Chapter References

automotive control systems
- 1.4: Chapter References

autonomous driving
- 6.8: Vision-Based Sensing
- 9.1: Three-Wheel Cart System Modeling

B

back EMF
- 5.2: Mechatronic Actuator Background

block diagram
- 2.4: Mechatronic System Modeling

Bode
- 4.1: Analog Signals and Their Frequency Analysis

Bode plot
- 3.1: Impulse, Step, and Sinusoidal Responses

bulk micromachining
- 6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing

butterworth filter
- 4.7: Advanced Analog and Digital Filter Design and Implementation

C

calibration
- 3.7: Chapter References

capacitive MEMS
- 6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing

capacitor
- 2.3: Electrical and Electronic System Modeling

cart speed
- 8.7: Experimental Study of the Mini-Segway Model-Free Control

Chapter 2
- 1.3: How to Use This Book

chebyshev filter
- 4.7: Advanced Analog and Digital Filter Design and Implementation

circuit analysis
- 2.3: Electrical and Electronic System Modeling

circuit design
- 5.3: DC-Motor Drive Circuits

closed loop system performance
- 2.0: Prelude

communication protocols
- 7.2: Microcontrollers Used for Mechatronic Systems

computer systems
- 1: Introduction

constant speed
- 11.4: DC-Motor Control Utilizing Arduino Hardware

continuous time Laplace transforms
- 2: Mechatronic System Modeling

continuous_time_systems
- 8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input

control
- 1.0: Prelude
- 11.9: Mini-Segway Stabilization and Drive Project
- 3: System Responses and Controls
- 5.5: Discrete Electrohydraulic Valve Actuation (EHVA)
- 7.1: Mechatronic System Architecture
- 8.0: Prelude

control architecture
- 8: Mini-Segway System — Mechatronic System Example I
- 8.2: Model-Free Dual-PID Mini-Segway Control System Architecture
- 8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers

control command
5.1: Mechatronic Actuator Background
control design
3.0: Prelude
8.1: Mini-Segway System Modeling
9.3: Model-Based Control of a Three-Wheel Cart
control design model
2.0: Prelude
control methodology
9.0: Prelude
control optimization
1.4: Chapter References
control performance
3.6: Machine Learning-Based Control
8.2: Model-Free Dual-PID Mini-Segway Control System Architecture
8.8: Model-Based Control Experiments
control software
9.5: Experimental Investigation of PID Control Strategies
control strategies
1.2: Introduction of Mechatronic Systems
control strategy
8.3: Model-Based Mini-Segway Control
9.3: Model-Based Control of a Three-Wheel Cart
9.4: Simulation Study of PID and Model-Based Control Strategies
9.6: Experimental Investigation of Model-Based Control Strategy
control system
11.6: Sensor Fusion vs. Single Sensor
11.7: PID Tuning for Mini-Segway Simulink Model
8.3: Model-Based Mini-Segway Control
8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller
9.0: Prelude
control system design
7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process
Control System Toolbox
11.0: MATLAB/Simulink Installation
control systems
1: Introduction
1.1: Introduction of Mechatronic Systems
1.2: Introduction of Mechatronic Systems
1.3: How to Use This Book
10: Appendix A - MATLAB and Simulink Software
11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink
2.1: Laplace Transforms
3.7: Chapter References
4.0: Prelude
5.7: Chapter References
6.9: Chapter References
7.0: Prelude
7.1: Mechatronic System Architecture
7.4: Mechatronic Control System Architecture- Speed Control Example
9.7: Chapter References
control theory
2.4: Mechatronic System Modeling
3.5: Review of Model-Based Control Design and Gain Tuning Methods
4.3: Digital Signals and Their Frequency Analysis
control tuning
8: Mini-Segway System — Mechatronic System Example I
control_design
8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input
controller design
8.3: Model-Based Mini-Segway Control
controller structure
3.4: PID Control and Its Gain Tuning Methods
conversion process
4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)
corner frequency
4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters
course
11.9: Mini-Segway Stabilization and Drive Project
CPU
7.2: Microcontrollers Used for Mechatronic Systems
current feedback
6.1: Mechatronic Sensor Background
cutoff frequency
11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi

D

D/A conversion
11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)
4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)
data analysis
11.6: Sensor Fusion vs. Single Sensor
data clustering
6.9: Chapter References
data visualization
10: Appendix A - MATLAB and Simulink Software
DC
5: Mechatronic System Component- Actuators
DC motor
11.4: DC-Motor Control Utilizing Arduino Hardware
11.5: Simulink GUI (Graphic User Interface) Design and Programming
2.4: Mechatronic System Modeling
4.6: Pulse-Width and Frequency Modulated Signals
5.1: Mechatronic Actuator Background
5.2: Mechatronic Actuator Background
5.3: DC-Motor Drive Circuits
7.0: Prelude
8.1: Mini-Segway System Modeling
9.1: Three-Wheel Cart System Modeling
DC motors
5.0: Prelude
6.1: Mechatronic Sensor Background
derivative
8.2: Model-Free Dual-PID Mini-Segway Control System Architecture
design
7.1: Mechatronic System Architecture
7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process
8.0: Prelude
design process
7.0: Prelude
7.2: Microcontrollers Used for Mechatronic Systems
diesel engine control
9.7: Chapter References
differential equation
2.7: Chapter References
differential equations
2.3: Electrical and Electronic System Modeling
8: Mini-Segway System — Mechatronic System Example I
digital control
7.1: Mechatronic System Architecture
digital design
7.6: Chapter References
digital filter concepts
4.7: Advanced Analog and Digital Filter Design and Implementation
digital filter simulation
11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi
digital filtering
4.4: Analog and Digital Filter Design
digital filters
4.0: Prelude
digital representation
4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)
digital sensor
6.2: Rotational Speed Sensors and Their Speed Calculation
digital sensors
6.0: Prelude
digital signal
4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters
digital signal processing
1.3: How to Use This Book
1.4: Chapter References
4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)
4.3: Digital Signals and Their Frequency Analysis
digital signals
4: Signal Processing
4.0: Prelude
4.3: Digital Signals and Their Frequency Analysis
4.6: Pulse-Width and Frequency Modulated Signals
5.0: Prelude
digital system
11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)
digital systems
11: Appendix B - Mechatronics Lab Sessions
digital twin
7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process
discrete fourier transform
4.3: Digital Signals and Their Frequency Analysis
discrete Fourier transforms
4: Signal Processing
discrete_time_systems
8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input
discretization
11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)

drive circuits
5.3: DC-Motor Drive Circuits
driving electronics
5.0: Prelude
dual PID control
9: Three-Wheel Cart - Mechatronic System Example 2
dynamic characteristics
1.1: Introduction of Mechatronic Systems
dynamic modeling
5.7: Chapter References
dynamic system
2.5: System Linearization
2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning
dynamic system simulation
11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink
dynamic_systems
8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input
dynamics
2.2: Mechanical System Modeling
5.5: Discrete Electrohydraulic Valve Actuation (EHVA)
8.1: Mini-Segway System Modeling
8.3: Model-Based Mini-Segway Control

E

EHVA
5.4: Discrete Electromagnetic Valve Actuation (EMVA)
electric current sensing
6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing
electric electronic subsystems
2.0: Prelude
electric motor
5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control
electric motors
5.2: Mechatronic Actuator Background
electrical
1.0: Prelude
electrical energy
5.1: Mechatronic Actuator Background
electrical engineering
4.0: Prelude
4.6: Pulse-Width and Frequency Modulated Signals
electrical machines
5.7: Chapter References
electrical system modeling
2: Mechatronic System Modeling
electrical systems
2.3: Electrical and Electronic System Modeling
electrohydraulic
5: Mechatronic System Component- Actuators
5.4: Discrete Electromagnetic Valve Actuation (EMVA)
electrohydraulic valves
5.0: Prelude
electromagnetic
5: Mechatronic System Component- Actuators
5.4: Discrete Electromagnetic Valve Actuation (EMVA)
electromagnetic sensing
6.1: Mechatronic Sensor Background
electromechanical valve actuator
5.7: Chapter References
electromechanical valves
5.0: Prelude
electronic
1.0: Prelude
electronic engineering
1: Introduction
electronic systems
2.3: Electrical and Electronic System Modeling
electronics
6: Mechatronic System Component- Sensors
7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi
electropneumatic
5.4: Discrete Electromagnetic Valve Actuation (EMVA)
EMVA
5.4: Discrete Electromagnetic Valve Actuation (EMVA)
encoder reading
11.4: DC-Motor Control Utilizing Arduino Hardware
11.5: Simulink GUI (Graphic User Interface) Design and Programming
engine
5.5: Discrete Electrohydraulic Valve Actuation (EHVA)
engineering
1.0: Prelude
3.7: Chapter References
9.7: Chapter References
engineering applications
2.1: Laplace Transforms
engineering design
1.1: Introduction of Mechatronic Systems
engineering disciplines
1: Introduction
engineering education
1.2: Introduction of Mechatronic Systems
1.3: How to Use This Book
EPVA
5.4: Discrete Electromagnetic Valve Actuation (EMVA)
equations of motion
2.5: System Linearization
8.1: Mini-Segway System Modeling
9: Three-Wheel Cart - Mechatronic System Example 2
error reduction
4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)
error signal
11.7: PID Tuning for Mini-Segway Simulink Model
8.2: Model-Free Dual-PID Mini-Segway Control System Architecture
estimates
6.5: Sensor Fusion
evaluation kits
7.6: Chapter References
evolutionary algorithms
3.7: Chapter References
experimental study
9.5: Experimental Investigation of PID Control Strategies
experimental tuning
9.6: Experimental Investigation of Model-Based Control Strategy
experimental validation
9: Three-Wheel Cart - Mechatronic System Example 2
9.0: Prelude

F

fast Fourier transform
11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)
4.3: Digital Signals and Their Frequency Analysis
fast Fourier transforms
4: Signal Processing
fault detection
6.8: Vision-Based Sensing
feedback
11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink
11.8: PID Tuning for Mini-Segway Hardware System
11.9: Mini-Segway Stabilization and Drive Project
9.4: Simulation Study of PID and Model-Based Control Strategies
feedback control
1.1: Introduction of Mechatronic Systems
11.7: PID Tuning for Mini-Segway Simulink Model
3.5: Review of Model-Based Control Design and Gain Tuning Methods
5.3: DC-Motor Drive Circuits
6.1: Mechatronic Sensor Background
8.7: Experimental Study of the Mini-Segway Model-Free Control
8.8: Model-Based Control Experiments
9.3: Model-Based Control of a Three-Wheel Cart
feedback signals
6.0: Prelude
9.5: Experimental Investigation of PID Control Strategies
feedback_control
8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input
feedforward control
11.4: DC-Motor Control Utilizing Arduino Hardware
FFT
11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)
filter design
4.4: Analog and Digital Filter Design
4.7: Advanced Analog and Digital Filter Design and Implementation
filter design validation
11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi
finite element analysis
2.2: Mechanical System Modeling
first order system
3.3: The First and Second Order System Responses and Associated Pole Locations
first order systems
11: Appendix B - Mechatronics Lab Sessions
3: System Responses and Controls
forces
2.2: Mechanical System Modeling

forward gain
2.4: Mechatronic System Modeling
frequency
4.1: Analog Signals and Their Frequency Analysis
frequency analysis
4: Signal Processing
4.3: Digital Signals and Their Frequency Analysis
frequency component
4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters
frequency domain
2.1: Laplace Transforms
2.5: System Linearization
4.4: Analog and Digital Filter Design
frequency domain analysis
4.7: Advanced Analog and Digital Filter Design and Implementation
frequency response
11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi
3.0: Prelude
3.1: Impulse, Step, and Sinusoidal Responses
4.3: Digital Signals and Their Frequency Analysis
function
4.1: Analog Signals and Their Frequency Analysis
fusion
6.5: Sensor Fusion

G

gain tuning
3: System Responses and Controls
3.0: Prelude
3.4: PID Control and Its Gain Tuning Methods
8.8: Model-Based Control Experiments
gains
11.8: PID Tuning for Mini-Segway Hardware System
genetic algorithm
3.6: Machine Learning-Based Control
genetic algorithms
3.7: Chapter References
graduate classes
1: Introduction
gravity projection angles
9.2: Dual- and Triple-PID Control of a Three-Wheel Cart
GUI design
11.5: Simulink GUI (Graphic User Interface) Design and Programming
gyroscope
11.6: Sensor Fusion vs. Single Sensor
6.4: Angular Velocity Sensors (Rate Gyro)
gyroscopes
6.9: Chapter References

H

hardware
11.8: PID Tuning for Mini-Segway Hardware System
7.1: Mechatronic System Architecture
7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi
hardware architecture
7: Mechatronic Control Systems
7.4: Mechatronic Control System Architecture- Speed Control Example
8.7: Experimental Study of the Mini-Segway Model-Free Control
HVAC control strategy
1.4: Chapter References
hydraulic
5.5: Discrete Electrohydraulic Valve Actuation (EHVA)

I

I2C communication
11.6: Sensor Fusion vs. Single Sensor
IC engine
5.4: Discrete Electromagnetic Valve Actuation (EMVA)
image processing
6.1: Mechatronic Sensor Background
6.8: Vision-Based Sensing
Image Processing Toolbox
11.0: MATLAB/Simulink Installation
imaging
6.9: Chapter References
impulse functions
2: Mechatronic System Modeling
impulse response
3.1: Impulse, Step, and Sinusoidal Responses
induction motor
5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control
inductor
2.3: Electrical and Electronic System Modeling
inertial sensor
6.4: Angular Velocity Sensors (Rate Gyro)
infrared
6.8: Vision-Based Sensing
integral
8.2: Model-Free Dual-PID Mini-Segway Control System Architecture
integration
1.0: Prelude
interactive development
10: Appendix A - MATLAB and Simulink Software
interdisciplinary
1.0: Prelude
internal combustion
5.4: Discrete Electromagnetic Valve Actuation (EMVA)
internal combustion engines
5.7: Chapter References
inverse Laplace transform
2.1: Laplace Transforms
inverter circuit
5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control
IoT
7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi
iterative design process
7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process

K

Kalman filter
6.5: Sensor Fusion
Kirchhoff's law
2.3: Electrical and Electronic System Modeling

L

Lab
11.8: PID Tuning for Mini-Segway Hardware System
lab objectives
11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink
11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)
11.4: DC-Motor Control Utilizing Arduino Hardware
11.5: Simulink GUI (Graphic User Interface) Design and Programming
lab sessions
11: Appendix B - Mechatronics Lab Sessions
lane detection
6.8: Vision-Based Sensing
Laplace
4.1: Analog Signals and Their Frequency Analysis
Laplace transform
2.1: Laplace Transforms
2.5: System Linearization
3.3: The First and Second Order System Responses and Associated Pole Locations
Laplace transforms
4: Signal Processing
lateral motion model
9.2: Dual- and Triple-PID Control of a Three-Wheel Cart
learning objectives
1: Introduction
2: Mechatronic System Modeling
4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)
8.8: Model-Based Control Experiments
linear actuators
5.1: Mechatronic Actuator Background
linear system
2.5: System Linearization
linearization
8.3: Model-Based Mini-Segway Control
linearized differential equations
8.3: Model-Based Mini-Segway Control
linearized model
8.1: Mini-Segway System Modeling
8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers
loop gain
2.4: Mechatronic System Modeling
LPV control
9.7: Chapter References
LQG
3.5: Review of Model-Based Control Design and Gain Tuning Methods
LQR
3.5: Review of Model-Based Control Design and Gain Tuning Methods
9.3: Model-Based Control of a Three-Wheel Cart

LQTI control
9.4: Simulation Study of PID and Model-Based Control Strategies
LQTI controller
9.6: Experimental Investigation of Model-Based Control Strategy

M

machine efficiency
5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control
machine learning
10: Appendix A - MATLAB and Simulink Software
2: Mechatronic System Modeling
2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning
3.0: Prelude
3.6: Machine Learning-Based Control
magnetic field
6.2: Rotational Speed Sensors and Their Speed Calculation
magnitude
4.1: Analog Signals and Their Frequency Analysis
mathematical model
2.0: Prelude
mathematical modeling
7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process
MATLAB
10: Appendix A - MATLAB and Simulink Software
11: Appendix B - Mechatronics Lab Sessions
11.0: MATLAB/Simulink Installation
11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink
11.6: Sensor Fusion vs. Single Sensor
7: Mechatronic Control Systems
7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi
8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller
8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers
9.0: Prelude
9.5: Experimental Investigation of PID Control Strategies
9.6: Experimental Investigation of Model-Based Control Strategy
MATLAB example
2.7: Chapter References
matrix operations
10: Appendix A - MATLAB and Simulink Software
MCU
7.2: Microcontrollers Used for Mechatronic Systems
measurements
6.5: Sensor Fusion
mechanical
1.0: Prelude
2.2: Mechanical System Modeling
mechanical energy
5.1: Mechatronic Actuator Background
mechanical engineering
1: Introduction
4.0: Prelude
mechanical subsystems
2.0: Prelude
mechanical system modeling
2: Mechatronic System Modeling
mechatronic
1.0: Prelude
7.0: Prelude
7.1: Mechatronic System Architecture
8: Mini-Segway System — Mechatronic System Example I
8.0: Prelude
9.0: Prelude
mechatronic actuators
5.1: Mechatronic Actuator Background
mechatronic sensors
6.1: Mechatronic Sensor Background
mechatronic system
1.2: Introduction of Mechatronic Systems
11: Appendix B - Mechatronics Lab Sessions
2.0: Prelude
7: Mechatronic Control Systems
7.1: Mechatronic System Architecture
8.1: Mini-Segway System Modeling
9: Three-Wheel Cart - Mechatronic System Example 2
9.1: Three-Wheel Cart System Modeling
mechatronic systems
1: Introduction
1.1: Introduction of Mechatronic Systems
3.0: Prelude
4.0: Prelude
5.0: Prelude
6.0: Prelude
7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process
9.3: Model-Based Control of a Three-Wheel Cart
mechatronics
1.1: Introduction of Mechatronic Systems
1.2: Introduction of Mechatronic Systems
1.3: How to Use This Book
2.4: Mechatronic System Modeling
3: System Responses and Controls
5: Mechatronic System Component- Actuators
5.2: Mechatronic Actuator Background
6: Mechatronic System Component- Sensors
7.4: Mechatronic Control System Architecture- Speed Control Example
medical engineering
6.9: Chapter References
memory
7.2: Microcontrollers Used for Mechatronic Systems
7.6: Chapter References
MEMS
6.4: Angular Velocity Sensors (Rate Gyro)
6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing
microcontroller
1.0: Prelude
4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)
7.0: Prelude
7.2: Microcontrollers Used for Mechatronic Systems
7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi
7.4: Mechatronic Control System Architecture- Speed Control Example
7.6: Chapter References
microcontrollers
7: Mechatronic Control Systems
microprocessor
6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing
7.2: Microcontrollers Used for Mechatronic Systems
microprocessors
6.0: Prelude
7: Mechatronic Control Systems
MKR 1000
7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi
model
5.5: Discrete Electrohydraulic Valve Actuation (EHVA)
model based control design
2.0: Prelude
modeling
1.2: Introduction of Mechatronic Systems
2.2: Mechanical System Modeling
2.4: Mechatronic System Modeling
9: Three-Wheel Cart - Mechatronic System Example 2
9.0: Prelude
9.1: Three-Wheel Cart System Modeling
modeling errors
2.0: Prelude
modeling process
2.7: Chapter References
modeling techniques
2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning
modelling
8.0: Prelude
modern control systems
5.7: Chapter References
motion
2.2: Mechanical System Modeling
motion control
11.5: Simulink GUI (Graphic User Interface) Design and Programming
motor carrier board
7.4: Mechatronic Control System Architecture- Speed Control Example
motor control
5.3: DC-Motor Drive Circuits
9.1: Three-Wheel Cart System Modeling
9.4: Simulation Study of PID and Model-Based Control Strategies
9.5: Experimental Investigation of PID Control Strategies
motor efficiency
5.2: Mechatronic Actuator Background
motor speed
5.3: DC-Motor Drive Circuits
motor speed control
11.4: DC-Motor Control Utilizing Arduino Hardware
motor speed measurement
11.4: DC-Motor Control Utilizing Arduino Hardware
motors
5: Mechatronic System Component- Actuators
MPU 6050
6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing
9.2: Dual- and Triple-PID Control of a Three-Wheel Cart

N

navigation
6.9: Chapter References
negative feedback
2.4: Mechatronic System Modeling
noise
6.5: Sensor Fusion
nonlinear
5.5: Discrete Electrohydraulic Valve Actuation (EHVA)
nonlinear model
8.1: Mini-Segway System Modeling
nonlinear system
2.5: System Linearization
nonlinear systems
3.6: Machine Learning-Based Control
8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers
9.7: Chapter References
notch filter
4.4: Analog and Digital Filter Design
NTC thermistor
6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing
numerical computing
10: Appendix A - MATLAB and Simulink Software

O

Objectives
11.8: PID Tuning for Mini-Segway Hardware System
obstacles
11.9: Mini-Segway Stabilization and Drive Project
operational amplifier
2.3: Electrical and Electronic System Modeling
operational principle
5.2: Mechatronic Actuator Background
optimal control
3.5: Review of Model-Based Control Design and Gain Tuning Methods
optimization
3.6: Machine Learning-Based Control
8.0: Prelude
8.8: Model-Based Control Experiments
optimization toolbox
10: Appendix A - MATLAB and Simulink Software
ordinary differential equations
2: Mechatronic System Modeling
2.1: Laplace Transforms
2.2: Mechanical System Modeling
orientation control
6.4: Angular Velocity Sensors (Rate Gyro)

P

parameter estimation
2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning
parameters
11.8: PID Tuning for Mini-Segway Hardware System
pendulum
8.1: Mini-Segway System Modeling
pendulum angle
8.7: Experimental Study of the Mini-Segway Model-Free Control
pendulum stabilization
8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller
percentage overshoot
3.3: The First and Second Order System Responses and Associated Pole Locations
performance
11.7: PID Tuning for Mini-Segway Simulink Model
11.8: PID Tuning for Mini-Segway Hardware System
5.5: Discrete Electrohydraulic Valve Actuation (EHVA)
8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers
performance analysis
1.2: Introduction of Mechatronic Systems
9.3: Model-Based Control of a Three-Wheel Cart
9.5: Experimental Investigation of PID Control Strategies
performance indices
1.3: How to Use This Book
3.0: Prelude
performance measures
3: System Responses and Controls
performance metrics
8.8: Model-Based Control Experiments
peripherals
7.2: Microcontrollers Used for Mechatronic Systems
permanent magnet synchronous motor
5.0: Prelude
permanent magnetic synchronous motor
5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control
phase
4.1: Analog Signals and Their Frequency Analysis
PID
11.9: Mini-Segway Stabilization and Drive Project
PID control
1.2: Introduction of Mechatronic Systems
3: System Responses and Controls
3.4: PID Control and Its Gain Tuning Methods
3.6: Machine Learning-Based Control
6.1: Mechatronic Sensor Background
7.0: Prelude
8.2: Model-Free Dual-PID Mini-Segway Control System Architecture
9.4: Simulation Study of PID and Model-Based Control Strategies
9.5: Experimental Investigation of PID Control Strategies
9.7: Chapter References
PID controller
1.3: How to Use This Book
11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink
11.5: Simulink GUI (Graphic User Interface) Design and Programming
11.7: PID Tuning for Mini-Segway Simulink Model
11.8: PID Tuning for Mini-Segway Hardware System
3.0: Prelude
5.3: DC-Motor Drive Circuits
8.7: Experimental Study of the Mini-Segway Model-Free Control
9.0: Prelude
9.2: Dual- and Triple-PID Control of a Three-Wheel Cart
PID controllers
3.7: Chapter References
PID gain selection
8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller
PID tuning
11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink
plot
4.1: Analog Signals and Their Frequency Analysis
positioning
6.9: Chapter References
power electronics
5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control
5.7: Chapter References
power requirements
7.6: Chapter References
practical application
9.2: Dual- and Triple-PID Control of a Three-Wheel Cart
pressure sensing
6: Mechatronic System Component- Sensors
pressure sensors
6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing
processing
8.0: Prelude
project
11.9: Mini-Segway Stabilization and Drive Project
proportional
8.2: Model-Free Dual-PID Mini-Segway Control System Architecture
pulse train
6.2: Rotational Speed Sensors and Their Speed Calculation
PWM
4.6: Pulse-Width and Frequency Modulated Signals
5.3: DC-Motor Drive Circuits
PWM signal
11.4: DC-Motor Control Utilizing Arduino Hardware
PWM signals
5.0: Prelude

Q

quantization error
4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)

R

rate gyro
6.4: Angular Velocity Sensors (Rate Gyro)

rate gyro sensor
6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing

reading assignment
2.7: Chapter References

references
6.9: Chapter References

regression model
2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning

regulation tracking
8: Mini-Segway System — Mechatronic System Example I

reinforcement learning
1.4: Chapter References

reliability requirements
7.6: Chapter References

resistance temperature detector
6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing

resistor
2.3: Electrical and Electronic System Modeling

response
4.1: Analog Signals and Their Frequency Analysis

response performance
3.3: The First and Second Order System Responses and Associated Pole Locations
8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller

Riccati equation
8.3: Model-Based Mini-Segway Control

rigid body dynamics
9.1: Three-Wheel Cart System Modeling

rise time
11.7: PID Tuning for Mini-Segway Simulink Model

robotics
11.9: Mini-Segway Stabilization and Drive Project
5.1: Mechatronic Actuator Background

root_locus_analysis
8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input

rotational actuators
5.1: Mechatronic Actuator Background

rotational speed calculations
6.2: Rotational Speed Sensors and Their Speed Calculation

rotational speed sensor
6.2: Rotational Speed Sensors and Their Speed Calculation

rotational speed sensors
6: Mechatronic System Component- Sensors

RTD
6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing

S

sample and hold
11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)

sample theory
4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters

sampling rate
4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters

script editor
11.0: MATLAB/Simulink Installation

second order system
3.3: The First and Second Order System Responses and Associated Pole Locations

second order systems
11: Appendix B - Mechatronics Lab Sessions
3: System Responses and Controls

sensing technologies
6.1: Mechatronic Sensor Background

sensor
6.5: Sensor Fusion

sensor applications
6.4: Angular Velocity Sensors (Rate Gyro)

sensor fusion
11.6: Sensor Fusion vs. Single Sensor
6: Mechatronic System Component- Sensors
6.0: Prelude
6.1: Mechatronic Sensor Background
6.8: Vision-Based Sensing
6.9: Chapter References

sensor information
11.5: Simulink GUI (Graphic User Interface) Design and Programming

sensor integration
9.6: Experimental Investigation of Model-Based Control Strategy

sensor selection
7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process

sensor technology
6.4: Angular Velocity Sensors (Rate Gyro)
6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing

sensors
1.1: Introduction of Mechatronic Systems
6: Mechatronic System Component- Sensors
6.0: Prelude
6.9: Chapter References
8.0: Prelude

settling time
3.3: The First and Second Order System Responses and Associated Pole Locations

signal
8.0: Prelude

signal conditioning
4.4: Analog and Digital Filter Design

signal conversion
4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions)

signal processing
11.2: Analog and Digital Systems and Fast Fourier Transform (FFT)
11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi
4: Signal Processing
4.0: Prelude
4.6: Pulse-Width and Frequency Modulated Signals
4.7: Advanced Analog and Digital Filter Design and Implementation
6: Mechatronic System Component- Sensors
6.0: Prelude

signal processing algorithms
4.3: Digital Signals and Their Frequency Analysis

Signal Processing Toolbox
11.0: MATLAB/Simulink Installation

signals
6.5: Sensor Fusion

simulation
11: Appendix B - Mechatronics Lab Sessions
11.0: MATLAB/Simulink Installation
5.5: Discrete Electrohydraulic Valve Actuation (EHVA)
8.3: Model-Based Mini-Segway Control
8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller
9.0: Prelude
9.1: Three-Wheel Cart System Modeling

simulation studies
1.2: Introduction of Mechatronic Systems
2.0: Prelude

simulation study
7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process
8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers

simulation validation
9: Three-Wheel Cart - Mechatronic System Example 2

Simulink
10: Appendix A - MATLAB and Simulink Software
11: Appendix B - Mechatronics Lab Sessions
11.0: MATLAB/Simulink Installation
11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink
11.4: DC-Motor Control Utilizing Arduino Hardware
11.6: Sensor Fusion vs. Single Sensor
11.7: PID Tuning for Mini-Segway Simulink Model
11.8: PID Tuning for Mini-Segway Hardware System
7: Mechatronic Control Systems
7.4: Mechatronic Control System Architecture- Speed Control Example
8.2: Model-Free Dual-PID Mini-Segway Control System Architecture
8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller
8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers
8.7: Experimental Study of the Mini-Segway Model-Free Control
9.4: Simulation Study of PID and Model-Based Control Strategies
9.5: Experimental Investigation of PID Control Strategies
9.6: Experimental Investigation of Model-Based Control Strategy

Simulink implementation
11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi

Simulink subsystem
11.5: Simulink GUI (Graphic User Interface) Design and Programming

sinusoidal
4.1: Analog Signals and Their Frequency Analysis

sinusoidal input
3.1: Impulse, Step, and Sinusoidal Responses

SISO system
2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning
software
7.1: Mechatronic System Architecture
7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi
software architecture
7: Mechatronic Control Systems
7.4: Mechatronic Control System Architecture- Speed Control Example
8.7: Experimental Study of the Mini-Segway Model-Free Control
software tool chain
7.6: Chapter References
speed
11.9: Mini-Segway Stabilization and Drive Project
speed control
5.1: Mechatronic Actuator Background
5.2: Mechatronic Actuator Background
7.4: Mechatronic Control System Architecture- Speed Control Example
speed control strategy
9.2: Dual- and Triple-PID Control of a Three-Wheel Cart
speed measurement
6.2: Rotational Speed Sensors and Their Speed Calculation
stability
11.9: Mini-Segway Stabilization and Drive Project
3.4: PID Control and Its Gain Tuning Methods
stability analysis
3.7: Chapter References
stabilization
8.3: Model-Based Mini-Segway Control
8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input
stabilization performance
8.8: Model-Based Control Experiments
stable
11.7: PID Tuning for Mini-Segway Simulink Model
stable system
3.1: Impulse, Step, and Sinusoidal Responses
State Flow
11.6: Sensor Fusion vs. Single Sensor
state space
6.5: Sensor Fusion
state_variables
8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input
Stateflow
11.0: MATLAB/Simulink Installation
steady state
2.7: Chapter References
step functions
2: Mechatronic System Modeling
step response
3.1: Impulse, Step, and Sinusoidal Responses
stepper motors
5.7: Chapter References
surface micromachining
6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing
system
7.1: Mechatronic System Architecture
8.0: Prelude
system analysis
2.0: Prelude
3: System Responses and Controls
system architecture
1: Introduction
1.1: Introduction of Mechatronic Systems
7.0: Prelude
system controllability
3.5: Review of Model-Based Control Design and Gain Tuning Methods
system dynamics
1.2: Introduction of Mechatronic Systems
2.4: Mechatronic System Modeling
8.8: Model-Based Control Experiments
system identification
1.3: How to Use This Book
2: Mechatronic System Modeling
2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning
2.7: Chapter References
system model linearization
2.5: System Linearization
system modeling
1.4: Chapter References
3.0: Prelude
8: Mini-Segway System — Mechatronic System Example I
system observability
3.5: Review of Model-Based Control Design and Gain Tuning Methods
system outputs
11.7: PID Tuning for Mini-Segway Simulink Model
system performance
1.1: Introduction of Mechatronic Systems
3.1: Impulse, Step, and Sinusoidal Responses
3.3: The First and Second Order System Responses and Associated Pole Locations
3.4: PID Control and Its Gain Tuning Methods
system pole locations
3.3: The First and Second Order System Responses and Associated Pole Locations
system requirements
7.2: Microcontrollers Used for Mechatronic Systems
system response
11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink
system responses
3.7: Chapter References
system specifications
7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process
system stability
3: System Responses and Controls
3.0: Prelude
system transfer function
3.1: Impulse, Step, and Sinusoidal Responses
system_dynamics
8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input
systems
1.0: Prelude
2.2: Mechanical System Modeling
5: Mechatronic System Component- Actuators

T

technical report
2.7: Chapter References
technology
6.5: Sensor Fusion
temperature requirements
7.6: Chapter References
temperature sensing
6: Mechatronic System Component- Sensors
6.1: Mechatronic Sensor Background
temperature sensors
6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing
textbook
9.7: Chapter References
theorem
3.1: Impulse, Step, and Sinusoidal Responses
thermal camera
6.8: Vision-Based Sensing
thermistor
6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing
thermocouple
6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing
time domain
2.1: Laplace Transforms
time domain signals
4.3: Digital Signals and Their Frequency Analysis
time response
3.3: The First and Second Order System Responses and Associated Pole Locations
tire dynamics
9.1: Three-Wheel Cart System Modeling
toolboxes
11.0: MATLAB/Simulink Installation
torque
5: Mechatronic System Component- Actuators
5.1: Mechatronic Actuator Background
8.1: Mini-Segway System Modeling
torque control
5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control
torque generation
5.2: Mechatronic Actuator Background
tracking control
8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers
9.3: Model-Based Control of a Three-Wheel Cart
trajectory design
9.5: Experimental Investigation of PID Control Strategies
9.6: Experimental Investigation of Model-Based Control Strategy
transfer function
2.1: Laplace Transforms
2.3: Electrical and Electronic System Modeling
2.4: Mechatronic System Modeling
2.5: System Linearization
2.7: Chapter References
3.3: The First and Second Order System Responses and Associated Pole Locations
4.4: Analog and Digital Filter Design
transfer functions
2: Mechatronic System Modeling
2.2: Mechanical System Modeling

transfer_function
8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input
tuning
11.8: PID Tuning for Mini-Segway Hardware System
3.6: Machine Learning-Based Control
3.7: Chapter References
tuning process
11.7: PID Tuning for Mini-Segway Simulink Model
turning speed
9.2: Dual- and Triple-PID Control of a Three-Wheel Cart

U

undergraduate classes
1: Introduction
unit impulse function
2.1: Laplace Transforms
unit step function
2.1: Laplace Transforms
unmanned aerial vehicle
6.8: Vision-Based Sensing

V

valve
5.4: Discrete Electromagnetic Valve Actuation (EMVA)
5.5: Discrete Electrohydraulic Valve Actuation (EHVA)
variable valve actuation
5.7: Chapter References
vector control
5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control
vehicle dynamics
2.5: System Linearization
9.1: Three-Wheel Cart System Modeling
voltage sensing
6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing

This page titled Index is shared under a Public Domain license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

GLOSSARY

This page titled Glossary is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

GLOSSARY

$$\frac{1}{} \tag{1}$$

Keywords and Definitions

A/D converter: This electronic device transforms an analog voltage signal from the physical domain to a digital number in the information domain with a predetermined number of bits.

Accelerometer transfer function: This mathematical representation, typically a ratio of Laplace transforms in the s-domain, describes the relationship between the input acceleration and the sensor output voltage.

Affine function: This mathematical function is a linear transformation with a nonzero offset.

Aliasing: This effect occurs when a sampled signal appears to have a lower frequency than its true value, causing a misrepresentation of the original signal. More specifically, it occurs when a signal is sampled at a rate lower than twice of its highest frequency, as described by the Nyquist-Shannon sampling theorem.

Analog filter: These filters operate on continuous, unsampled signals. They are often constructed using physical components like resistors, capacitors and inductors. One example of an analog filter application is the anti-aliasing filter before the A/D converter.

Analog frequency response: This measure of how a dynamic system reproduces or processes signals of different frequencies can be calculated using the Laplace transform and can be presented in Bode plot.

Analog rotational speed sensing: This sensor produces a continuous, varying analog signal, such as a sine wave or a voltage proportional to speed, to measure the speed of a rotating object.

Analog signal: This continuously variable physical measurement, such as voltage or current, is used to represent real-world phenomena like sound or temperature, differing from digital signals that use discrete binary values.

Anti-aliasing filter (**AAF**): This analog filter is used before a signal sampler (such as A/D converter) to restrict the bandwidth of the sampled signal so that the Nyquist-Shannon sampling theorem is satisfied over the band of interest.

Anti-windup PID controller: This PID control mechanism is implemented in either a continuous- or discrete-time domain. It limits positive integration when the PID control output is saturated above and negative integration when the PID control output is saturated below.

Arduino IDE software: Arduino IDE (integrated development environment) is an open-source software application designed for writing, compiling, and uploading code to Arduino microcontroller boards. It serves as the primary interface for users to interact with Arduino hardware.

Arduino microcontroller: This microcontroller provides accessibility to beginners for building interactive electronic projects, allowing them to read inputs from sensors and control outputs like LEDs, motors, and other devices family.

Arduino MKR 1000 Wi-Fi: This microcontroller board is designed for internet of things (IoT) projects, featuring a built-in WiFi module for internet connectivity, an ARM Cortex-M0+ processor, a cryptographic chip for secure communications, and an onboard battery charger for easy portability.

Arduino MKR motor carrier board: This add-on board is designed to control a variety of motors (servo, DC, and stepper) and to connect other sensors. Compatible with the Arduino MKR family, it provides connections for four servo motors, four DC motors (two high-performance and two standard), two encoder sensors, and four analog sensors.

Arduino Simulink library: This selection of Simulink blocks is specifically designed for interacting with Arduino hardware. These blocks enable configuration and access to Arduino sensors, actuators, and communication interfaces (e.g., I2C, SPI, serial).

Asymptotical stability: Asymptotic stability refers to a property of a dynamical system where after being disturbed, the system will return to its equilibrium state as time approaches infinity.

Back electromotive force coefficient (K_m): This constant relates a DC motor rotational speed to the back EMF (electromotive force) voltage generated, which is the voltage that opposes the armature voltage applied.

Band-kill (notch) filter: This filter attenuates signals with its frequency between the selected low and high cutoff (corner) frequencies. It passes signals with frequencies lower than the low cutoff frequency and higher than the high cutoff frequency.

Band-pass filter: This filter passes signals with its frequency between the selected low and high cutoff (corner) frequencies. It attenuates signals with frequencies lower than the low cutoff frequency and higher than the high cutoff frequency.

BIBO stability: It is a system property, where any finite and limited input to a dynamic system will result in a finite and limited output.

Bilinear transformation: This method is useful for converting a continuous-time system (in the Laplace transform or s-domain) into a discrete-time system (in the z-domain) for applications in signal processing and control systems.

Black-box model: This system model is constructed based upon the system input and output data, assuming no system model architecture (structure).

Bode plot: It provides two frequency response graphs of a system transfer function, which are a combination of a Bode magnitude plot with magnitude in decibels and a Bode phase plot with phase in degrees. For both plots, the horizontal axis of angular frequency is in logarithm of base 10.

Bounded-input and bounded-output (BIBO) stability: In this system property, any finite and limited input to a dynamic system will result in a finite and limited output.

Butterworth filter: This type of signal processing filter (analog or digital) is designed to have a frequency response as flat as possible in the passband. It is also referred to as a maximally flat magnitude filter.

Capacitance-based acceleration sensor: This accelerometer employs the changing capacitance of a moving mass to measure dynamic changes in mechanical variables.

Chebyshev filter: This filter, which is either analog or digital, has a steeper roll-off frequency response than the Butterworth filter and has either a passband ripple (type I) or a stopband ripple (type II). A Chebyshev filter minimizes the error between the idealized and the actual filter characteristic over the operating frequency range of the filter.

Closed-loop control: It uses feedback signals to automatically adjust system inputs to maintain desired output by continuously monitoring and comparing actual outputs against set points.

Complementary filter: This filter combines data from two different sensors to develop a more accurate estimate of a system state, such as orientation.

Continuous-time PID controller: This PID control mechanism is implemented in the continuous-time domain. That is, both integration and derivative calculations are in the continuous-time domain.

Control system architecture: This structure comprises the physical hardware components and the software that organizes and controls them. The hardware components include sensors, microcontrollers, and actuators. The software architecture defines the overall structure.

Coordinate transformation: It is the process of converting coordinates from one reference system to another, which is essential for comparing or aligning different frames of reference.

Coriolis force: This inertia (or fictitious) force applies to objects in motion within a reference coordinate (frame) that rotates with respect to the inertia coordinate.

CPU: Short for central processing unit, this electronic brain of a computer is responsible for executing instructions, performing calculations, and managing most system operations.

Coulomb friction model: This model of dry friction states that friction force is directly proportional to the normal force pressing two surfaces together and is independent of the contact area.

Current source: This electrical circuit provides a constant electric current to other circuits, regardless of the voltage across its terminals.

D/A converter: This electronic device transforms a digital number in the information domain to an analog voltage signal to be used in the physical domain.

DC-AC converter: This electronic device converts direct current (DC) power into alternating current (AC) power. It uses electronic switches to rapidly switch the DC power on and off, reversing the polarity to create an AC waveform.

DC motor: It is an electrical machine that converts direct current (DC) electrical energy into mechanical energy, using magnetic forces to produce rotational motion.

DC-motor torque coefficient (K_τ): This constant value is expressed in Newton-meter per Ampere (Nm/A), which represents the relationship between the DC motor armature current and the produced torque (produced torque is equal to the product of K_τ and armature current).

DC-motor transfer function: This system transfer function has DC motor armature voltage as input and motor speed or displacement as output.

Detailed system model: Also called a digital twin, this dynamic, virtual replica of a mechatronic system is created and validated using real-time data from sensors and other devices.

Digital filter: In digital signal processing (DSP), digital filter is an algorithm that mathematically manipulates sampled, discrete-time signals to achieve desired characteristics, such as separating signals or restoring a signal.

Digital signal: This signal represents data as a sequence of discrete values. At any given time, it can take on, at most, only one of a finite number of values. The signal contrasts with an analog signal, which represents continuous values.

Direct PID gain tuning: This PID control gain turning method is for systems that can be stabilized with a proportional (P) gain. The turning starts with P gain by pushing the closed-loop system to marginally stable as gain increases and then set the P gain around half of the marginally stable one. We then fix the P gain and tune the integral (I) gain in the same way as the P gain; and last fix both P and I gains, tuning the derivative gain in the same way as the P and I gains.

Direct torque control: This advanced AC motor control technique directly regulates the motor's electromagnetic torque and magnetic flux, rather than indirectly controlling motor currents.

Discrete Fourier transform (**DFT):** This transform is a discrete version of the continuous Fourier transform. It converts a finite sequence of equally spaced samples of a function into a same-length sequence of equally spaced samples of the discrete Fourier transform, a complex-valued function of frequency.

Discrete solenoid linear actuator: This linear actuator is powered by one or more solenoids with two distinguished states: rest and activated.

Discrete state actuator: This device can be set to a specific, limited number of positions or states, unlike a continuous actuator that can move to any position within a range.

Discrete-time PID controller: This PID control mechanism is implemented in the discrete-time domain. That is, both integration and derivative calculations are in the discrete-time domain.

Dual-loop model-based control strategy: This model-based control architecture has one continuous-time inner-loop controller and one discrete-time outer loop controller.

Dual-PID control of a three-wheel cart: This three-wheel cart control strategy uses two PID controllers. One regulates the right wheel speed to track its reference and the other controls the left wheel speed. The cart forward driving and turning motion can be achieved by adjusting the reference speeds of the two driven wheels independently.

Dual-PID mini-Segway control: This mini-Segway control strategy uses two PID controllers. One stabilizes the pendulum and maintains it at an upright position. The other drives the mini-Segway.

Dual-PID strategy validation: The real-time Simulink model of a dual-PID control strategy is implemented and validated

experimentally for the three-wheel cart.

EHVA: This electro-hydraulic magnetic valve actuation (EHVA) is used in internal combustion engines to actuate intake and exhaust valves.

Electric/electronic circuit modeling: It is the process of creating a virtual representation of a circuit for analyzing and simulating its behavior before building a physical prototype.

EMVA: Short for electromagnetic valve actuation, EMVA is used in internal combustion engines to actuate intake and exhaust valves.

EMVA equivalent circuit: This circuit is a model for the electromagnetic valve actuator in an internal combustion engine

Equation of motion (EOM): This mathematical function describes how the position, velocity, and acceleration of a physical system change over time.

Fast Fourier transform (FFT): This highly efficient algorithm analyzes complex signals by breaking them down into their constituent frequencies. It transforms time-domain data into a frequency-domain representation, enabling quick identification of signal components, such as those in audio signals or vibrations. FFT is crucial in many fields, including signal processing, engineering, music, and science, due to its fast transform speed advantage over the standard discrete Fourier transform (DFT).

Feasible conversion region: An NMP continuous-time system can be converted into a MP discrete-time system using a special sample and hold input. The feasible conversion region is the one with the selected sample period and duty-cycle leading to a MP discrete-time system.

Field-oriented control: This method for controlling AC motors mathematically separates the motor stator current into two orthogonal components: one for magnetic flux (d-axis) and one for torque (q-axis).

Finite element analysis: Continuous mechanical system modeling.

First order system: The behavior of this dynamical system can be described by a first order ordinary differential equation (ODE). The system is characterized by a single energy storage element, resulting in a transfer function with one pole defined by its time constant.

Fixed-point: In computing, fixed point is a method of representing fractional (non-integer) numbers by storing a fixed number of digits of their fractional part.

Floating-point: This method represents non-integer, decimal values in a computer, similar to scientific notation. Computers use the IEEE 754 standard to store these numbers, dividing the available bits into a sign, an exponent, and a mantissa, which determines magnitude and precision of the number.

Flow sensor: This device measures the rate or quantity of a liquid or gas moving through a pipe or duct.

FM: The frequency modulated (FM) signal has fixed duty-cycle with varying frequency. The single can be in the form of varying frequency sinusoidal or square wave signals.

Gain margin: It measures the absolute stability of a control system, indicating how much extra gain can be added before the system becomes unstable. It can be determined at the frequency where the phase shift reaches -180°, and a higher gain margin (expressed in dB) signifies a more stable system.

Gain- scheduled PID control: This strategy adjusts the proportional (P), integral (I), and derivative (D) gains of a PID controller based on the current operating conditions of a dynamic system.

Generic algorithm (GA): This optimization technique mimics natural selection to find solutions to complex problems, or to a generic programming concept where algorithms are written to be flexible and reusable across various data types.

Gray-box model: This system model is constructed based upon the system input and output data with a given system model architecture (structure).

H-bridge: This actuator drive circuit is capable of driving an actuator (such as a DC motor) in two directions.

Hall-effect sensor: This magnetic sensor can be used for detecting the strength and direction of a magnetic field. The field might be produced from either a permanent magnet or an electromagnet with its output varying in proportion to the strength of the magnetic field being detected. Hall-effect sensors are solid state devices. They are becoming increasingly popular because they can be used in many different types of applications such as sensing position, velocity, and directional movement.

High-pass filter: This filter passes signals with a frequency higher than a selected cutoff (corner) frequency and attenuates signals with frequencies lower than the cutoff frequency.

High-side driver circuit: This circuit controls a power transistor by connecting the load between the transistor drain or collector and the ground with the transistor source or emitter connected to a power supply.

Hardware-in-the-Loop (HiL) simulation: This technique validates mechatronic control systems by connecting real controller hardware to a virtual, real-time simulated environment. It accelerates development and improves safety by enabling testing of control algorithms without needing the full, physical system.

IGBT: Short for insulated-gate bipolar transistor, this is a type of electronic switch used in power electronics. It combines the high input impedance and fast switching of a MOSFET with the high current-handling capability and low saturation voltage of a bipolar junction transistor.

Image/video-based sensor: This device detects and converts light waves or other electromagnetic radiation into electrical signals to form a digital image or video sequence in the information domain.

Impedance: It is the effective resistance of an electric circuit or component to alternating current due to the combined effects of ohmic resistance and reactance, such as inductance and capacitance.

Impulse response: It is the system output with an impulse function as input.

Incremental encoder: This linear or rotary electromechanical device has two output signals, A and B, that issue pulses when the device is moved.

Induction motor: This electric motor uses electromagnetic induction to create rotation, making it the most common type of electric motor used in industrial and household applications.

Inertia-based acceleration sensor: This accelerometer measures the linear acceleration of an object by detecting the force of inertia on a tiny, suspended mass.

Information and physical domain interfaces: These interfaces connect the digital information domain and the analog physical domain. For example, signal conditioning circuits (such as anti-aliasing filters) connect from physical to information domains whereas power amplifiers connect from information to physical domains.

Infrared (IR) camera: Also called a thermal camera, this device detects and captures images of infrared radiation emitted by objects, which is invisible to the human eye.

Infrared (IR) image: This visual representation is created by capturing infrared radiation, which is a form of electromagnetic radiation invisible to the human eye.

Inverse Laplace transform: This integral transform converts a frequency-domain function into a function of a real variable in time domain.

Kalman filter: This filter is an optimal, recursive algorithm that estimates the state of a dynamic system (e.g., position, velocity) by combining noisy sensor measurements with a mathematical model, providing more accurate results than using data source alone.

Lane detection: This is computer vision process identifies and localizes lane markings on a road, typically for autonomous or semi-autonomous driving systems.

Laplace transform: It is an integral transform converting a time-domain function into a function of a complex variable, denoted by "s", representing frequency or the Laplace domain.

Least Squares (LS): This statistical and mathematical method is used to find the best-fit model (often a line or curve) for a

given set of data by minimizing the sum of the squared differences (residuals) between the observed data points and the values predicted by the model.

Lenz Law: If a wire cuts through a magnetic field, it experiences a force that is in the opposite direction to its velocity, which is an example of Lenz Law. That is, the direction of the induced EMF opposes the change that caused it.

Linear actuator: This device converts energy into linear motion to push, pull, lift, or tilt a load. It is often realized by transforming rotational energy from an electric motor into linear motion via components like lead screws and nuts, by using direct electric-magnetic actuation, or by directly using fluid pressure in hydraulic or pneumatic cylinders.

Linear system model: It is a mathematical representation of a dynamic system with its output as a dynamic linear function of its input, meaning it obeys the principles of superposition (the response to a sum of inputs is the sum of the responses to each input) and homogeneity/scaling (scaling the input by a constant also scales the output by the same constant).

Lorentz force law: This law describes the total electromagnetic force on a charged particle as the sum of electric and magnetic forces.

Low-pass filter: This filter passes signals with a frequency lower than a selected cutoff (corner) frequency and attenuates signals with frequencies higher than the cutoff frequency.

Low-side driver circuit: This circuit controls a power transistor by connecting the load between the power supply and the transistor drain or collector with the transistor source or emitter connected to the ground.

LPV model: Short for linear parameter-varying model, this model's dynamics vary as a function of certain time-varying parameters called scheduling parameters. An LPV model is often represented in a state-space form with independent model coefficients (parameters).

LQG control: LQG is an acronym for linear quadratic Gaussian. LQG control is a modern state-space control design technique to obtain optimal dynamic regulators and servo controllers. This technique allows us to trade off regulation performance and control effort, considering disturbances and measurement noise.

LQR: LQR is an acronym for linear quadratic regulation, which is a model-based design method for controllers and regulators. It minimizes a weighted quadratic summation of performance and control effort.

LQR mini-Segway pendulum stabilization: This mini-Segway pendulum position regulation control utilizes the LQR (linear quadratic regulation) control technique.

LQR SiL simulation: This SiL simulation in Simulink for the mini-Segway utilizes the LQR state feedback controller with linearized model.

LQR control of three-wheel cart: This three-wheel cart control strategy regulates any initial cart speed to its rest position.

LQTI control of three-wheel cart: This three-wheel cart control strategy tracks both cart and turning speed references so that the cart can be driven at wish.

LQTI control strategy implementation: This LQTI control strategy is realized by replacing the matrix-vector multiplication with entry-by-entry multiplication.

LQTI mini-Segway control: This mini-Segway control strategy stabilizes its pendulum and tracks the desired mini-Segway speed simultaneously, utilizing LQTI (linear quadratic tracking with integration) control technique.

LQTI SiL simulation: This SiL simulation in Simulink for the mini-Segway utilizes the LQTI state feedback controller with a linearized model.

LQTI SiL simulation with Coulomb friction: This SiL simulation in Simulink for the mini-Segway utilizes the LQTI state feedback controller based on a nonlinear model with Coulomb friction.

LQTI strategy with feedforward control: This LQTI strategy combines both open-loop feedforward control with the closed-loop LQTI control to generate control signals for both DC motors in the three-wheel cart.

Mass-damper-spring system: This mechanical system consists of discrete mass nodes interconnected via a network of springs and dampers.

MATLAB and Simulink: These are a pair of software environments from MathWorks used for technical computing and model-based control design.

Mechatronic actuator: A mechatronic actuator is a device that generates force, motion, heat, or other forms of effects based upon a control command produced in the information domain. It utilizes the supplied electric, hydraulic, or other forms of energies.

Mechatronic sensor: A mechatronic sensor is a device that converts physical variables into electronic signals to be input to microcontrollers of a mechatronic system.

Mechatronic system: It consists of mechanical, electrical/electronic, and mechatronic subsystems. A microcontroller (or multiple microcontrollers) in the information domain is often used to collect the sensor and reference signals from a mechatronic system. It then makes decisions based upon the collected signals inside the microcontroller, converts these decisions into actuator commands, and sends them to the related actuators for execution in a closed-loop manner.

Mechatronic system model: It is a mathematical representation of a given mechatronic system using differential equations or frequency domain transfer functions.

MEMS: An acronym for micro-electro-mechanical systems, MEMS is a class of devices that integrates microscopic mechanical components with microelectronic circuitry on a single silicon chip. They produce micrometer-scaled sensors, actuators, and other tools that are too small to be seen with the naked eye.

MEMS architecture: This three-dimensional structure integrates micromechanical components like sensors and actuators with electronic circuits on a single silicon chip.

Microcontroller: This compact, self-contained integrated circuit (IC) functions as a miniature computer on a single chip, housing a central processing unit, memory, and input/output peripherals designed to control specific functions of an electronic device or embedded system.

Microcontroller architecture: This internal design of a microcontroller includes a CPU, memory (RAM and ROM), and input/output (I/O) ports, all integrated onto a single chip.

Microprocessor unit (MPU): This single-chip computer contains the CPU's core components, such as an arithmetic and logic unit (ALU), control unit, and registers.

Minimal phase (MP) system: This linear system has all its zeroes located at the open-left half complex plane.

Mini-Segway: It is a two-wheeled, self-balancing robot (mechatronic system) that is a miniaturized version of the original Segway human transporter.

Mini-Segway equations of motion: This set of differential equations describe the mini-Segway dynamic and motion.

Mini-Segway hardware architecture: The mini-Segway consists of two geared motors with wheel attached, an Arduino MKR 1000 microcontroller with MKR motor carrier board attached, and the three-axis accelerometer and rate gyro.

Mini-Segway model: It describes the mini-Segway model dynamics with differential equations for the two-wheeled and self-balancing robot.

Mini-Segway software architecture: The mini-Segway software is implemented in a MATLAB and Simulink environment consisting of sensor reading and actuator driving software, dual PID controllers, and a user interface for driving the mini-Segway.

Model-based control gain tuning: This approach adopts a process of designing a sequence of closed-loop controllers with gain low to high and then evaluating them with control gain from low-to-high to find the best closed-loop performance.

MOSFET: Short for metal oxide semiconductor field-effect transistor, this semiconductor device is used for switching. It acts as a voltage-controlled switch with the voltage applied to the gate terminal controlling the current flow between the drain and the source terminals.

Moving average (MA) filter: This filter averages a defined number of consecutive data points to remove high-frequency noise and smooth a signal, acting as a low-pass filter.

MPU 6050: This IMU (inertia measurement unit) integrates both three-axis accelerometers and three-axis gyroscopes on a single chip.

No-load speed: This measure refers to the maximum speed that can be achieved by a DC motor without any load torque applied.

Nonlinear system model: This model describes a relationship with its dynamic output not directly proportional to the input. Thus, it does not follow the principles of superposition and homogeneity/scaling.

Nonlinear three-wheel cart model for SiL simulation: This nonlinear model of three-wheel cart system is developed for SiL simulations with all feedback signals required for dual-PID, triple-PID, and LQTI control strategies.

Non-minimal phase (NMP) system: This linear system has at least one zero located at the open-right half complex plane**.**

NSGA-II (Non-dominated Sorting Genetic Algorithm II): This popular, elitist algorithm solves multi-objective optimization problems.

Nyquist–Shannon sample theorem: This theorem is an essential principle for digital signal processing; it links the frequency range of a signal and the sample rate required to avoid a type of distortion called aliasing. The theorem states that the sample rate must be more than twice the bandwidth of the signal in order to avoid aliasing. In practice, it is used to select band-limiting filters that keep aliasing below an acceptable amount when an analog signal is sampled.

OCC: Short for output covariance constraint, this optimal control design method minimizes the quadratic control effort subject to multiple quadratic output constraints.

Open-loop control: In this non-feedback control method, the control action is independent of the resulting process output. That is, the control does not use output feedback to adjust its operation.

Operational amplifier (OA): A high-gain voltage amplifier with a differential input and a single output, it is designed to perform mathematical operations in analog circuits.

Operational amplifier circuit modeling: This device measures angular velocity (rotation rate) by using the Sagnac effect to analyze the difference in travel time between two counter-propagating beams of light.

Optical rate gyro sensor: It is a device that measures angular velocity (rotation rate) by using the Sagnac effect to analyze the difference in travel time between two counter-propagating beams of light.

Partial fractional expansion (PFE): This method for decomposing a rational function into a sum of simpler terms makes it easier to find the inverse Laplace transform and determine the time-domain response.

Percentage overshoot (PO): For a step input, the percentage overshoot is a measure of how much a system response exceeds its final steady-state value, expressed as a percentage of the final value. It is calculated as the difference between the maximum peak value and the final value, divided by the final value, then multiplied by 100%.

Permanent-magnet synchronous motor (PMSM): This motor uses permanent magnets embedded in the rotor to create a constant magnetic field. The stator carries windings connected to an AC electricity supply to produce a rotating magnetic field (as in an asynchronous motor). Note that the AC supply can also be created using a power inverter with DC power supply.

Physics-based sensor: This device detects and measures a physical quantity by applying the principles of physics to convert a stimulus, like heat or motion, into a measurable electronic signal.

PID (proportional-integral-derivative) control: This control loop mechanism continuously calculates an error value and applies a correction based on three weighted terms: proportional (P), integral (I), and derivative (D). The mechanism then brings system output to a desired set point and maintains its stability.

PID DC motor speed control: This control utilizes a feedback mechanism based on the difference between a desired motor speed and the actual speed. It continuously adjusts the armature voltage of a DC motor via a PID control signal.

Piezoelectric accelerometer: This sensor measures dynamic acceleration and vibration by utilizing the piezoelectric effect, where materials like quartz or ceramics generate an electric charge under mechanical stress.

Pole assignment: This method, which is used in control system theory, places the closed-loop poles of a dynamic system modeled in its state-space form in predetermined locations in the complex plane.

Pole location: In control systems, it determines the system stability and performance, with locations in the left half of the complex plane indicating stability.

Pressure sensor: This device measures the force of a gas or liquid and converts it into an electrical signal.

PWM: An acronym for pulse-width modulated, the PWM signal is often used as a control signal for DC motors to reduce the energy lose in the power amplifier.

q-Markov Cover: Short for q-Markov covariance equivalent realization, this system identification method is based upon system responses associated with either white noise, impulse, or PRBS inputs.

Quantization: In mathematics and digital signal processing, this process maps input values from a large set (often a continuous set) to output values in a (countable or digital) smaller set, often with a finite number of elements.

Rate gyro: This type of gyroscope measures the angular velocity (or angular rate) of an object, rather than its direction.

Rogowski coil: This toroid of wire measures an alternating current $i(t)$ through a cable encircled by the toroid.

Routh-Hurwitz criterion: It is a mathematical test in control system that determines if a linear time-invariant (LTI) system is stable by examining the coefficients of its characteristic equation without solving for the roots.

Sample and hold: This circuit captures an analog voltage signal at a specific instant and holds it at that constant level until it is triggered to sample again.

Second order system: The behavior of this dynamical system can be described by a second order ordinary differential equation. The system is characterized by a second-order transfer function with two poles defined by undamped natural frequency and damping ratio.

Sensor fusion: This process combines data from multiple sensors to produce more accurate, reliable, and comprehensive information than any single sensor alone.

Settling time: For a unit step input, the system settling time (t_s) is the time for the system to reach its given error band of 2% (or 5%), where the error band is defined as the range between $1 + 2\%$ (or 5%) and $1 - 2\%$ (or 5%). Note that if 2% (or 5%) is used, it is called 2% (or 5%) settling time.

Signal-to-noise ratio (SNR): This ratio measures the strength of a desired signal compared to the strength of background noise, indicating signal quality. A higher SNR signifies a stronger signal relative to noise, leading to a clearer output.

SiL simulation: Short for software-in-the-loop simulation, this method tests and validates control software with a physical system model in a simulation environment, in order to quickly catch control software bugs and evaluate closed-loop system performance.

Simulation validation in Simulink This SiL simulation technique is implemented in MATLAB and Simulink software platform.

Sinusoidal response: This response is the system output with a sinusoidal function as input.

Slip-angle model: Slip angle is the angle between the direction a tire is pointed and the direction it is traveling. An empirical model was developed in this book.

Special sample and hold input: Different from the conventional zero-order sample and hold input, the special sample and hold input uses a special function, such as square pulse, forward and backward triangles, over the sample period.

Speed calculation using encoder signals: Speed is calculated from encoder signals first by measuring pulses over a set time or by measuring the time between pulses, and then using the encoder's pulses per revolution (PPR) to convert the data to speed.

Stall torque: This measure is the maximum rotational torque a motor produces when its shaft is completely stopped or locked, meaning that its shaft speed is zero.

State-space model of mini-Segway: This mini-Segway model utilizes a set of first-order differential or difference equations that represent its internal state and its relationship to observable outputs.

State-space model of three-wheel cart: This model linearizes the nonlinear three-wheel cart model at its operational condition without any lateral-slip.

Step motor: Also known as a stepper motor or a stepping motor, brushless DC electric motor rotates in a series of small and discrete angular steps.

Step response: It is the system output with a step function as input.

Step response performance indices: These performance indices include system response time, settling time, system overshoot, system steady state error, etc.

System model linearization: In mathematics, linearizing a nonlinear model is finding a linear model that approximates the nonlinear one at a given operational condition.

Taylor expansion: Also called Taylor series, this expansion mathematically expresses a given function as an infinite sum of terms relative to the function's derivatives at a given point (called the equilibrium point).

Temperature sensor: This device measures temperature by converting thermal energy into an electrical signal.

Three-wheel cart: A three-wheel cart system has two driving wheels in the front and one swivel caster wheel in the back.

Three-wheel cart model: This dynamic model of the three-wheel cart system is developed based upon Newton's law, considering wheel lateral slip.

Three-wheel cart SiL simulation: This SiL simulation environment is used to study the three-wheel cart performance for given driving and turning references utilizing dual-PID, triple-PID, and LQTI control strategies.

Time constant: This constant is a fundamental measure of how quickly a system responds to a change (e.g., unit step input), taking the time of one time constant to reach 63.2% of its new steady-state value.

Torque ripple: This ripple is the periodic fluctuation in the output torque of an electric motor, causing unsmooth rotation.

Transfer function: This Laplace transform is a mathematical model representing input-output relationship of a linear system as a ratio of polynomials, simplifying complex differential equations into algebraic ones for easier analysis, design, and stability checks in control systems and electronics.

Triple-PID control of a three-wheel cart: This three-wheel cart control strategy uses three PID controllers. The first and second PID controllers regulate the left and right driven wheel speeds to track their references, respectively, and the third PID controller tracks the cart turning speed reference.

Triple-PID strategy validation: The real-time Simulink model of triple-PID control strategy is implemented and validated experimentally for the three-wheel cart.

V-design process: This framework for mechatronic systems combines hardware and software development. Each development phase is paired with either simulation or experimental validation to ensure early identification of design frauds. An iterative process is also embedded in the V-design process to improve system performance.

Vibration-based gyroscope (sensor): This Coriolis vibratory gyroscope uses a vibrating structure (as opposed to rotating) as its orientation reference.

z-transform: It is a mathematical tool converting a discrete-time signal or system from its time-domain representation into a complex frequency-domain representation called the z-domain. The discrete-time equivalent of the Laplace transform, it is used in digital signal processing (DSP) and control theory to analyze and design digital systems by simplifying difference equations into algebraic ones.

Ziegler-Nichols method: It is a popular, pragmatic approach for tuning PID controller gains. Note that a PID controller is a type of industrial control system used to maintain desired process variables.

This page titled Glossary A is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu (Co-DREAM OER).

DETAILED LICENSING

Overview

Title: Mechatronics: Fundamentals, Design, Integration, and Validation (Zhu)

Webpages: 106

Applicable Restrictions: Noncommercial

All licenses found:

- CC BY-NC 4.0: 99.1% (105 pages)
- Public Domain: 0.9% (1 page)

By Page

- Mechatronics: Fundamentals, Design, Integration, and Validation (Zhu) - *CC BY-NC 4.0*
  - Front Matter - *CC BY-NC 4.0*
    - TitlePage - *CC BY-NC 4.0*
    - InfoPage - *CC BY-NC 4.0*
    - Table of Contents - *CC BY-NC 4.0*
    - Licensing - *CC BY-NC 4.0*
    - Acknowledgements - *CC BY-NC 4.0*
  - 1: Introduction - *CC BY-NC 4.0*
    - 1.0: Prelude - *CC BY-NC 4.0*
    - 1.1: Introduction of Mechatronic Systems - *CC BY-NC 4.0*
    - 1.2: Introduction of Mechatronic Systems - *CC BY-NC 4.0*
    - 1.3: How to Use This Book - *CC BY-NC 4.0*
    - 1.4: Chapter References - *CC BY-NC 4.0*
  - 2: Mechatronic System Modeling - *CC BY-NC 4.0*
    - 2.0: Prelude - *CC BY-NC 4.0*
    - 2.1: Laplace Transforms - *CC BY-NC 4.0*
    - 2.2: Mechanical System Modeling - *CC BY-NC 4.0*
    - 2.3: Electrical and Electronic System Modeling - *CC BY-NC 4.0*
    - 2.4: Mechatronic System Modeling - *CC BY-NC 4.0*
    - 2.5: System Linearization - *CC BY-NC 4.0*
    - 2.6: Introduction of Gray- and Black-Box Modeling Using System Identification and Machine Learning - *CC BY-NC 4.0*
    - 2.7: Chapter References - *CC BY-NC 4.0*
  - 3: System Responses and Controls - *CC BY-NC 4.0*
    - 3.0: Prelude - *CC BY-NC 4.0*
    - 3.1: Impulse, Step, and Sinusoidal Responses - *CC BY-NC 4.0*
    - 3.2: System Stability and Its Performance Measures - *CC BY-NC 4.0*
    - 3.3: The First and Second Order System Responses and Associated Pole Locations - *CC BY-NC 4.0*
    - 3.4: PID Control and Its Gain Tuning Methods - *CC BY-NC 4.0*
    - 3.5: Review of Model-Based Control Design and Gain Tuning Methods - *CC BY-NC 4.0*
    - 3.6: Machine Learning-Based Control - *CC BY-NC 4.0*
    - 3.7: Chapter References - *CC BY-NC 4.0*
  - 4: Signal Processing - *CC BY-NC 4.0*
    - 4.0: Prelude - *CC BY-NC 4.0*
    - 4.1: Analog Signals and Their Frequency Analysis - *CC BY-NC 4.0*
    - 4.2: Conversion between Analog and Digital Signals (A/D and D/A Conversions) - *CC BY-NC 4.0*
    - 4.3: Digital Signals and Their Frequency Analysis - *CC BY-NC 4.0*
    - 4.4: Analog and Digital Filter Design - *CC BY-NC 4.0*
    - 4.5: Nyquist-Shannon Sample Theorem, Sample Aliasing, and Anti-Aliasing Filters - *CC BY-NC 4.0*
    - 4.6: Pulse-Width and Frequency Modulated Signals - *CC BY-NC 4.0*
    - 4.7: Advanced Analog and Digital Filter Design and Implementation - *CC BY-NC 4.0*
    - 4.8: Chapter References - *CC BY-NC 4.0*
  - 5: Mechatronic System Component- Actuators - *CC BY-NC 4.0*
    - 5.0: Prelude - *CC BY-NC 4.0*
    - 5.1: Mechatronic Actuator Background - *CC BY-NC 4.0*
    - 5.2: Mechatronic Actuator Background - *CC BY-NC 4.0*
    - 5.3: DC-Motor Drive Circuits - *CC BY-NC 4.0*
    - 5.4: Discrete Electromagnetic Valve Actuation (EMVA) - *CC BY-NC 4.0*
    - 5.5: Discrete Electrohydraulic Valve Actuation

(EHVA) - *CC BY-NC 4.0*
- 5.6: Introduction of Permanent Magnetic Motor Principle, Drive, and Control - *CC BY-NC 4.0*
- 5.7: Chapter References - *CC BY-NC 4.0*

◦ 6: Mechatronic System Component- Sensors - *CC BY-NC 4.0*
- 6.0: Prelude - *CC BY-NC 4.0*
- 6.1: Mechatronic Sensor Background - *CC BY-NC 4.0*
- 6.2: Rotational Speed Sensors and Their Speed Calculation - *CC BY-NC 4.0*
- 6.3: Acceleration Sensors - *CC BY-NC 4.0*
- 6.4: Angular Velocity Sensors (Rate Gyro) - *CC BY-NC 4.0*
- 6.5: Sensor Fusion - *CC BY-NC 4.0*
- 6.6: MEMS-Based Integrated Acceleration and Angular Velocity Sensing - *CC BY-NC 4.0*
- 6.7: Temperature, Pressure, Flow, Current, and Voltage Sensing - *CC BY-NC 4.0*
- 6.8: Vision-Based Sensing - *CC BY-NC 4.0*
- 6.9: Chapter References - *CC BY-NC 4.0*

◦ 7: Mechatronic Control Systems - *CC BY-NC 4.0*
- 7.0: Prelude - *CC BY-NC 4.0*
- 7.1: Mechatronic System Architecture - *CC BY-NC 4.0*
- 7.2: Microcontrollers Used for Mechatronic Systems - *CC BY-NC 4.0*
- 7.3: Sample Microcontroller- Arduino MKR 1000 Wi-Fi - *CC BY-NC 4.0*
- 7.4: Mechatronic Control System Architecture- Speed Control Example - *CC BY-NC 4.0*
- 7.5: Iterative Mechatronic Control System Development and Validation Process- The V-Design Process - *CC BY-NC 4.0*
- 7.6: Chapter References - *CC BY-NC 4.0*

◦ 8: Mini-Segway System — Mechatronic System Example I - *CC BY-NC 4.0*
- 8.0: Prelude - *CC BY-NC 4.0*
- 8.1: Mini-Segway System Modeling - *CC BY-NC 4.0*
- 8.2: Model-Free Dual-PID Mini-Segway Control System Architecture - *CC BY-NC 4.0*
- 8.3: Model-Based Mini-Segway Control - *CC BY-NC 4.0*
- 8.4: Model-Based Mini-Segway Control with Special Sample and Hold Input - *CC BY-NC 4.0*
- 8.5: Mini-Segway SiL Simulation Validation of a Dual-PID Controller - *CC BY-NC 4.0*
- 8.6: Mini-Segway SiL Simulation Validation of Model-Based Controllers - *CC BY-NC 4.0*
- 8.7: Experimental Study of the Mini-Segway Model-Free Control - *CC BY-NC 4.0*
- 8.8: Model-Based Control Experiments - *CC BY-NC 4.0*
- 8.9: Chapter References - *CC BY-NC 4.0*

◦ 9: Three-Wheel Cart - Mechatronic System Example 2 - *CC BY-NC 4.0*
- 9.0: Prelude - *CC BY-NC 4.0*
- 9.1: Three-Wheel Cart System Modeling - *CC BY-NC 4.0*
- 9.2: Dual- and Triple-PID Control of a Three-Wheel Cart - *CC BY-NC 4.0*
- 9.3: Model-Based Control of a Three-Wheel Cart - *CC BY-NC 4.0*
- 9.4: Simulation Study of PID and Model-Based Control Strategies - *CC BY-NC 4.0*
- 9.5: Experimental Investigation of PID Control Strategies - *CC BY-NC 4.0*
- 9.6: Experimental Investigation of Model-Based Control Strategy - *CC BY-NC 4.0*
- 9.7: Chapter References - *CC BY-NC 4.0*

◦ 10: Appendix A - MATLAB and Simulink Software - *CC BY-NC 4.0*

◦ 11: Appendix B - Mechatronics Lab Sessions - *CC BY-NC 4.0*
- 11.0: MATLAB/Simulink Installation - *CC BY-NC 4.0*
- 11.1: Simulation of the 1st and 2nd Order Systems in MATLAB/Simulink - *CC BY-NC 4.0*
- 11.2: Analog and Digital Systems and Fast Fourier Transform (FFT) - *CC BY-NC 4.0*
- 11.3: Filter Design and Introduction to Arduino MKR 1000 Wi-Fi - *CC BY-NC 4.0*
- 11.4: DC-Motor Control Utilizing Arduino Hardware - *CC BY-NC 4.0*
- 11.5: Simulink GUI (Graphic User Interface) Design and Programming - *CC BY-NC 4.0*
- 11.6: Sensor Fusion vs. Single Sensor - *CC BY-NC 4.0*
- 11.7: PID Tuning for Mini-Segway Simulink Model - *CC BY-NC 4.0*
- 11.8: PID Tuning for Mini-Segway Hardware System - *CC BY-NC 4.0*
- 11.9: Mini-Segway Stabilization and Drive Project - *CC BY-NC 4.0*

◦ Back Matter - *CC BY-NC 4.0*
- Index - *Public Domain*
- Glossary - *CC BY-NC 4.0*
- Glossary A - *CC BY-NC 4.0*
- Detailed Licensing - *CC BY-NC 4.0*

This page titled Mechatronics: Fundamentals, Design, Integration, and Validation (Zhu) is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Guoming Zhu.