

Introduction to Feedback Control

USING DESIGN STUDIES

RANDAL W. BEARD
TIMOTHY W. McLAIN

Revised: October 27, 2017

© 2016 Randal W. Beard
All rights reserved.

This work may not be distributed and/or modified without written consent of the authors.

Contents

1	Introduction	1
	Design Study A: Single Link Robot Arm	4
	Design Study B: Pendulum on a Cart	5
	Design Study C: Satellite Attitude Control	6
I	Simulation Models	7
2	Kinetic Energy of Mechanical Systems	11
3	The Euler Lagrange Equations	24
II	Design Models	34
4	Equilibria and Linearization	36
5	Transfer Function Models	45
6	State Space Models	56
III	PID Control Design	68
7	Pole Placement for Second Order Systems	72
8	Design Strategies for Second Order Systems	78
9	System Type and Integrators	99
10	Digital Implementation of PID Controllers	116
IV	Observer Based Control Design	128
11	Full State Feedback	130
12	Integrator with Full State Feedback	154
13	Observers	172
14	Disturbance Observers	195
V	Loopshaping Control Design	213
15	Frequency Response of LTI Systems	215
16	Frequency Domain Specifications	237
17	Stability and Robustness Margins	256
18	Compensator Design	271
VI	Homework Problems	322
	Design Study D: Mass Spring Damper	323
	Design Study E: Ball on Beam	331

CONTENTS

Design Study F: Planar VTOL	340
Appendices	322
a Creating Animations in Simulink	353
b Modeling in Simulink Using S-functions	369
c Review of Ordinary Differential Equations	383
d Numerical Solutions to Differential Equations	390
e Root Locus	393
f Review of Linear Algebra	402
Bibliography	411
Index	411

Preface

Does the world really need another introductory textbook on feedback control? For many years we have felt that the many quality textbooks that currently exist are more than sufficient. However, several years ago the first author (RWB) had an experience that changed our minds. I had just finished teaching the introductory feedback control course at BYU. I had a need for a new masters student in my research group and hired one of the top students in the class. We had designed a gimbal system for a small unmanned air vehicle, and needed a control system implemented on a microcontroller for the gimbal. I tasked this new masters student to model the gimbal and design the control system. I was surprised by how much the student struggled with this task, especially understanding where to begin and how to model the gimbal. If I had given him a transfer function and asked him to design a PID controller, or if I had given him a state space model and asked him to design an observer and controller for the system, he would not have had any difficulty. But he did not know how to do an end-to-end design, that required developing models for the system, including physical constraints. It was this experience that convinced me that my current approach to teaching feedback control was inadequate.

The next time that I taught the course, rather than focusing on the theory of feedback design, I focused the lectures on the end-to-end design process. Accordingly, I spent significantly more time talking about the modeling process, adding a few lectures on the Euler-Lagrange method. I also added several lectures on linearization about set points, and developing linear models. When I taught feedback design methods, I focused on design specifications, and the need to account for saturation, sensor noise, model uncertainty, and external disturbances on the system. Over the next several years, we developed several complete design studies and then used these design studies throughout the course to illustrate the material. That course reorganization, and the realization that existing textbooks on feedback control did not fit the pedagogy, were the genesis of this textbook.

Therefore, this textbook is unique in several ways. Most importantly, it is focused, in its organization, in its examples, and in its homework problems, on a particular, but fairly general, end-to-end design strategy and methodology. Topics were included in the book only if they aided in the design process. For example, the book does not include a chapter on the root locus (although it is discussed in an appendix). The second unique feature is that the examples

CONTENTS

and homework problems follow a small set of design studies throughout the book. The text provides complete worked examples for three design studies: a single link robot arm, a pendulum on a cart, and a satellite attitude control problem. The homework problems parallel the examples with three additional design studies: a mass-spring-damper, a ball on beam, and a planar vertical take-off and landing aircraft, similar to a quadrotor. A third unique feature of the book is that complete solutions to the three example design studies are provided in Matlab/Simulink and Python at <http://controlbook.byu.edu>.

The design methodology culminates in computer code that implements the controllers. In the examples provided in the book, the computer code interacts with a simulation model of the plant composed of coupled nonlinear differential equations, rather than the physical plant. However, the example computer code is written in a way that can be directly implemented on a microcontroller, or other computing devices. We have used Matlab/Simulink because many students are already familiar with these tools. But we also provide Python examples because Python is free of charge and therefore accessible to everyone, and because for many applications, Python can be used to implement the controller in hardware.

One criticism of the book might be that all of the design problems are mechanical systems, and that they are fairly similar to each other. The student may come away with the impression that feedback control, and the design methods discussed in the book are only applicable to mechanical systems. Of course this is not true. The tools taught in the book are widely used in many applications including electrical power systems, disk drives, aerospace systems, chemical processing, biological systems, queueing systems, and many more. Although this book is focused on mechanical systems, we believe that the student who masters the material will be able to easily transition to other application domains. Mechanical systems, like those covered in this book, have compelling pedagogical advantage because the control objectives are intuitive, and the behavior of the system can be easily visualized. Therefore, we feel that they offer the best setting for developing intuition behind feedback design.

We had two other motivations for writing this textbook. The first is the exorbitant prices currently being charged for textbooks. Our intention is to provide an electronic version of this book to students free of charge. The book has required significant time and energy to develop, and so we hope that “free” is not equivalent “low quality.” We also intend to self-publish the book through Amazon for those desiring a hard copy. Given the amount of high quality open source software, students may get the wrong impression about monetary compensation for technical work. Good work should be compensated. However, since our day jobs provide adequate financial remuneration, we ask to be compensated by your feedback about your educational experience with the book, including typos and other errors. This will allow us to update and improve the book. We also hope that you will understand that we will probably not be very responsive to email asking for help with the homework problems. Another motivation for writing the book is our observation that this generation of students seem to prefer to learn using electronic resources. The book has therefore been

CONTENTS

written with the assumption that it will be studied electronically. Therefore, hyperlinks and other electronic aids are embedded throughout the text. We hope that these enhance the educational experience.

RWB
TWM

Chapter 1

Introduction

The objective of this book is to prepare the reader to design feedback control systems for dynamic systems and to lay the groundwork for further studies in the area. The design philosophy that we follow throughout the book is illustrated schematically in Figure 1.1. The objective is to design a control system for the “Physical System” shown in Figure 1.1. The physical system include actuators like motors and propellers, and sensors like accelerometers, gyros, GPS, cameras, and pressure sensors. The first step in the design process is to model the physical system using nonlinear differential equations. While approximations and simplifications will be necessary at this step, the hope is to capture in mathematics all of the important characteristics of the physical system. While there are many different methods for developing the equations of motion of the physical systems, in this book we will introduce one specific method, the Euler-Lagrange method, that is applicable to a wide variety of mechanical systems. We will for the most part, abstract the actuators to applied forces and torques on the system. Similarly the sensors will be abstracted to general measurements of certain physical quantities like position and velocity. The resulting model is called the “Simulation Model” as shown in Figure 1.1. In the design process, the simulation model is used for the high fidelity computer simulation of the physical system. Every control design will be tested thoroughly on the simulation model. However, the simulation model is only a mathematical approximation of the physical system, and simply because a design is effective on the simulation model, we should not assume that it will function properly on the physical system. Part I of the book introduces the Euler-Lagrange method and demonstrates how to derive simulation models for mechanical systems using this method.

The simulation model is typically nonlinear and high order and is too mathematically complex to be useful for control design. Therefore, to facilitate design, the simulation model is simplified and usually linearized to create lower-order design models. For any physical system, there may be multiple design models that capture certain aspects of the system that are important for a particular design. In this book we will introduce the two most common design models used

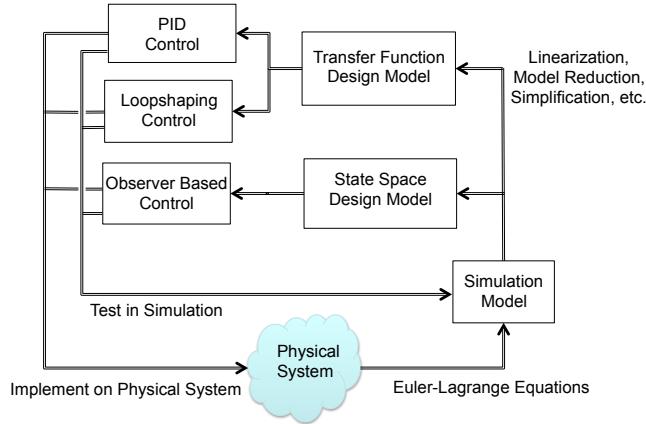


Figure 1.1: The design process. Using principles of physics, the physical system is modeled mathematically, resulting in the simulation model. The simulation model is simplified to create design models that are used for the control design. The control design is then tested and debugged in simulation and finally implemented on the physical system.

for control systems analysis and design: transfer function models and state space models. In both cases we will linearize the system about a particular operating condition. The advantage of transfer function models is that they capture the frequency dependence of the input-output relationship in dynamic systems. The advantage of state space models is that they are more intuitive and are better adapted to numerical implementation. Part II of the book shows how to linearize the equations of motion and how to create transfer function and state space design models.

The next three parts, Part III–V introduce three different control design methods. Part III shows how to design Proportional-Integral-Derivative (PID) controllers for second order systems. The PID method is the most common control technique used in industry. It can be understood from both a time-domain and frequency-domain perspective. The advantage of PID is its simplicity in design and implementation, and the fact that it can be used to achieve high performance for a surprising variety of systems. However, the disadvantage of PID control is that stability and performance can typically only be guaranteed for second order systems. Therefore Part III is focused exclusively on second order systems.

In Part IV we introduce the observer-based control method. Observer based methods can be thought of as a natural extension of PID controllers to high order systems. However, observer based methods are much more general than PID and can be used to obtain higher performance. In addition, most observer based techniques extend to nonlinear and time-varying systems. The disadvantage of observer-based methods is that they require a higher level of mathematical sophistication to understand and use, and they are primarily time-domain

CHAPTER 1. INTRODUCTION

methods.

The final design technique, which is covered in Part V is the loopshaping control design method. The loopshaping methods is a frequency domain techniques that explicitly addresses robustness of the system. The loopshaping method is also a natural extension of PID control to higher order systems. One of the advantage of loopshaping techniques is that they can be used when only a frequency response model of the system is available.

The book is organized around several design studies. For each new concept that is introduced, we will apply that concept to three specific design problems. The design problems are introduced in the following three sections and include a single link robot arm, a pendulum on a cart, and a simple satellite attitude control problem. The homework problems which are provided in Part VI will follow the same format with problems that are identical to those worked in the book, but applied to different physical systems. Our hope is to illustrate the control design process with specific, easy to understand problems. We note however, that the concepts introduced in this book are applicable to a much wider variety of problems than those illustrated in the book. We focus on mechanical systems, but a similar process (with the exception of the modeling section) can be followed for any system that can be modeled using ordinary differential equations. Examples include aerodynamic systems, electrical system, chemical processes, large structures like buildings and bridges, biological feedback systems, economic systems, queuing systems, and many more.

A Design Study: Single Link Robot Arm

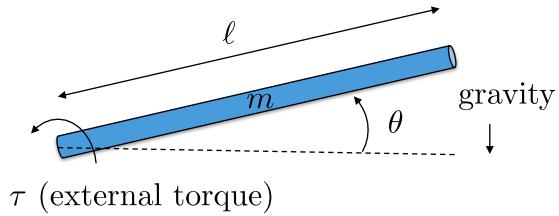


Figure 1.2: Single Link Robot Arm

Figure 1.2 shows a single link robot arm system. The robot arm has mass m and length ℓ . The angle of the robot is given by θ as measured from horizontal to the ground. The angular speed of the arm is $\dot{\theta}$. There is an applied torque τ at the joint. There is also a damping torque that opposes the rotation of the joint of magnitude, which can be modeled as $-b\dot{\theta}$.

The actual physical parameters of the system are $m = 0.5$ kg, $\ell = 0.3$ m, $g = 9.8$ m/s², $b = 0.01$ Nms. The torque is limited by $|\tau| \leq 1$ Nm.

Homework Problems (with solutions):

- | | |
|---|---|
| A.2 Kinetic energy. | A.d Root locus. |
| A.a Simulink animation. | A.10 Digital PID. |
| A.3 Equations of Motion. | A.11 Full state feedback. |
| A.b Simulink S-function. | A.12 Full state with integrator. |
| A.4 Linearize equations of motion. | A.13 Observer based control. |
| A.5 Transfer function model. | A.14 Disturbance observer. |
| A.6 State space model. | A.15 Frequency Response. |
| A.7 Pole placement using PD. | A.16 Loop gain. |
| A.8 Second order design. | A.17 Stability margins. |
| A.9 Integrators and system type. | A.18 Loopshaping design. |

B Design Study: Pendulum on a Cart

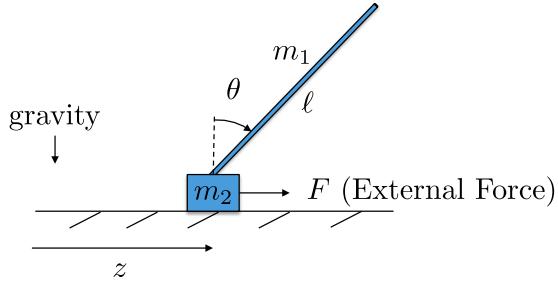


Figure 1.3: Pendulum on a cart.

Figure 1.3 shows the pendulum on a cart system. The position of the cart measured from the origin is z and the linear speed of the cart is \dot{z} . The angle of the pendulum from straight up is given by θ and the angular velocity is $\dot{\theta}$. The rod is of length ℓ and has mass m_1 is approximated as being infinitely thin. Gravity acts in the down direction. The only applied force is F which acts in the direction of z . The cart slides on a frictionless surface but air friction produces a damping force equal to $-b\dot{z}$.

The physical constants are $m_1 = 0.25$ kg, $m_2 = 1.0$ kg, $\ell = 1.0$ m, $g = 9.8$ m/s², $b = 0.05$ Ns. The force is limited by $|F| \leq 5$ N.

Homework Problems (with solutions):

- | | |
|---|---|
| B.2 Kinetic energy. | B.10 Digital PID. |
| B.a Simulink animation. | B.11 Full state feedback. |
| B.3 Equations of Motion. | B.12 Full state with integrator. |
| B.b Simulink S-function. | B.13 Observer based control. |
| B.4 Linearize equations of motion. | B.14 Disturbance observer. |
| B.5 Transfer function model. | B.15 Frequency Response. |
| B.6 State space model. | B.16 Loop gain. |
| B.8 Successive loop closure. | B.17 Stability margins. |
| B.9 Integrators and system type. | B.18 Loopshaping design. |
| B.d Root locus. | |

C Design Study: Satellite Attitude Control

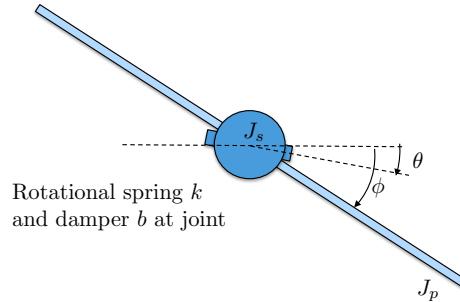


Figure 1.4: Satellite with flexible solar panels.

Figure 1.4 shows a simplified version of a satellite with flexible solar panels. We will model the flexible panels using a rigid sheet with moment of inertia J_p connected to the main satellite body by a torsional compliant element with spring constant k and damping constant b . The moment of inertia of the satellite J_s . The angle of the satellite body from the inertial reference is θ and the angle of the panel from the inertial reference is denoted as ϕ . Thrusters are used to command an external torque of τ about the axis of the satellite body.

The physical constants are $J_s = 5 \text{ kg m}^2$, $J_p = 1 \text{ kg m}^2$, $k = 0.15 \text{ N m}$, $b = 0.05 \text{ Nms}$. The torque is limited by $|\tau| \leq 5 \text{ Nm}$.

Homework Problems (with solutions):

- | | |
|---|---|
| C.2 Kinetic energy. | C.10 Digital PID. |
| C.a Simulink animation. | C.11 Full state feedback. |
| C.3 Equations of Motion. | C.12 Full state with integrator. |
| C.b Simulink S-function. | C.13 Observer based control. |
| C.5 Transfer function model. | C.14 Disturbance observer. |
| C.6 State space model. | C.15 Frequency Response. |
| C.8 Successive loop closure. | C.16 Loop gain. |
| C.9 Integrators and system type. | C.17 Stability margins. |
| C.d Root locus. | C.18 Loopshaping design. |

Part I

Simulation Models

To design efficient feedback control systems, we use mathematical models to understand the system, and then design controllers that achieve the desired specifications. Developing suitable mathematics models is problems specific and can be very complicated requiring specialized knowledge. However, there are some well known techniques for certain classes of systems. This book only covers one general technique that is useful for a large class of rigid body mechanical systems. The technique is known as the Euler-Lagrange method and requires a mathematical description of the kinetic and potential energy of the system, given its (generalized) position and velocity. The theory behind Euler-Lagrange method is deep, and well beyond the scope of this book. However, we will give a brief introduction that will enable the reader to apply Euler-Lagrange methods to a surprisingly large class of problems.

As motivation for the Euler-Lagrange equations, consider a particle of mass m in a gravity field, as shown in Figure 1.5, where g is the force of gravity at sea level, f_a is an externally applied force, and the friction due to air resistance is modeled by a gain b times the velocity \dot{y} . Since all of the forces are in the

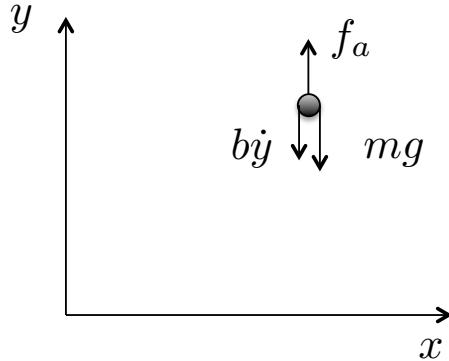


Figure 1.5: Point mass in a gravity field.

vertical direction, the motion of the vehicle can be completely described by the vertical position of particle $y(t)$. From Newton's law, the equations of motion are

$$m\ddot{y} = f_a - mg - b\dot{y}, \quad (1.1)$$

where dots over the variable will be used throughout the book to denote differentiation with respect to time, i.e., $\dot{y} \triangleq \frac{dy}{dt}$ and $\ddot{y} \triangleq \frac{d^2y}{dt^2}$. Rearranging (1.1) gives

$$m\ddot{y} + mg = f_a - b\dot{y}. \quad (1.2)$$

The kinetic energy for the particle is given by $K(\dot{y}) = \frac{1}{2}mv^2 = \frac{1}{2}m\dot{y}^2$. The potential energy, which is induced by gravitational pull proportional to the distance from the center of the earth, is given by $P(y) = P_0 + mgy$, where P_0 is the potential energy of the particle when $y = 0$. Note that if the velocity \dot{y} is

thought of as independent of y , then

$$\frac{\partial K(\dot{y})}{\partial \dot{y}} = m\dot{y}$$

and that

$$\frac{d}{dt} \left(\frac{\partial K(\dot{y})}{\partial \dot{y}} \right) = m\ddot{y}.$$

Also note that

$$\frac{\partial P(y)}{\partial y} = mg.$$

Therefore, Equation (1.2) can be written as

$$\frac{d}{dt} \left(\frac{\partial K(\dot{y})}{\partial \dot{y}} \right) + \frac{\partial P(y)}{\partial y} = f_a - b\dot{y}. \quad (1.3)$$

Equation (1.3) is a particular form of the so-called Euler-Lagrange equation. The generalization to more complicated system takes the form

$$\frac{d}{dt} \left(\frac{\partial K(y, \dot{y})}{\partial \dot{y}} - \frac{\partial P(y)}{\partial y} \right) - \left(\frac{\partial K(y, \dot{y})}{\partial y} - \frac{\partial P(y)}{\partial y} \right) = f_a - b\dot{y}, \quad (1.4)$$

which reduces to Equation (1.3) for our particular case since K is only a function of \dot{y} and P is only a function of y . To simplify the notation, we define the so-called *Lagrangian* as

$$L(y, \dot{y}) \triangleq K(y, \dot{y}) - P(y),$$

and write Equation (1.4) as

$$\frac{d}{dt} \left(\frac{\partial L(y, \dot{y})}{\partial \dot{y}} \right) - \left(\frac{\partial L(y, \dot{y})}{\partial y} \right) = f_a - b\dot{y}. \quad (1.5)$$

The position of the system can be generalized to both positions and angles, and the velocity can be generalized to velocity and angular velocities. Throughout the book we will use the notation $\mathbf{q} = (q_1, q_2, \dots, q_n)^\top$ to denote the general configuration of the mechanical system that we are modeling. The vector \mathbf{q} is called the generalized coordinates. The kinetic energy of the system is generally a function of both the generalized coordinates \mathbf{q} and the generalized velocity $\dot{\mathbf{q}}$, therefore we denote the kinetic energy as $K(\mathbf{q}, \dot{\mathbf{q}})$. On the other hand, in this book we will only consider potential fields that are conservative, and that only depend on the configuration variable \mathbf{q} . Under these assumptions, the general form of the Euler-Lagrange equations is given by

$$\begin{aligned} \frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_1} \right) - \frac{\partial L(\mathbf{q})}{\partial q_1} &= \tau_1 - b_{11}\dot{q}_1 - \dots - b_{1n}\dot{q}_n \\ \frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_2} \right) - \frac{\partial L(\mathbf{q})}{\partial q_2} &= \tau_2 - b_{21}\dot{q}_1 - \dots - b_{2n}\dot{q}_n \end{aligned} \quad (1.6)$$

$$\vdots \quad (1.7)$$

$$\frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_n} \right) - \frac{\partial L(\mathbf{q})}{\partial q_n} = \tau_n - b_{n1}\dot{q}_1 - \dots - b_{nn}\dot{q}_n,$$

where τ_i are the generalized forces and torques that are applied to the system.
Or in vector form we can write

$$\frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}} \right) - \frac{\partial L(\mathbf{q})}{\partial \mathbf{q}} = \boldsymbol{\tau}_a - B\dot{\mathbf{q}}, \quad (1.8)$$

where B is a matrix with positive elements and $\boldsymbol{\tau} = (\tau_1, \dots, \tau_n)^\top$ are the generalized forces. The Euler-Lagrange equation will be discussed in more detail in Chapter 3. However, before showing how to use Equation (1.8) to derive the equations of motion for practical systems, we need to show how to compute the kinetic and potential energy for general (holonomic) rigid body systems.

Chapter 2

Kinetic Energy of Mechanical Systems

2.1 Theory

Consider a point mass with mass m traveling at velocity \mathbf{v} , as shown in Figure 2.1. If the velocity is resolved with respect to a coordinate axis, then $\mathbf{v} = (v_x, v_y, v_z)^\top$. The kinetic energy of the point mass is given by

$$\begin{aligned} K &= \frac{1}{2}m\|\mathbf{v}\|^2 \\ &= \frac{1}{2}m\mathbf{v}^\top\mathbf{v} \\ &= \frac{1}{2}m(v_x^2 + v_y^2 + v_z^2). \end{aligned}$$

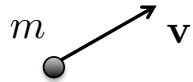


Figure 2.1: Point mass moving at velocity \mathbf{v} .

Now consider two point masses as shown in Figure 2.2. The total kinetic energy of the system is given by summing the kinetic energy of each of the point masses as

$$K = \frac{1}{2}m_1\mathbf{v}_1^\top\mathbf{v}_1 + \frac{1}{2}m_2\mathbf{v}_2^\top\mathbf{v}_2. \quad (2.1)$$

If the two masses are connected by a mass-less rod, and the particles are spinning about a point on the rod, as shown in Figure 2.2, where the angular velocity vector $\boldsymbol{\omega}$ points out of the page and the magnitude is given by $\dot{\theta}$, i.e., $\boldsymbol{\omega} =$

CHAPTER 2. KINETIC ENERGY OF MECHANICAL SYSTEMS

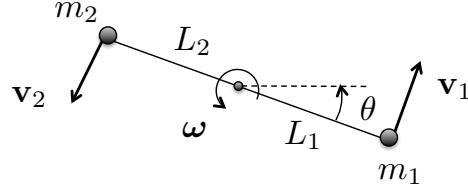


Figure 2.2: Two constrained point masses spinning about a point on adjoining axis.

$(0, 0, \dot{\theta})^\top$. From elementary physics, the velocity of point i is given by

$$\mathbf{v}_i = \boldsymbol{\omega} \times \mathbf{p}_i,$$

where in our case $\mathbf{p}_1 = (L_1 \cos \theta, L_1 \sin \theta, 0)^\top$, and $\mathbf{p}_2 = (-L_2 \cos \theta, -L_2 \sin \theta, 0)^\top$. Therefore

$$\begin{aligned}\mathbf{v}_1 &= \begin{pmatrix} 0 \\ 0 \\ \dot{\theta} \end{pmatrix} \times \begin{pmatrix} L_1 \cos \theta \\ L_1 \sin \theta \\ 0 \end{pmatrix} = \begin{pmatrix} -L_1 \dot{\theta} \sin \theta \\ L_1 \dot{\theta} \cos \theta \\ 0 \end{pmatrix} \\ \mathbf{v}_2 &= \begin{pmatrix} 0 \\ 0 \\ \dot{\theta} \end{pmatrix} \times \begin{pmatrix} -L_2 \cos \theta \\ -L_2 \sin \theta \\ 0 \end{pmatrix} = \begin{pmatrix} L_2 \dot{\theta} \sin \theta \\ -L_2 \dot{\theta} \cos \theta \\ 0 \end{pmatrix}.\end{aligned}$$

An alternative method for computing the velocities \mathbf{v}_1 and \mathbf{v}_2 is to differentiate \mathbf{p}_1 and \mathbf{p}_2 with respect to time to obtain

$$\begin{aligned}\mathbf{v}_1 &\triangleq \dot{\mathbf{p}}_1 = \frac{d\mathbf{p}_1}{dt} = \frac{d}{dt}(L_1 \cos \theta, L_1 \sin \theta, 0)^\top \\ &= \left(-L_1 \left(\frac{d\theta}{dt} \right) \sin \theta, \quad L_1 \left(\frac{d\theta}{dt} \right) \cos \theta, \quad 0 \right)^\top = \left(-L_1 \dot{\theta} \sin \theta, \quad L_1 \dot{\theta} \cos \theta, \quad 0 \right)^\top \\ \mathbf{v}_2 &\triangleq \dot{\mathbf{p}}_2 = \frac{d\mathbf{p}_2}{dt} = \frac{d}{dt}(-L_2 \cos \theta, -L_2 \sin \theta, 0)^\top \\ &= \left(L_2 \left(\frac{d\theta}{dt} \right) \sin \theta, \quad -L_2 \left(\frac{d\theta}{dt} \right) \cos \theta, \quad 0 \right)^\top = \left(L_2 \dot{\theta} \sin \theta, \quad -L_2 \dot{\theta} \cos \theta, \quad 0 \right)^\top.\end{aligned}$$

Therefore, using Equation (2.1) the kinetic energy is

$$\begin{aligned}K &= \frac{1}{2}m_1 \mathbf{v}_1^\top \mathbf{v}_1 + \frac{1}{2}m_2 \mathbf{v}_2^\top \mathbf{v}_2 \\ &= \frac{1}{2}m_1 \left[(-L_1 \dot{\theta} \sin \theta)^2 + (L_1 \dot{\theta} \cos \theta)^2 \right] + \frac{1}{2}m_2 \left[(L_2 \dot{\theta} \sin \theta)^2 + (-L_2 \dot{\theta} \cos \theta)^2 \right] \\ &= \frac{1}{2}m_1 \left[L_1^2 \dot{\theta}^2 \sin^2 \theta + L_1^2 \dot{\theta}^2 \cos^2 \theta \right] + \frac{1}{2}m_2 \left[L_2^2 \dot{\theta}^2 \sin^2 \theta + L_2^2 \dot{\theta}^2 \cos^2 \theta \right] \\ &= \frac{1}{2}(m_1 L_1^2 + m_2 L_2^2) \dot{\theta}^2.\end{aligned}$$

Now consider the case where there are N point masses spinning about a general angular velocity vector $\boldsymbol{\omega}$, where the position of the i^{th} point mass,

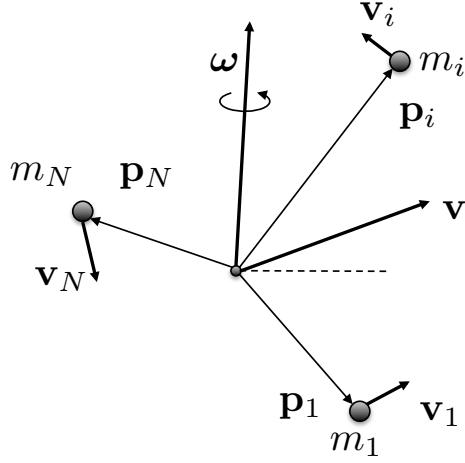


Figure 2.3: Many masses in a rigid body spinning about a common axis and moving at velocity \mathbf{v} .

$i = 1, \dots, N$ in a specified coordinate system are given by \mathbf{p}_i , as shown in Figure 2.3. We will see in the discussion below that it is most convenient to pick the coordinate system to be at the center of mass. The kinetic energy of the i^{th} particle is $K_i = \frac{1}{2}m_i\mathbf{v}_i^\top \mathbf{v}_i$. Therefore, the kinetic energy of the total system is given by

$$K = \sum_{i=1}^N \frac{1}{2}m_i\mathbf{v}_i^\top \mathbf{v}_i.$$

Using the fact that $\mathbf{v}_i = \boldsymbol{\omega} \times \mathbf{p}_i$ gives

$$K = \sum_{i=1}^N \frac{1}{2}m_i (\boldsymbol{\omega} \times \mathbf{p}_i)^\top (\boldsymbol{\omega} \times \mathbf{p}_i).$$

Recalling the cross product rule

$$(\mathbf{a} \times \mathbf{b})^\top (\mathbf{c} \times \mathbf{d}) = (\mathbf{a}^\top \mathbf{c})(\mathbf{b}^\top \mathbf{d}) - (\mathbf{a}^\top \mathbf{d})(\mathbf{b}^\top \mathbf{c})$$

gives

$$\begin{aligned} K &= \frac{1}{2} \sum_{i=1}^N m_i [(\boldsymbol{\omega}^\top \boldsymbol{\omega})(\mathbf{p}_i^\top \mathbf{p}_i) - (\boldsymbol{\omega}^\top \mathbf{p}_i)(\mathbf{p}_i^\top \boldsymbol{\omega})] \\ &= \frac{1}{2} \sum_{i=1}^N m_i \boldsymbol{\omega}^\top [(\mathbf{p}_i^\top \mathbf{p}_i)I_3 - \mathbf{p}_i \mathbf{p}_i^\top] \boldsymbol{\omega} \\ &= \frac{1}{2} \boldsymbol{\omega}^\top \left[\sum_{i=1}^N m_i ((\mathbf{p}_i^\top \mathbf{p}_i)I_3 - \mathbf{p}_i \mathbf{p}_i^\top) \right] \boldsymbol{\omega} \\ &\triangleq \frac{1}{2} \boldsymbol{\omega}^\top J \boldsymbol{\omega}. \end{aligned}$$

CHAPTER 2. KINETIC ENERGY OF MECHANICAL SYSTEMS

where I_3 is the 3×3 identity matrix. The matrix

$$J = \sum_{i=1}^N m_i ((\mathbf{p}_i^\top \mathbf{p}_i) I_3 - \mathbf{p}_i \mathbf{p}_i^\top)$$

is called the inertia matrix of the mass system. If the mass system is solid then there is an infinite number of particles, and the sum becomes an integral resulting in

$$J = \int_{\mathbf{p} \in \text{body}} ((\mathbf{p}^\top \mathbf{p}) I_3 - \mathbf{p} \mathbf{p}^\top) dm(\mathbf{p}),$$

where $dm(\mathbf{p})$ is the elemental mass located at position \mathbf{p} and the integral is over all elemental masses in the system.

Suppose that $\mathbf{p} = (x, y, z)$ is expressed with respect to a coordinate system fixed in a rigid body. Then

$$\begin{aligned} J &= \int_{(x,y,z) \in \text{body}} \left[(x \ y \ z) \begin{pmatrix} x \\ y \\ z \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} x \\ y \\ z \end{pmatrix} (x \ y \ z) \right] dm(x, y, z) \\ &= \int_{(x,y,z) \in \text{body}} \left[(x^2 + y^2 + z^2) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} x^2 & xy & xz \\ yx & y^2 & yz \\ zx & zy & z^2 \end{pmatrix} \right] dm(x, y, z) \\ &= \int_{(x,y,z) \in \text{body}} \begin{pmatrix} y^2 + z^2 & -xy & -xz \\ -yx & x^2 + z^2 & -yz \\ -zx & -zy & x^2 + y^2 \end{pmatrix} dm(x, y, z). \end{aligned}$$

As an example, to compute the inertia matrix of the thin rod of mass m about its center of mass, as shown in Figure 2.4, the first step is to define a coordinate system at the center of mass, as shown in Figure 2.4, where \hat{i} and \hat{j} are orthogonal unit vectors pointing parallel and perpendicular to the rod, respectively. The unit vector \hat{k} forms a right handed coordinate system and points out of the page.

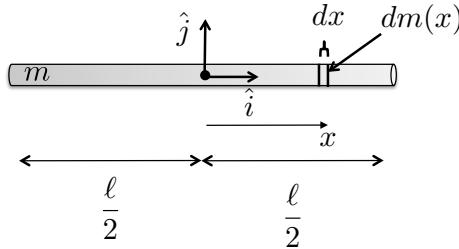


Figure 2.4: Computing the inertia of a thin rod.

Assuming that the rod is infinitely thin, and that the mass is evenly distributed throughout the rod, then the elemental mass is given by

$$dm(x) = \frac{m}{\ell} dx.$$

CHAPTER 2. KINETIC ENERGY OF MECHANICAL SYSTEMS

Since the rod is infinitely thin in the \hat{j} and \hat{k} directions, the inertia matrix is given by

$$\begin{aligned}
 J &= \int_{x=-\ell/2}^{\ell/2} \int_{y=0}^0 \int_{z=0}^0 \begin{pmatrix} y^2 + z^2 & -xy & -xz \\ -yx & x^2 + z^2 & -yz \\ -zx & -zy & x^2 + y^2 \end{pmatrix} dz dy \frac{m}{\ell} dx \\
 &= \frac{m}{\ell} \int_{x=-\ell/2}^{\ell/2} \begin{pmatrix} 0 & 0 & 0 \\ 0 & x^2 & 0 \\ 0 & 0 & x^2 \end{pmatrix} dx \\
 &= \frac{m}{\ell} \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{x^3}{3} \Big|_{x=-\frac{\ell}{2}}^{\frac{\ell}{2}} & 0 \\ 0 & 0 & \frac{x^3}{3} \Big|_{x=-\frac{\ell}{2}}^{\frac{\ell}{2}} \end{pmatrix} \\
 &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{m\ell^2}{12} & 0 \\ 0 & 0 & \frac{m\ell^2}{12} \end{pmatrix}. \tag{2.2}
 \end{aligned}$$

Note that the infinitely thin rod assumption implies that there is only inertia, or rotational mass, about the \hat{j} and \hat{k} axis.

As another example, to compute the inertia matrix of the thin plate of mass m about its center of mass, as shown in Figure 2.5, the first step is to define a coordinate system at the center of mass, as shown in Figure 2.5, where \hat{i} and \hat{j} are orthogonal unit vectors pointing along the length and width of the plate, respectively, and \hat{k} forms a right handed coordinate system and points perpendicular to the plate.

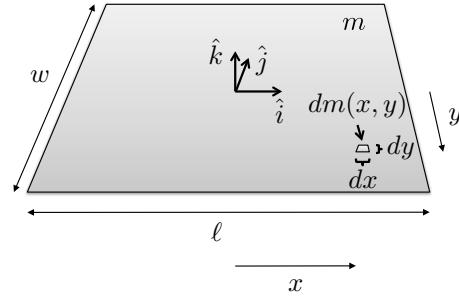


Figure 2.5: Computing the inertia of a thin plate.

Assuming that the plate is infinitely thin, and that the mass is evenly distributed throughout the plate, then the elemental mass is given by

$$dm(x, y) = \frac{m}{\ell w} dx dy.$$

Since the plate is infinitely thin in the \hat{k} direction, the inertia matrix is given

CHAPTER 2. KINETIC ENERGY OF MECHANICAL SYSTEMS

by

$$\begin{aligned}
J &= \int_{x=-\ell/2}^{\ell/2} \int_{y=-w/2}^{w/2} \int_{z=0}^0 \begin{pmatrix} y^2 + z^2 & -xy & -xz \\ -yx & x^2 + z^2 & -yz \\ -zx & -zy & x^2 + y^2 \end{pmatrix} dz dy \frac{m}{\ell w} dy dx \\
&= \frac{m}{\ell w} \int_{x=-\ell/2}^{\ell/2} \int_{y=-w/2}^{w/2} \begin{pmatrix} y^2 & -xy & 0 \\ -yx & x^2 & 0 \\ 0 & 0 & x^2 + y^2 \end{pmatrix} dy dx \\
&= \frac{m}{\ell w} \int_{x=-\ell/2}^{\ell/2} \begin{pmatrix} \frac{y^3}{3} \Big|_{y=-w/2}^{w/2} & -x \frac{y^2}{2} \Big|_{y=-w/2}^{w/2} & 0 \\ -\frac{y^2}{2} x \Big|_{y=-w/2}^{w/2} & x^2 y \Big|_{y=-w/2}^{w/2} & 0 \\ 0 & 0 & x^2 y \Big|_{y=-w/2}^{w/2} + \frac{y^3}{3} \Big|_{y=-w/2}^{w/2} \end{pmatrix} dx \\
&= \frac{m}{\ell w} \int_{x=-\ell/2}^{\ell/2} \begin{pmatrix} \frac{w^3}{12} & 0 & 0 \\ 0 & x^2 w & 0 \\ 0 & 0 & x^2 w + \frac{w^3}{12} \end{pmatrix} dx \\
&= \frac{m}{\ell w} \begin{pmatrix} \frac{w^3}{12} x \Big|_{x=-\ell/2}^{\ell/2} & 0 & 0 \\ 0 & \frac{x^3}{3} w \Big|_{x=-\ell/2}^{\ell/2} & 0 \\ 0 & 0 & \frac{x^3}{3} w \Big|_{x=-\ell/2}^{\ell/2} + \frac{w^3}{12} x \Big|_{x=-\ell/2}^{\ell/2} \end{pmatrix} dx \\
&= \frac{m}{\ell w} \begin{pmatrix} \frac{w^3}{12} \ell & 0 & 0 \\ 0 & \frac{\ell^3}{12} w & 0 \\ 0 & 0 & \frac{\ell^3}{12} w + \frac{w^3}{12} \ell \end{pmatrix} \\
&= \begin{pmatrix} \frac{mw^2}{12} & 0 & 0 \\ 0 & \frac{m\ell^2}{12} & 0 \\ 0 & 0 & \frac{m\ell^2}{12} + \frac{mw^2}{12} \end{pmatrix}.
\end{aligned}$$

The diagonal elements of J are called the moments of inertia and represent the rotational mass about the \hat{i} , \hat{j} , and \hat{k} axes respectively.

We have shown that if the motion of the system of N masses is caused purely by spinning about the angular velocity vector $\boldsymbol{\omega}$, that the kinetic energy is given by $K = \frac{1}{2}\boldsymbol{\omega}^\top J\boldsymbol{\omega}$. If the system of mass is also translating with velocity \mathbf{v} , then the velocity of the i^{th} particle is given by

$$\mathbf{v}_i = \mathbf{v} + \boldsymbol{\omega} \times \mathbf{p}_i.$$

CHAPTER 2. KINETIC ENERGY OF MECHANICAL SYSTEMS

In that case the kinetic energy is given by

$$\begin{aligned}
K &= \frac{1}{2} \sum_{i=1}^N m_i \mathbf{v}_i^\top \mathbf{v}_i \\
&= \frac{1}{2} \sum_{i=1}^N m_i (\mathbf{v} + \boldsymbol{\omega} \times \mathbf{p}_i)^\top (\mathbf{v} + \boldsymbol{\omega} \times \mathbf{p}_i) \\
&= \frac{1}{2} \sum_{i=1}^N m_i (\mathbf{v}^\top \mathbf{v} + \mathbf{v}^\top (\boldsymbol{\omega} \times \mathbf{p}_i) + (\boldsymbol{\omega} \times \mathbf{p}_i)^\top \mathbf{v} + (\boldsymbol{\omega} \times \mathbf{p}_i)^\top (\boldsymbol{\omega} \times \mathbf{p}_i)) \\
&= \frac{1}{2} \left(\sum_{i=1}^N m_i \right) \mathbf{v}^\top \mathbf{v} + \frac{1}{2} \sum_{i=1}^N m_i (\boldsymbol{\omega} \times \mathbf{p}_i)^\top (\boldsymbol{\omega} \times \mathbf{p}_i) \\
&\quad + \frac{1}{2} \sum_{i=1}^N m_i [\mathbf{v}^\top (\boldsymbol{\omega} \times \mathbf{p}_i) + (\boldsymbol{\omega} \times \mathbf{p}_i)^\top \mathbf{v}].
\end{aligned}$$

Using the cross product property

$$\mathbf{a}^\top (\mathbf{b} \times \mathbf{c}) = \mathbf{b}^\top (\mathbf{c} \times \mathbf{a}) = \mathbf{c}^\top (\mathbf{a} \times \mathbf{b})$$

we get that

$$\sum_{i=1}^N m_i [\mathbf{v}^\top (\boldsymbol{\omega} \times \mathbf{p}_i) + (\boldsymbol{\omega} \times \mathbf{p}_i)^\top \mathbf{v}] = \left(\sum_{i=1}^N m_i \mathbf{p}_i \right)^\top (\mathbf{v} \times \boldsymbol{\omega}) + \left(\sum_{i=1}^N m_i \mathbf{p}_i \right)^\top (\mathbf{v} \times \boldsymbol{\omega}).$$

If we choose the center of the coordinate system to be the center of mass, then $\sum_{i=1}^N m_i \mathbf{p}_i = 0$. Defining the total mass as $m = \sum_{i=1}^N m_i$, the kinetic energy is given by

$$\begin{aligned}
K &= \frac{1}{2} \left(\sum_{i=1}^N m_i \right) \mathbf{v}_{cm}^\top \mathbf{v}_{cm} + \frac{1}{2} \sum_{i=1}^N m_i (\boldsymbol{\omega} \times \mathbf{p}_i)^\top (\boldsymbol{\omega} \times \mathbf{p}_i) \\
&= \frac{1}{2} m \mathbf{v}_{cm}^\top \mathbf{v}_{cm} + \frac{1}{2} \boldsymbol{\omega}^\top J_{cm} \boldsymbol{\omega}.
\end{aligned}$$

Therefore, the kinetic energy of a rigid body moving with velocity \mathbf{v}_{cm} and spinning at angular velocity $\boldsymbol{\omega}$ is therefore

$$K = \frac{1}{2} m \mathbf{v}_{cm}^\top \mathbf{v}_{cm} + \frac{1}{2} \boldsymbol{\omega}^\top J_{cm} \boldsymbol{\omega}. \quad (2.3)$$

If the system consists of n rigid bodies, then the total kinetic energy is computed by summing the kinetic energy for each body as

$$K = \sum_{j=1}^n \left[\frac{1}{2} m_j \mathbf{v}_{cm,j}^\top \mathbf{v}_{cm,j} + \frac{1}{2} \boldsymbol{\omega}_j^\top J_{cm,j} \boldsymbol{\omega}_j \right].$$

2.1.1 Example: Mass Spring System

As an example, consider the mass spring system shown in Figure 2.6 where two masses are moving along a surface and are connected by springs with constants k_1 and k_2 and dampers with constants b_1 and b_2 . The kinetic energy of the system is determined by the velocity of the two masses and is therefore given by

$$K = \frac{1}{2}m_1\dot{z}_1^2 + \frac{1}{2}m_2\dot{z}_2^2.$$

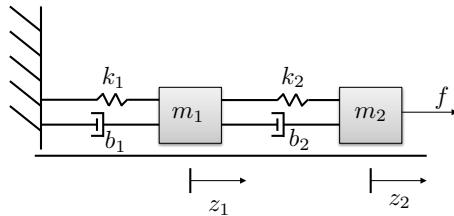


Figure 2.6: Mass spring system with two masses connected by springs and dampers.

2.1.2 Example: Spinning Dumbbell

As another simple example, consider the spinning dumbbell system shown in Figure 2.7, where two masses are spinning about a point on the adjoining line between the masses. The angular velocity vector is pointing out of the page with magnitude $\dot{\theta}$ and the dumbbell system is moving to the right at velocity \dot{z} .

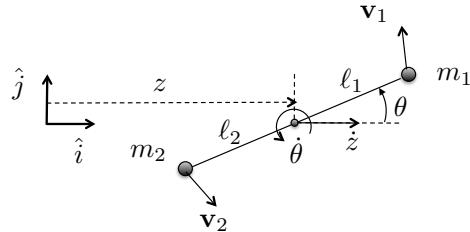


Figure 2.7: Spinning dumbbell. The system is spinning about the pointing out of the page and moving to the right at velocity \dot{z} .

The first step in computing the kinetic energy is to define an inertial coordinate system as shown in Figure 2.7 where \hat{i} is a unit vector along the direction of travel, \hat{j} is perpendicular to the direction of travel, and where \hat{k} forms a right handed coordinate system and points out of the page. In this coordinate system, the position of the point of rotation is given by $\mathbf{p}(t) = (z(t), 0, 0)^\top$. Accordingly,

CHAPTER 2. KINETIC ENERGY OF MECHANICAL SYSTEMS

the positions of m_1 and m_2 are given by

$$\begin{aligned}\mathbf{p}_1(t) &= (z(t) + \ell_1 \cos \theta(t), \ell_1 \sin \theta, 0)^\top, \\ \mathbf{p}_2(t) &= (z(t) - \ell_2 \cos \theta(t), -\ell_2 \sin \theta, 0)^\top.\end{aligned}$$

Since z and θ are functions of time, using the chain rule

$$\frac{d}{dt} f(g(t)) = \frac{\partial f}{\partial x} \frac{dg}{dt}$$

and differentiating with respect to time, we get the velocities of the masses as

$$\begin{aligned}\mathbf{v}_1 &= (\dot{z} - \ell_1 \dot{\theta} \sin \theta, \ell_1 \dot{\theta} \cos \theta, 0)^\top \\ \mathbf{v}_2 &= (\dot{z} + \ell_2 \dot{\theta} \sin \theta, -\ell_2 \dot{\theta} \cos \theta, 0)^\top.\end{aligned}$$

The kinetic energy is therefore given by

$$\begin{aligned}K &= \frac{1}{2} m_1 \mathbf{v}_1^\top \mathbf{v}_1 + \frac{1}{2} m_2 \mathbf{v}_2^\top \mathbf{v}_2 \\ &= \frac{1}{2} m_1 [(\dot{z} - \ell_1 \dot{\theta} \sin \theta)^2 + (\ell_1 \dot{\theta} \cos \theta)^2] + \frac{1}{2} m_2 [(\dot{z} + \ell_2 \dot{\theta} \sin \theta)^2 + (-\ell_2 \dot{\theta} \cos \theta)^2] \\ &= \frac{1}{2} m_1 [(\dot{z}^2 - 2\ell_1 \dot{z} \dot{\theta} \sin \theta + \ell_1^2 \dot{\theta}^2 \sin^2 \theta + \ell_1^2 \dot{\theta}^2 \cos^2 \theta)^2] \\ &\quad + \frac{1}{2} m_2 [(\dot{z}^2 + 2\ell_2 \dot{z} \dot{\theta} \sin \theta + \ell_2^2 \dot{\theta}^2 \sin^2 \theta + \ell_2^2 \dot{\theta}^2 \cos^2 \theta)^2] \\ &= \frac{1}{2} (m_1 + m_2) \dot{z}^2 + \frac{1}{2} (m_1 \ell_1^2 + m_2 \ell_2^2) \dot{\theta}^2 + (m_2 \ell_2 - m_1 \ell_1) \dot{z} \dot{\theta} \sin \theta.\end{aligned}$$

Note that if the pivot point is located at the center of mass, then $m_1 \ell_1 = m_2 \ell_2$, and the kinetic energy is simply given by

$$K = \frac{1}{2} (m_1 + m_2) \dot{z}^2 + \frac{1}{2} (m_1 \ell_1^2 + m_2 \ell_2^2) \dot{\theta}^2,$$

which is in the form

$$K = \frac{1}{2} m \mathbf{v}_{cm}^\top \mathbf{v}_{cm} + \frac{1}{2} \boldsymbol{\omega}^\top J \boldsymbol{\omega},$$

where the total mass is $m = m_1 + m_2$, the velocity of the center of mass is $\mathbf{v}_{cm} = (\dot{z}, 0, 0)^\top$, the angular velocity vector is $\boldsymbol{\omega} = (0, 0, \dot{\theta})^\top$ and the (3,3) element of J is given by $J_z = m_1 \ell_1^2 + m_2 \ell_2^2$.

2.2 Design Study A: Single Link Robot Arm

A definition of the single link robot arm is given in Design Study A.



Homework Problem A.2

Using the configuration variable θ , write an expression for the kinetic energy of the system.

Solution

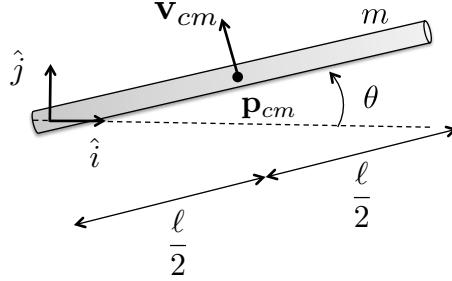


Figure 2.8: Single link robot arm. Compute the velocity of the center of mass and the angular velocity about the center of mass.

The first step is to define an inertial coordinate system as shown in Figure 2.8. If the pivot point is the origin of the coordinate system as shown in Figure 2.8, then the position of the center of mass is

$$\mathbf{p}_{cm}(t) = \begin{pmatrix} \frac{\ell}{2} \cos \theta(t) \\ \frac{\ell}{2} \sin \theta(t) \\ 0 \end{pmatrix}.$$

Differentiating to obtain the velocity we get

$$\mathbf{v}_{cm} = \frac{\ell}{2} \dot{\theta} \begin{pmatrix} -\sin \theta \\ \cos \theta \\ 0 \end{pmatrix}.$$

The angular velocity about the center of mass is given by

$$\boldsymbol{\omega} = \begin{pmatrix} 0 \\ 0 \\ \dot{\theta} \end{pmatrix}.$$

Observe that

$$\boldsymbol{\omega}^\top J \boldsymbol{\omega} = (0 \ 0 \ \dot{\theta}) \begin{pmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \dot{\theta} \end{pmatrix} = J_z \dot{\theta}^2,$$

therefore we only need the moment of inertia about the \hat{k} axis J_z . From the inertia matrix for a thin rod given in Equation (2.2), we have that $J_z = \frac{m\ell^2}{12}$.

Therefore, the kinetic energy of the single link robot arm is

$$\begin{aligned} K &= \frac{1}{2} m \mathbf{v}_{cm}^\top \mathbf{v}_{cm} + \frac{1}{2} \boldsymbol{\omega}^\top J \boldsymbol{\omega} \\ &= \frac{1}{2} m \frac{\ell^2}{4} \dot{\theta}^2 (\sin^2 \theta + \cos^2 \theta) + \frac{1}{2} \dot{\theta}^2 \frac{m\ell^2}{12} \\ &= \frac{1}{2} \frac{m\ell^2}{3} \dot{\theta}^2. \end{aligned}$$

2.3 Design Study B. Pendulum on Cart

A definition of the inverted pendulum is given in Design Study B



Homework Problem B.2

Using the configuration variables z and θ , write an expression for the kinetic energy of the system.

Solution

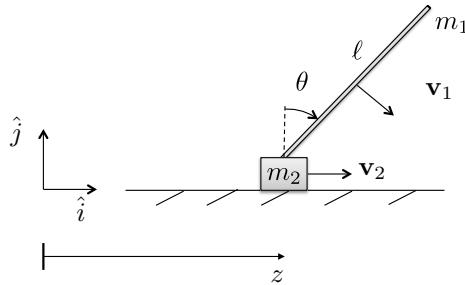


Figure 2.9: Pendulum on a cart. Compute the velocity of the bob and of the cart

Define the inertial coordinate frame as in Figure 2.9, with \hat{k} out of the page. The position of the center of mass of the rod with mass m_1 is given by

$$\mathbf{p}_1 = \begin{pmatrix} z(t) + \frac{\ell}{2} \sin \theta(t) \\ \frac{\ell}{2} \cos \theta(t) \\ 0 \end{pmatrix},$$

and the horizontal position of the cart with mass m_2 is given by

$$\mathbf{p}_2 = \begin{pmatrix} z(t) \\ 0 \\ 0 \end{pmatrix}.$$

Differentiating to obtain the velocities of m_1 and m_2 we obtain

$$\mathbf{v}_1 = \begin{pmatrix} \dot{z} + \frac{\ell}{2} \dot{\theta} \cos \theta \\ -\frac{\ell}{2} \dot{\theta} \sin \theta \\ 0 \end{pmatrix}, \quad \mathbf{v}_2 = \begin{pmatrix} \dot{z} \\ 0 \\ 0 \end{pmatrix}.$$

Since we are modeling the cart and the bob as point masses and the rod as massless, there is no associated moments of inertia. Therefore the kinetic energy

of the system is given by

$$\begin{aligned}
 K &= \frac{1}{2}m_1\mathbf{v}_1^\top\mathbf{v}_1 + \frac{1}{2}m_2\mathbf{v}_2^\top\mathbf{v}_2 \\
 &= \frac{1}{2}m_1\left[(\dot{z} + \frac{\ell}{2}\dot{\theta}\cos\theta)^2 + (-\frac{\ell}{2}\dot{\theta}\sin\theta)^2\right] + \frac{1}{2}m_2\dot{z}^2 \\
 &= \frac{1}{2}m_1\left[\dot{z}^2 + 2\frac{\ell}{2}\dot{z}\dot{\theta}\cos\theta + \frac{\ell^2}{4}\dot{\theta}^2\cos^2\theta + \frac{\ell^2}{4}\dot{\theta}^2\sin^2\theta\right] + \frac{1}{2}m_2\dot{z}^2 \\
 &= \frac{1}{2}(m_1 + m_2)\dot{z}^2 + \frac{1}{2}m_1\frac{\ell^2}{4}\dot{\theta}^2 + m_1\frac{\ell}{2}\dot{z}\dot{\theta}\cos\theta. \tag{2.4}
 \end{aligned}$$

2.4 Design Study C. Satellite Attitude Control

A definition of the satellite attitude control problem is given in Design Study [C](#)



Homework Problem C.2

Using the configuration variables θ and ϕ , write an expression for the kinetic energy of the system.

Solution

Since the satellite consists of two rotational masses, the kinetic energy is the sum of the rotational kinetic energy of each mass. Therefore the kinetic energy is given by

$$K = \frac{1}{2}J_s\dot{\theta}^2 + \frac{1}{2}J_p\dot{\phi}^2. \tag{2.5}$$

Notes and References

A table that lists the moment of inertia about certain axes, and the inertia matrix for several simple shapes can be found at [Wikipedia-Lists of moments of inertia](#).

The inertia matrix in Equation (2.3) is computed about the center of mass of the object. When an object is composed of several rigid bodies connected to each other, where the inertia matrix may be known for each rigid body. To find the total inertia about the center of mass, it is convenient to use the parallel axis theorem which states that if J_{cm} is the inertia matrix about the center of mass, then the inertia matrix about a point that is located at vector \mathbf{r} from the center of mass is

$$J(\mathbf{r}) = J_{cm} + m(\mathbf{r}^\top\mathbf{r}I_3 - \mathbf{r}\mathbf{r}^\top).$$

For the moment of inertia about a single axis, where the axis is shifted from the center of mass by distance d , the parallel axis theorem implies that

$$I(d) = I_{cm} + md^2.$$

CHAPTER 2. KINETIC ENERGY OF MECHANICAL SYSTEMS

Additional information about the parallel axis theorem can be found at [Wikipedia-Parallel axis theorem](#).

Chapter 3

The Euler Lagrange Equations

3.1 Theory

In this chapter we show how to define the potential energy for simple mechanical systems, and how to define the generalized forces. The Euler-Lagrange equations given in Equation (1.8) can then be used to derive the equations of motion for the system.

3.1.1 Potential Energy

In this book we will be concerned with two sources of potential energy: potential energy due to gravity, and potential energy due to springs. Consider a point

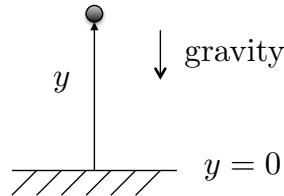


Figure 3.1: The potential energy of a unit mass m in gravity is given by $P = mgy$.

mass m in a gravity field that is at position y as shown in Figure 3.1. The potential energy is given by

$$P = mgy + P_0,$$

where P_0 is the potential energy when $y = 0$. Typically we desire an expression for potential energy that is zero when the configuration variables are zero.

The second source of potential energy is from translational and rotational springs. The spring shown in Figure 3.2 is stretched from rest by a distance of

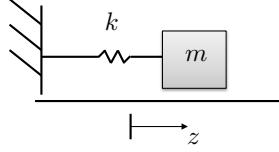


Figure 3.2: The potential energy of spring stretched by distance z is $P = \frac{1}{2}kz^2$.

z . The potential energy of a linear stretched/compressed spring is given by

$$P = \frac{1}{2}kz^2,$$

where k is the spring constant. Similarly, for a rotational spring, the potential energy is given by $P = \frac{1}{2}k\theta^2$, where θ is the rotational angle and $\theta = 0$ is the angle at which the rotational spring is at rest.

3.1.2 Generalized Coordinates and Generalized Forces

The *generalized coordinates* are the minimum set of configuration variables. For example, for the mass-spring-damper system shown in Figure 2.6 the generalized coordinates are $\mathbf{q} = (z_1, z_2)^\top$, for the spinning dumbbell system shown in Figure 2.7, and the single link robot arm shown in Figure 2.8 the generalized coordinate is $\mathbf{q} = \theta$, for the pendulum on a cart system shown in Figure 2.9 the generalized coordinates are $\mathbf{q} = (z, \theta)^\top$, and for the satellite system shown in Figure 1.4 the generalized coordinates are $\mathbf{q} = (\theta, \phi)^\top$.

The *generalized forces* are the nonconservative forces and torques acting along each generalized coordinate. In other words, they are the applied forces and torques on the system. For example, for the mass-spring-damper system shown in Figure 2.6, the generalized force acting along z_1 is zero, and the generalized force acting along z_2 is the applied force F . Therefore the generalized force is $\boldsymbol{\tau} = (0 \ F)^\top$.

3.1.3 Damping Forces and Torques

Friction and applied dampers also exert forces on the system. Damping forces are externally applied forces that are proportional to the velocity or angular velocity of the system. For example, in the mass-spring-damper system shown in Figure 2.6, the damping force acting along z_1 is due to the first damper and the damper between the two masses and is given by $-b_1\dot{z}_1 - b_2(\dot{z}_1 - \dot{z}_2)$. The damping acting along z_2 is given by $-b_2(\dot{z}_2 - \dot{z}_1)$. Therefore the damping force on the system is given by

$$-B\dot{\mathbf{q}} = -\begin{pmatrix} b_1 + b_2 & -b_2 \\ -b_2 & b_2 \end{pmatrix} \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} -b_1\dot{z}_1 - b_2(\dot{z}_1 - \dot{z}_2) \\ -b_2(\dot{z}_2 - \dot{z}_1) \end{pmatrix}.$$

3.1.4 Euler Lagrange Equations

The equations of motion are given by the Euler-Lagrange equations. Let the Lagrangian be given by

$$L(\mathbf{q}, \dot{\mathbf{q}}) = K(\mathbf{q}, \dot{\mathbf{q}}) - P(\mathbf{q}),$$

where $K(\mathbf{q}, \dot{\mathbf{q}})$ is the kinetic energy of the system and is, in general, a function of the generalized coordinates and the generalized velocities, and $P(\mathbf{q})$ is the potential energy which is, in general, a function of the generalized coordinates. As shown in Equation (1.8) the Euler-Lagrange equations are given in vector form as

$$\frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}} \right) - \frac{\partial L(\mathbf{q})}{\partial \mathbf{q}} = \boldsymbol{\tau} - B\dot{\mathbf{q}}.$$

3.2 Design Study A. Single Link Robot Arm



Homework Problem A.3

- (a) Find the potential energy for the system.
- (b) Define the generalized coordinates.
- (c) Find the generalized forces and damping forces.
- (d) Derive the equations of motion using the Euler-Lagrange equations.

Solution

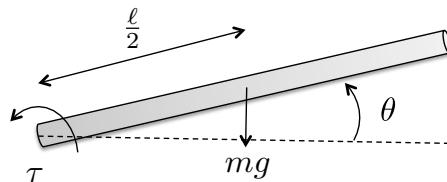


Figure 3.3: Energy calculation for the single link robot arm

Let P_0 be the potential when $\theta = 0$. The height of the center of mass is $\ell/2 \sin \theta$, which is zero when $\theta = 0$. Accordingly, the potential energy is given by

$$P = P_0 + mg \frac{\ell}{2} \sin \theta.$$

The generalized coordinate is

$$q_1 = \theta.$$

CHAPTER 3. THE EULER LAGRANGE EQUATIONS

The generalized force is

$$\tau_1 = \tau,$$

and the generalized damping is

$$-B\dot{\mathbf{q}} = -b\dot{\theta}.$$

From Homework A.1 we found that the kinetic energy is given by

$$K = \frac{1}{2} \left(\frac{m\ell^2}{3} \right) \dot{\theta}^2.$$

Therefore the Lagrangian is

$$L = K - P = \frac{1}{2} \left(\frac{m\ell^2}{3} \right) \dot{\theta}^2 - P_0 - mg \frac{\ell}{2} \sin \theta.$$

For this case, the Euler-Lagrange equation is given by

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} = \tau_1 - b\dot{\theta}$$

where

$$\begin{aligned} \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) &= \frac{d}{dt} \left(\frac{m\ell^3}{3} \dot{\theta} \right) = \frac{m\ell^3}{3} \ddot{\theta} \\ \frac{\partial L}{\partial \theta} &= -mg \frac{\ell}{2} \cos \theta. \end{aligned}$$

Therefore, the Euler Lagrange equation becomes

$$\frac{m\ell^2}{3} \ddot{\theta} + mg \frac{\ell}{2} \cos \theta = \tau - b\dot{\theta}. \quad (3.1)$$

3.3 Design Study B. Inverted Pendulum



Homework Problem B.3

- (a) Find the potential energy for the system.
- (b) Define the generalized coordinates.
- (c) Find the generalized forces and damping forces.
- (d) Derive the equations of motion using the Euler-Lagrange equations.

Solution

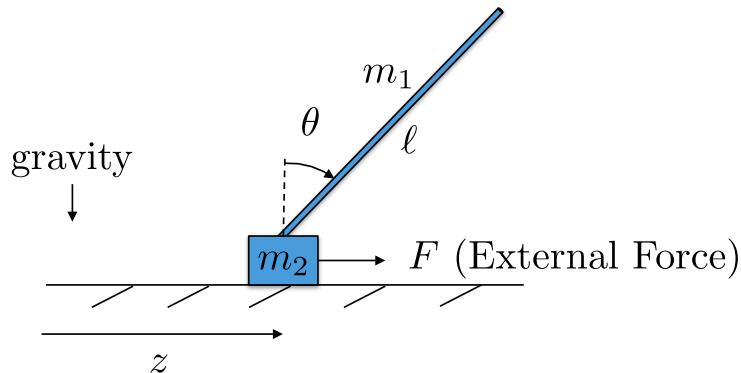


Figure 3.4: Pendulum on a cart.

Let P_0 be the potential energy when $\theta = 0$ and $z = 0$. Then the potential energy of the pendulum system is given by

$$P = P_0 + m_1 g \frac{\ell}{2} \cos \theta,$$

where $\ell \cos \theta / 2$ is the height of the center of mass of the rod. Note that the potential energy decreases as θ increases.

The generalized coordinates for the system are the horizontal position z and the angle θ . Therefore, let $\mathbf{q} = (z, \theta)^\top$.

The external forces acting in the direction of z is $\tau_1 = F$, and the external torque acting in the direction of θ is $\tau_2 = 0$. Therefore, the generalized forces are $\boldsymbol{\tau} = (F, 0)^\top$. A damping term acts in the direction of z , which implies that $-B\dot{\mathbf{q}} = (-b\dot{z}, 0)^\top$.

CHAPTER 3. THE EULER LAGRANGE EQUATIONS

Using Equation (2.4), the kinetic energy can be written in terms of the generalized coordinates as

$$\begin{aligned} K(\mathbf{q}, \dot{\mathbf{q}}) &= \frac{1}{2}(m_1 + m_2)\dot{z}^2 + \frac{1}{2}m_1\frac{\ell^2}{4}\dot{\theta}^2 + m_1\frac{\ell}{2}\dot{z}\dot{\theta}\cos\theta \\ &= \frac{1}{2}(m_1 + m_2)\dot{q}_1^2 + \frac{1}{2}m_1\frac{\ell^2}{4}\dot{q}_2^2 + m_1\frac{\ell}{2}\dot{q}_1\dot{q}_2\cos q_2, \end{aligned}$$

and the potential energy can be written as

$$\begin{aligned} P(\mathbf{q}) &= P_0 + m_1g\frac{\ell}{2}\cos\theta \\ &= P_0 + m_1g\frac{\ell}{2}\cos q_2. \end{aligned}$$

The Lagrangian is therefore given by

$$L(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}(m_1 + m_2)\dot{q}_1^2 + \frac{1}{2}m_1\frac{\ell^2}{4}\dot{q}_2^2 + m_1\frac{\ell}{2}\dot{q}_1\dot{q}_2\cos q_2 - P_0 - m_1g\frac{\ell}{2}\cos q_2.$$

Therefore

$$\begin{aligned} \frac{\partial L}{\partial \dot{\mathbf{q}}} &= \begin{pmatrix} (m_1 + m_2)\dot{z} + m_1\frac{\ell}{2}\dot{\theta}\cos\theta \\ m_1\frac{\ell^2}{4}\dot{\theta} - m_1\frac{\ell}{2}\dot{z}\cos\theta \end{pmatrix} \\ \frac{\partial L}{\partial \mathbf{q}} &= \begin{pmatrix} 0 \\ -m_1\frac{\ell}{2}\dot{z}\dot{\theta}\sin\theta + m_1g\frac{\ell}{2}\sin\theta \end{pmatrix}. \end{aligned}$$

Differentiating $\frac{\partial L}{\partial \dot{\mathbf{q}}}$ with respect to time gives

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{q}}} \right) = \begin{pmatrix} (m_1 + m_2)\ddot{z} + m_1\frac{\ell}{2}\ddot{\theta}\cos\theta - m_1\frac{\ell}{2}\dot{\theta}^2\sin\theta \\ m_1\frac{\ell^2}{4}\ddot{\theta} + m_1\frac{\ell}{2}\ddot{z}\cos\theta - m_1\frac{\ell}{2}\dot{z}\dot{\theta}\sin\theta \end{pmatrix}.$$

Therefore the Euler-Lagrange equation

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{q}}} \right) - \frac{\partial L}{\partial \mathbf{q}} = \boldsymbol{\tau} - B\dot{\mathbf{q}}$$

gives

$$\begin{pmatrix} (m_1 + m_2)\ddot{z} + m_1\frac{\ell}{2}\ddot{\theta}\cos\theta - m_1\frac{\ell}{2}\dot{\theta}^2\sin\theta \\ m_1\frac{\ell^2}{4}\ddot{\theta} + m_1\frac{\ell}{2}\ddot{z}\cos\theta - m_1\frac{\ell}{2}\dot{z}\dot{\theta}\sin\theta + m_1\frac{\ell}{2}\dot{z}\dot{\theta}\sin\theta - m_1g\frac{\ell}{2}\sin\theta \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix} - \begin{pmatrix} b\dot{z} \\ 0 \end{pmatrix}.$$

Simplifying and moving all second order derivatives to the left and side, and all other terms to the right hand side gives

$$\begin{pmatrix} (m_1 + m_2)\ddot{z} + m_1\frac{\ell}{2}\ddot{\theta}\cos\theta \\ m_1\frac{\ell^2}{4}\ddot{\theta} + m_1\frac{\ell}{2}\ddot{z}\cos\theta \end{pmatrix} = \begin{pmatrix} m_1\frac{\ell}{2}\dot{\theta}^2\sin\theta + F - b\dot{z} \\ m_1g\frac{\ell}{2}\sin\theta \end{pmatrix}.$$

CHAPTER 3. THE EULER LAGRANGE EQUATIONS

Using matrix notation, this equation can be rearranged to isolate the second order derivatives on the left and side

$$\begin{pmatrix} (m_1 + m_2) & m_1 \frac{\ell}{2} \cos \theta \\ m_1 \frac{\ell}{2} \cos \theta & m_1 \frac{\ell^2}{4} \end{pmatrix} \begin{pmatrix} \ddot{z} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} m_1 \frac{\ell}{2} \dot{\theta}^2 \sin \theta + F - b\dot{z} \\ m_1 g \frac{\ell}{2} \sin \theta \end{pmatrix}. \quad (3.2)$$

Equation (3.2) represents the simulation model for the pendulum on a cart system.

3.4 Design Study C. Satellite Attitude Control



Homework Problem C.3

- (a) Find the potential energy for the system.
- (b) Define the generalized coordinates.
- (c) Find the generalized forces and damping forces.
- (d) Derive the equations of motion using the Euler-Lagrange equations.

Solution

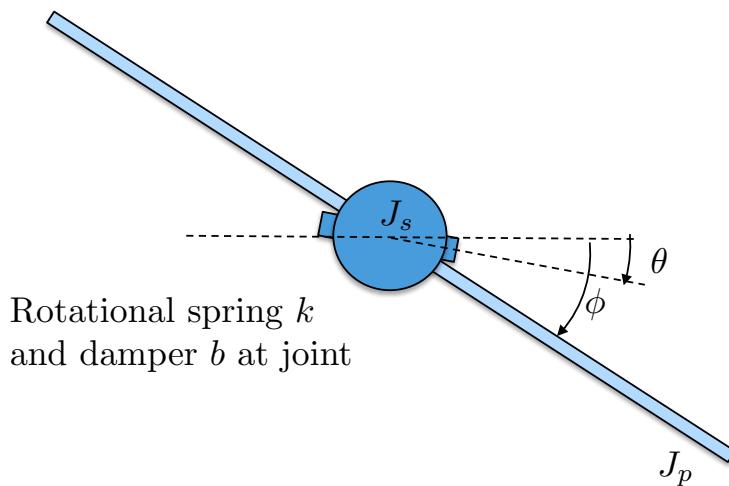


Figure 3.5: Satellite with flexible solar panels.

The generalized coordinates for the satellite attitude control problem are the angles θ and ϕ . Therefore, we define the generalized coordinates as $\mathbf{q} = (\theta, \phi)^\top$. We will assume the ability to apply torque on the main body (θ) of the satellite, but not on the solar panel. Therefore, the generalized force is $\boldsymbol{\tau} = (\tau, 0)^\top$. We will model the dissipation (friction) forces as proportional to the relative motion between the main body and the solar panel. Therefore, the damping term is given by

$$-B\dot{\mathbf{q}} = \begin{pmatrix} -b(\dot{\theta} - \dot{\phi}) \\ -b(\dot{\phi} - \dot{\theta}) \end{pmatrix} = - \begin{pmatrix} b & -b \\ -b & b \end{pmatrix} \begin{pmatrix} \dot{\theta} \\ \dot{\phi} \end{pmatrix}.$$

CHAPTER 3. THE EULER LAGRANGE EQUATIONS

Using Equation (2.5), the kinetic energy can be written in terms of the generalized coordinates as

$$\begin{aligned} K(\mathbf{q}, \dot{\mathbf{q}}) &= \frac{1}{2} J_s \dot{\theta}^2 + \frac{1}{2} J_p \dot{\phi}^2 \\ &= \frac{1}{2} J_s \dot{q}_1^2 + \frac{1}{2} J_p \dot{q}_2^2. \end{aligned}$$

The potential energy of the system is due to the spring force between the base and the solar panel, and is given by

$$\begin{aligned} P(\mathbf{q}) &= \frac{1}{2} k(\phi - \theta)^2 \\ &= \frac{1}{2} k(q_2 - q_1)^2, \end{aligned}$$

where k is the spring constant. The Lagrangian is therefore given by

$$L(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2} J_s \dot{q}_1^2 + \frac{1}{2} J_p \dot{q}_2^2 - \frac{1}{2} k(q_2 - q_1)^2.$$

Therefore

$$\begin{aligned} \frac{\partial L}{\partial \dot{\mathbf{q}}} &= \begin{pmatrix} J_s \dot{\theta} \\ J_p \dot{\phi} \end{pmatrix} \\ \frac{\partial L}{\partial \mathbf{q}} &= \begin{pmatrix} k(\phi - \theta) \\ -k(\phi - \theta) \end{pmatrix}. \end{aligned}$$

Differentiating $\frac{\partial L}{\partial \dot{\mathbf{q}}}$ with respect to time gives

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{q}}} \right) = \begin{pmatrix} J_s \ddot{\theta} \\ J_p \ddot{\phi} \end{pmatrix}.$$

Therefore the Euler-Lagrange equation

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{q}}} \right) - \frac{\partial L}{\partial \mathbf{q}} = \boldsymbol{\tau} - B \dot{\mathbf{q}}$$

gives

$$\begin{pmatrix} J_s \ddot{\theta} - k(\phi - \theta) \\ J_p \ddot{\phi} + k(\phi - \theta) \end{pmatrix} = \begin{pmatrix} \tau \\ 0 \end{pmatrix} + \begin{pmatrix} -b(\dot{\theta} - \dot{\phi}) \\ -b(\dot{\phi} - \dot{\theta}) \end{pmatrix}.$$

Simplifying and moving all second order derivatives to the left and side, and all other terms to the right hand side gives

$$\begin{pmatrix} J_s \ddot{\theta} \\ J_p \ddot{\phi} \end{pmatrix} = \begin{pmatrix} \tau - b(\dot{\theta} - \dot{\phi}) + k(\theta - \phi) \\ -b(\dot{\phi} - \dot{\theta}) - k(\phi - \theta) \end{pmatrix}.$$

Using matrix notation, this equation can be rearranged to isolate the second order derivatives on the left and side

$$\begin{pmatrix} J_s & 0 \\ 0 & J_p \end{pmatrix} \begin{pmatrix} \ddot{\theta} \\ \ddot{\phi} \end{pmatrix} = \begin{pmatrix} \tau - b(\dot{\theta} - \dot{\phi}) + k(\theta - \phi) \\ -b(\dot{\phi} - \dot{\theta}) - k(\phi - \theta) \end{pmatrix}. \quad (3.3)$$

Equation (3.3) represents the simulation model for the simplified satellite system.

CHAPTER 3. THE EULER LAGRANGE EQUATIONS

Notes and References

Derivation of the Lagrange-Euler equations can be found in many sources including [1, 2]. A simple derivation using only calculus can be found in [3].

Part II

Design Models

The equations of motion that we derived in the previous section are often too complicated to facilitate effective engineering design. Accordingly, the standard approach is to simplify the models into so-called *design models* that capture the essential features of the system. The design techniques studied in this class will require that the design models are linear and time-invariant. In addition, the PID method described in Part III will require that the design models are second order systems.

Our approach to developing linear time-invariant design models will be to linearize the simulation models developed in Part I and then to convert the linearized models into two standard forms, namely

- Transfer function models, and
- State space models.

Both forms will have advantages and disadvantages that will be explained throughout the remaining parts of the book. In Chapter 4 we will define the important concept of equilibrium and show how the equations of motion can be linearized about the equilibrium. Chapter 5 will show how to transform the linearized equations of motion into transfer function models, and chapter 6 will show how to put the equations of motion in state space form.

The remaining parts of the book will use the design models to develop feed-back controllers for the system of interest. In Part III we will describe the Proportional-Integral-Derivative (PID) controllers based on second order transfer function models. Part IV derives observer based controllers using state space models. Finally, Part V describes loopshaping design strategies using general transfer function models.

Chapter 4

Equilibria and Linearization

4.1 Theory

In this chapter we will describe two methods for linearizing the system, namely Jacobian linearization, and feedback linearization. Jacobian linearization is based on the idea of approximating the nonlinear terms in the equations of motion by the first two terms in their Taylor series expansion. Feedback linearization is based on the idea of using the input signal to artificially remove the nonlinear terms.

4.1.1 Jacobian Linearization

If the function $g : \mathbb{R} \rightarrow \mathbb{R}$ is a nonlinear function that can be differentiated an infinite number of times, then its Taylor series expansion about the point w_0 is given by

$$g(w) = g(w_0) + \frac{\partial g}{\partial w} \Big|_{w_0} (w - w_0) + \frac{1}{2!} \frac{\partial^2 g}{\partial w^2} \Big|_{w_0} (w - w_0)^2 + \dots$$

A pictorial representation of Taylor series expansion is shown in Figure 4.1, where it can be seen that in a small neighborhood around $w = w_0$, the function $g(w)$ is well approximated by the first two term in the Taylor series expansion, namely

$$g(w) \approx g(w_0) + \frac{\partial g}{\partial w} \Big|_{w_0} (w - w_0).$$

CHAPTER 4. EQUILIBRIA AND LINEARIZATION

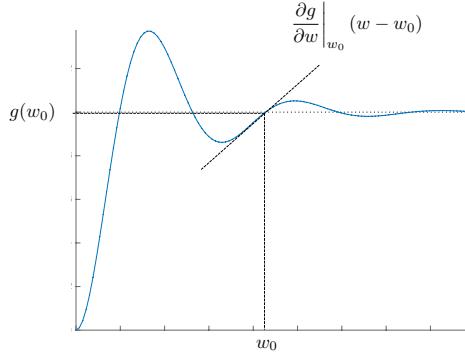


Figure 4.1: Taylor series expansion of $g(w)$ about $w = w_0$.

If g is a scalar function of several variables, then the Taylor series about $w_1 = w_{10}, w_2 = w_{20}, \dots, w_m = w_{m0}$ is given by

$$\begin{aligned} g(w_1, w_2, \dots, w_m) &= g(w_{10}, w_{20}, \dots, w_{m0}) + \frac{\partial g}{\partial w_1} \Big|_{w_{10}, w_{20}, \dots, w_{m0}} (w_1 - w_{10}) \\ &\quad + \frac{\partial g}{\partial w_2} \Big|_{w_{10}, w_{20}, \dots, w_{m0}} (w_2 - w_{20}) \\ &\quad + \cdots + \frac{\partial g}{\partial w_m} \Big|_{w_{10}, w_{20}, \dots, w_{m0}} (w_m - w_{m0}) + \cdots. \end{aligned} \quad (4.1)$$

For example, given the function $g(v, w) = v \cos w$, the constant and linear terms in the Taylor series, expanded around v_0, w_0 are

$$\begin{aligned} g(v, w) &\approx g(v_0, w_0) + \frac{\partial g}{\partial v} \Big|_{v_0, w_0} (v - v_0) + \frac{\partial g}{\partial w} \Big|_{v_0, w_0} (w - w_0) \\ &= v_0 \cos w_0 + (\cos w) \Big|_{v_0, w_0} (v - v_0) + (-v \sin w) \Big|_{v_0, w_0} (w - w_0) \\ &= v_0 \cos w_0 + (\cos w_0)(v - v_0) - (v_0 \sin w_0)(w - w_0). \end{aligned}$$

Since approximating a nonlinear function by the first two terms in its Taylor series is only a good approximation in a small neighborhood of w_0 , it is important to select w_0 judiciously. In particular, we will be interested in linearizing about equilibrium points of differential equations.

Definition. Given the differential equation $\dot{x} = f(x, u)$, the pair (x_e, u_e) is an *equilibrium point* if

$$f(x_e, u_e) = 0.$$

In other words, an equilibrium point is a state-input pair where there is not any motion in the system.

For example, suppose that the nonlinear differential equation of a second order system is given by

$$\ddot{y} + a\dot{y} + by + g(y, \dot{y}) = u, \quad (4.2)$$

CHAPTER 4. EQUILIBRIA AND LINEARIZATION

where a and b are constants and $g(y, \dot{y})$ is a known nonlinear function of y and \dot{y} , and u is the input signal. Defining $x = (y, \dot{y})^\top$, we get

$$\dot{x} = \begin{pmatrix} \dot{y} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \dot{y} \\ u - a\dot{y} - b y - g(y, \dot{y}) \end{pmatrix} \triangleq f(x, u).$$

The equilibrium is when $f(x_e, u_e) = 0$ or in other words when

$$y_e = \text{anything}, \quad \dot{y}_e = 0, \quad u_e = b y_e + g(y_e, 0).$$

Equivalently, at equilibrium there is no motion in the system, which implies that $\ddot{y}_e = \dot{y}_e = 0$, which from Equation (4.2) implies that $u_e = b y_e + g(y_e, 0)$.

Jacobian linearization then proceeds by replacing each term in the nonlinear differential equations describing the system, by the first two terms in the Taylor's series expansion about the equilibrium point.

For example, since the first term in Equation (4.2) is already linear, it can be expanded exactly as

$$\ddot{y} = \ddot{y}_e + \frac{\partial \ddot{y}}{\partial \dot{y}} \Big|_e (\ddot{y} - \ddot{y}_e) = \ddot{y},$$

where we have defined the linearized quantity $\tilde{y} \triangleq y - y_e$, and we have used the fact that $\ddot{y}_e = 0$. Similarly, the remaining terms in Equation (4.2) are linearized as

$$\begin{aligned} a\dot{y} &= a\dot{y}_e + a \frac{\partial \dot{y}}{\partial y} \Big|_e (\dot{y} - \dot{y}_e) = a\dot{\tilde{y}} \\ b y &= b y_e + b \frac{\partial y}{\partial \dot{y}} \Big|_e (y - y_e) = b y_e + b\tilde{y} \\ g(y, \dot{y}) &\approx g(y_e, \dot{y}_e) + \frac{\partial g}{\partial y} \Big|_e (y - y_e) + \frac{\partial g}{\partial \dot{y}} \Big|_e (\dot{y} - \dot{y}_e) = g(y_e, 0) + \frac{\partial g}{\partial y} \Big|_e \tilde{y} + \frac{\partial g}{\partial \dot{y}} \Big|_e \dot{\tilde{y}} \\ u &= u_e + \frac{\partial u}{\partial u} \Big|_e (u - u_e) = u_e + \tilde{u}. \end{aligned}$$

Substituting these expressions into Equation (4.2) the resulting linearized equations of motion are

$$[\ddot{\tilde{y}}] + [a\dot{\tilde{y}}] + [b y_e + b\tilde{y}] + \left[g(y_e, 0) + \frac{\partial g}{\partial y} \Big|_e \tilde{y} + \frac{\partial g}{\partial \dot{y}} \Big|_e \dot{\tilde{y}} \right] = [\tilde{u}]$$

Using the equilibrium $u_e = b y_e + g(y_e, 0)$ gives

$$\ddot{\tilde{y}} + \left(a + \frac{\partial g}{\partial \dot{y}} \Big|_e \right) \dot{\tilde{y}} + \left(b + \frac{\partial g}{\partial y} \Big|_e \right) \tilde{y} = \tilde{u}.$$

A block diagram that shows the design model for Equation (4.2) using Jacobian linearization is shown in Figure 4.2.

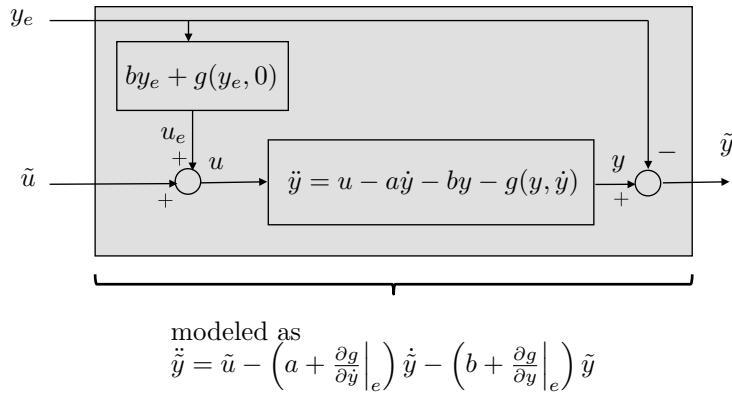


Figure 4.2: Linear model using Jacobian linearization

4.1.2 Feedback Linearization

It is often the case that the nonlinearities in the system can be directly canceled out by the judicious choice of the input variable. Consider again the nonlinear equations of motion in Equation (4.2). If u is selected as

$$u = g(y, \dot{y}) + \tilde{u}$$

then the resulting equations of motion are

$$\ddot{y} + a\dot{y} + by = \tilde{u},$$

which are linear. This technique is called *feedback linearization* because the feedback signal u has been used to linearize the system by directly canceling the nonlinearities. The advantage of this method is that the equations of motion are still globally defined and are not restricted to a small neighborhood of an equilibrium point. The disadvantage is that $g(y, \dot{y})$ may not be precisely known, and therefore may not be completely canceled by u . A block diagram that shows the design model for Equation (4.2) using feedback linearization is shown in Figure 4.3.

4.2 Design Study A. Single Link Robot Arm



Homework Problem A.4

For the single link robot arm:

- (a) Find the equilibria of the system.
- (b) Linearize the system about the equilibria using Jacobian linearization.
- (c) Linearize the system using feedback linearization.

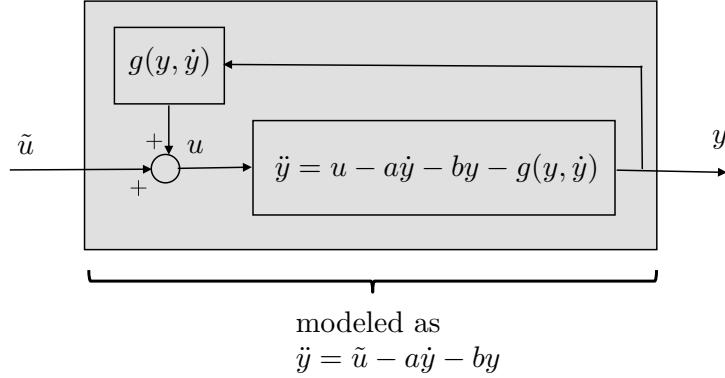


Figure 4.3: Linear model using feedback linearization

Solution

The differential equation describing the single link robot arm derived in HW A.3 is

$$\frac{m\ell^2}{3}\ddot{\theta} + \frac{mg\ell}{2}\cos\theta = \tau - b\dot{\theta}. \quad (4.3)$$

Defining $x = (\theta, \dot{\theta})$, and $u = \tau$ we get

$$\dot{x} = \begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \end{pmatrix} = \left(\frac{3}{m\ell^2}\tau - \frac{3b}{m\ell^2}\dot{\theta} - \frac{3g}{2\ell}\cos\theta \right) \triangleq f(x, u).$$

The equilibrium is when $f(x_e, u_e) = 0$, or in other words when

$$\theta_e = \text{anything}, \quad \dot{\theta}_e = 0, \quad \tau_e = \frac{mg\ell}{2}\cos\theta_e. \quad (4.4)$$

Equivalently, at equilibrium there is no motion in the system, which implies that $\ddot{\theta}_e = \dot{\theta}_e = 0$, which from Equation (4.3) implies that $\tau_e = \frac{mg\ell}{2}\cos\theta_e$.

Therefore, any pair (θ_e, τ_e) satisfying Equation (4.4) is an equilibria. Jacobian linearization then proceeds by replacing each term in the nonlinear differential equations describing the system, by the first two terms in the Taylor's series expansion about the equilibrium point. Defining $\tilde{\theta} \triangleq \theta - \theta_e$, $\tilde{\dot{\theta}} \triangleq \dot{\theta} - \dot{\theta}_e = \dot{\theta}$, $\tilde{\ddot{\theta}} = \ddot{\theta} - \ddot{\theta}_e = \ddot{\theta}$, and $\tilde{\tau} = \tau - \tau_e$, each term in Equation (4.3) can be expanded

CHAPTER 4. EQUILIBRIA AND LINEARIZATION

about the equilibrium as follows:

$$\begin{aligned} \frac{m\ell^2}{3}\ddot{\theta} &= \frac{m\ell^2}{3}\ddot{\theta}_e + \frac{m\ell^2}{3}\left.\frac{\partial\ddot{\theta}}{\partial\ddot{\theta}}\right|_e(\ddot{\theta} - \ddot{\theta}_e) = \frac{m\ell^2}{3}\tilde{\ddot{\theta}}, \\ \frac{mg\ell}{2}\cos\theta &\approx \frac{mg\ell}{2}\cos\theta_e + \frac{mg\ell}{2}\left.\frac{\partial}{\partial\theta}(\cos\theta)\right|_{\theta_e}(\theta - \theta_e) \\ &= \frac{mg\ell}{2}\cos\theta_e - \frac{mg\ell}{2}\sin\theta_e\tilde{\theta} \\ \tau &= \tau_e + \left.\frac{\partial\tau}{\partial\tau}\right|_e(\tau - \tau_e) = \tau_e + \tilde{\tau} \\ b\dot{\theta} &= b\dot{\theta}_e + b\left.\frac{\partial\dot{\theta}}{\partial\dot{\theta}}\right|_e(\dot{\theta} - \dot{\theta}_e) = b\dot{\tilde{\theta}}. \end{aligned}$$

Substituting into Equation (4.3) gives

$$\frac{m\ell^2}{3}\left[\ddot{\theta}_e + \tilde{\ddot{\theta}}\right] + \frac{mg\ell}{2}\left[\cos\theta_e - \sin\theta_e\tilde{\theta}\right] = [\tau_e + \tilde{\tau}] - b\left[\dot{\theta}_e + \dot{\tilde{\theta}}\right].$$

Simplifying this expression using Equation (??) gives

$$\frac{m\ell^2}{3}\tilde{\ddot{\theta}} - \frac{mg\ell}{2}(\sin\theta_e)\tilde{\theta} = \tilde{\tau} - b\dot{\tilde{\theta}}, \quad (4.5)$$

which are the linearized equations of motion using Jacobian linearization.

Feedback linearization proceeds by using the feedback linearizing control

$$\tau = \frac{mg\ell}{2}\cos\theta + \tilde{\tau} \quad (4.6)$$

in Equation (4.3) to obtain the feedback linearized equations of motion

$$\frac{m\ell^2}{3}\ddot{\theta} = \tilde{\tau} - b\dot{\tilde{\theta}}, \quad (4.7)$$

where we emphasize that the equation of motions are valid for any θ and $\dot{\theta}$, and not just in a small region about an equilibrium. It is interesting to compare Equation (4.6) to the control signal using Jacobian linearization which is

$$\tau = \tau_e + \tilde{\tau} = \frac{mg\ell}{2}\cos\theta_e + \tilde{\tau}. \quad (4.8)$$

The control signal in (4.8) uses the equilibrium angle θ_e whereas the control signal in (4.6) uses the actual angle θ . For the single link robot arm, in the remainder of the book we will use the design model obtained using feedback linearization.

4.3 Design Study B. Inverted Pendulum



Homework Problem B.4

For the inverted pendulum,

- (a) Find the equilibria of the system.
- (b) Linearize the system about the equilibria using Jacobian linearization.

Note that for the inverted pendulum, since the nonlinearities are not all contained in the same channel as the control force F , the system cannot be feedback linearized.

Solution

The differential equations describing the inverted derived in HW B.3 are

$$\begin{aligned} (m_1 + m_2)\ddot{z} + m_1 \frac{\ell}{2}\ddot{\theta} \cos \theta &= m_1 \frac{\ell}{2}\dot{\theta}^2 \sin \theta - b\dot{z} + F \\ m_1 \frac{\ell}{2}\ddot{z} \cos \theta + m_1 \frac{\ell^2}{4}\ddot{\theta} &= m_1 g \frac{\ell}{2} \sin \theta. \end{aligned} \quad (4.9)$$

The equilibria values (z_e, θ_e, F_e) are found by setting $\dot{z} = \ddot{z} = \dot{\theta} = \ddot{\theta} = 0$ in Equation (4.9) to obtain

$$F_e = 0 \quad (4.10)$$

$$m_1 g \frac{\ell}{2} \sin \theta_e = 0. \quad (4.11)$$

Therefore, any triple (z_e, θ_e, F_e) satisfying Equations (4.10) and (4.11) is an equilibria, or in other words z_e can be any value, $F_e = 0$ and $\theta_e = k\pi$, where k is an integer.

CHAPTER 4. EQUILIBRIA AND LINEARIZATION

To linearize around (z_e, θ_e, F_e) where k is an even integer, note that

$$\begin{aligned}
\ddot{\theta} \cos \theta &\approx \ddot{\theta}_e \cos \theta_e + \frac{\partial}{\partial \theta}(\ddot{\theta} \cos \theta) \Big|_{(\ddot{\theta}_e, \theta_e)} (\theta - \theta_e) + \frac{\partial}{\partial \ddot{\theta}}(\ddot{\theta} \cos \theta) \Big|_{(\ddot{\theta}_e, \theta_e)} (\ddot{\theta} - \ddot{\theta}_e) \\
&= \ddot{\theta}_e \cos \theta_e - (\ddot{\theta}_e \sin \theta_e) \tilde{\theta} + (\cos \theta_e) \ddot{\tilde{\theta}} \\
&= \ddot{\tilde{\theta}} \\
\dot{\theta}^2 \sin \theta &\approx \dot{\theta}_e^2 \sin \theta_e + \frac{\partial}{\partial \theta}(\dot{\theta}^2 \sin \theta) \Big|_{(\theta_e, \dot{\theta}_e)} (\theta - \theta_e) + \frac{\partial}{\partial \dot{\theta}}(\dot{\theta}^2 \sin \theta) \Big|_{(\theta_e, \dot{\theta}_e)} (\dot{\theta} - \dot{\theta}_e) \\
&= \dot{\theta}_e^2 \sin \theta_e + \dot{\theta}_e^2 \cos \theta_e \tilde{\theta} + 2\dot{\theta}_e \sin \theta_e \dot{\tilde{\theta}} \\
&= 0, \\
\ddot{z} \cos \theta &\approx \ddot{z}_e \cos \theta_e + \frac{\partial}{\partial \theta}(\ddot{z} \cos \theta) \Big|_{(\ddot{z}_e, \theta_e)} (\theta - \theta_e) + \frac{\partial}{\partial \ddot{z}}(\ddot{z} \cos \theta) \Big|_{(\ddot{z}_e, \theta_e)} (\ddot{z} - \ddot{z}_e) \\
&= \ddot{z}_e \cos \theta_e - (\ddot{z}_e \sin \theta_e) \tilde{\theta} + (\cos \theta_e) \ddot{\tilde{z}} \\
&= \ddot{\tilde{z}} \\
\sin \theta &\approx \sin \theta_e + \frac{\partial}{\partial \theta}(\sin \theta) \Big|_{\theta_e} (\theta - \theta_e) \\
&= \sin \theta_e + (\cos \theta_e) \tilde{\theta} \\
&= \tilde{\theta},
\end{aligned}$$

where we have defined $\tilde{\theta} \triangleq \theta - \theta_e$ and $\tilde{z} = z - z_e$. Also defining $\tilde{F} = F - F_e$, and noting that

$$\begin{aligned}
\theta &= \theta_e + \tilde{\theta} = \tilde{\theta} \\
\dot{\theta} &= \dot{\theta}_e + \dot{\tilde{\theta}} = \dot{\tilde{\theta}} \\
z &= z_e + \tilde{z} \\
\dot{z} &= \dot{z}_e + \dot{\tilde{z}} = \dot{\tilde{z}} \\
\ddot{z} &= \ddot{z}_e + \ddot{\tilde{z}} = \ddot{\tilde{z}} \\
F &= F_e + \tilde{F} = \tilde{F},
\end{aligned}$$

we can write Equation (4.9) in its linearized form as

$$\begin{aligned}
(m_1 + m_2)[\ddot{z}_e + \ddot{\tilde{z}}] + m_1 \frac{\ell}{2} [\ddot{\tilde{\theta}}] &= m_1 \frac{\ell}{2} [0] - b[\dot{z}_e + \dot{\tilde{z}}] + [F_e + \tilde{F}] \\
m_1 \frac{\ell}{2} [\ddot{\tilde{z}}] + m_1 \frac{\ell^2}{4} [\ddot{\theta}_e + \ddot{\tilde{\theta}}] &= m_1 g \frac{\ell}{2} [\tilde{\theta}],
\end{aligned}$$

which simplifies to

$$\begin{pmatrix} (m_1 + m_2) & m_1 \frac{\ell}{2} \\ m_1 \frac{\ell}{2} & m_1 \frac{\ell^2}{4} \end{pmatrix} \begin{pmatrix} \ddot{\tilde{z}} \\ \ddot{\tilde{\theta}} \end{pmatrix} = \begin{pmatrix} -b\dot{\tilde{z}} + \tilde{F} \\ m_1 g \frac{\ell}{2} \tilde{\theta} \end{pmatrix}, \quad (4.12)$$

which are the linearized equations of motion.

4.4 Design Study C. Satellite Attitude Control

Note that the satellite attitude control problem is already linear and so it doesn't make sense to linearize the equations of motion.

Notes and References

Jacobian linearization is a standard technique that is well documented in most introductory textbooks on control. For example, see [4, 5, 6, 7]. Feedback linearization is a much deeper topic than can be covered in an introductory textbook on control. A discussion that is similar to ours that assumes that the nonlinearity is in the input channel is in [5]. Feedback linearization is a fairly standard technique for fully actuated mechanical systems modeled by Euler-Lagrange equations, where the model of the system can be reduced to

$$M(q)\ddot{q} + c(q, \dot{q})\dot{q} + g(q) = \tau,$$

where $q \in \mathbb{R}^m$ is the generalized coordinate and $\tau \in \mathbb{R}^m$ is the generalized force, and where $M(q)$ is an invertible matrix. In that case we can write

$$\ddot{q} = M^{-1}(q)(\tau - c(q, \dot{q})\dot{q} - g(q)).$$

Selecting τ as

$$\tau = c(q, \dot{q})\dot{q} + g(q) + M(q)\nu, \quad (4.13)$$

results in the linear system

$$\ddot{q} = \nu,$$

where the new control variable ν can be designed to stabilize the system. The control law in Equation (4.13) is the feedback linearizing control. This a standard technique for robot manipulators and is often called the method of *computed torque* [8, 9, 10].

When the nonlinearity is not in the input channel, the system can still often be feedback linearized through a suitable change of variables. A good introduction to the topic is contained in [11]. More advanced coverage of the topic can be found in [12, 13, 14, 15].

Chapter 5

Transfer Function Models

5.1 Theory

The (one-sided) Laplace transform of a time domain signal $y(t)$ is defined as

$$\mathcal{L}\{y(t)\} \triangleq Y(s) = \int_0^\infty y(t)e^{-st} dt.$$

In essence, the Laplace transform indicates the content of the complex signal e^{-st} that is contained in $y(t)$. The Laplace transform of the time derivative of $y(t)$ is given by

$$\mathcal{L}\{\dot{y}\} = \int_0^\infty \dot{y}e^{-st} dt.$$

Using integration by parts $\int u dv = uv - \int v du$, where $u = e^{-st}$ and $dv = \dot{y}dt$, we get

$$\begin{aligned} \mathcal{L}\{\dot{y}\} &= \int_0^\infty \dot{y}e^{-st} dt \\ &= y(t)e^{-st} \Big|_0^\infty - \int_0^\infty y(t) (-se^{-st} dt) \\ &= \lim_{t \rightarrow \infty} y(t)e^{-st} - y(0) + s \int_0^\infty y(t)e^{-st} dt \\ &= sY(s) - y(0), \end{aligned}$$

assuming that $\lim_{t \rightarrow \infty} y(t)e^{-st}$, or that s is in the region of convergence of the Laplace transform. Similarly, the Laplace transform of the second derivative of $y(t)$ is given by

$$\begin{aligned} \mathcal{L}\{\ddot{y}\} &= s\mathcal{L}\{\dot{y}\} - \dot{y}(0) \\ &= s(sY(s) - y(0)) - \dot{y}(0) \\ &= s^2Y(s) - sy(0) - \dot{y}(0). \end{aligned}$$

CHAPTER 5. TRANSFER FUNCTION MODELS

The transfer function is typically defined as the relationship between the input and the output of the system. In that setting the transfer function is found when all initial conditions are set to zero. This is important to remember in the context of modeling. We will see that state space models explicitly account for initial conditions, but that transfer functions do not.

Given the n^{th} order differential equation

$$y^{(n)} + a_{n-1}y^{(n-1)} + \cdots + a_1\dot{y} + a_0y = b_m u^{(m)} + b_{m-1}u^{(m-1)} + \cdots + b_1\dot{u} + b_0u,$$

The transfer function from u to y is found by taking the Laplace transform of both sides of the equation and setting all initial conditions to zero to obtain

$$\begin{aligned} s^n Y(s) + a_{n-1}s^{n-1}Y(s) + \cdots + a_1sY(s) + a_0Y(s) \\ = b_m s^m U(s) + b_{m-1}s^{m-1}U(s) + \cdots + b_1sU(s) + b_0U(s). \end{aligned}$$

Factoring $Y(s)$ and $U(s)$ we get

$$\begin{aligned} (s^n + a_{n-1}s^{n-1} + \cdots + a_1s + a_0) Y(s) \\ = (b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1s + b_0) U(s). \end{aligned}$$

Solving for $Y(s)$ gives

$$Y(s) = \left(\frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1s + a_0} \right) U(s),$$

where

$$H(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1s + a_0}$$

is the transfer function from u to y .

Given the transfer function $H(s)$, the *poles* of the transfer function are the roots of the denominator polynomial

$$s^n + a_{n-1}s^{n-1} + \cdots + a_1s + a_0 = 0.$$

Similarly, the *zeros* of the transfer function are the roots of the numerator polynomial

$$b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1s + b_0 = 0.$$

As concrete example, consider the differential equation given by

$$y^{(3)} + 2\ddot{y} + 3\dot{y} + 4y = 5\dot{u} + 6u.$$

Taking the Laplace transform of both sides, setting the initial conditions to zero, and factoring $Y(s)$ and $U(s)$ gives

$$(s^3 + 2s^2 + 3s + 4)Y(s) = (5s + 6)U(s).$$

Solving for $Y(s)$ gives

$$Y(s) = \frac{5s + 6}{s^3 + 2s^2 + 3s + 4} U(s).$$

The transfer function from u to y is therefore

$$H(s) = \frac{5s + 6}{s^3 + 2s^2 + 3s + 4}.$$

The poles of the transfer function are the roots of $s^3 + 2s^2 + 3s + 4 = 0$ or $p_1 = -1.6506$, $p_2 = -0.1747 + j1.5469$, and $p_3 = -0.1747 - j1.5469$. The zeros of the transfer function are the roots of $5s + 6 = 0$ or $z_1 = -6/5$.

5.2 Design Study A. Single Link Robot Arm



Homework Problem A.5

For the single link robot arm, find the transfer function of the system from the torque τ to the angle θ .

Solution

As shown in Section 4.2, the feedback linearized model for the single link robot arm is given by Equation (4.7) as

$$\frac{m\ell^2}{3}\ddot{\theta} = \tilde{\tau} - b\dot{\theta}. \quad (5.1)$$

Taking the Laplace transform of Equation (5.1) and setting all initial conditions to zero we get

$$\frac{m\ell^2}{3}s^2\Theta(s) + bs\Theta(s) = \tilde{\tau}(s).$$

Solving for $\Theta(s)$ gives

$$\Theta(s) = \left(\frac{1}{\frac{m\ell^2}{3}s^2 + bs} \right) \tilde{\tau}(s).$$

The canonical form for transfer functions is for the leading coefficient in the denominator polynomial to be unity. This is called monic form. Putting the transfer function in monic form results in

$$\Theta(s) = \left(\frac{\frac{3}{m\ell^2}}{s^2 + \frac{3b}{m\ell^2}s} \right) \tilde{\tau}(s), \quad (5.2)$$

where the expression in the parenthesis is the transfer function from $\tilde{\tau}$ to θ , where $\tilde{\tau}$ indicate that we are working with feedback linearized control in Equation (4.6). The block diagram associated with Equation (5.2) is shown in Figure 5.1

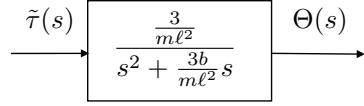


Figure 5.1: A block diagram of the single link robot arm.

5.3 SIMO Systems and Cascade Approximations

There are many systems that have multiple inputs and/or multiple outputs. In this case, a transfer function exists from each input to each output. For example, suppose that the system is given by the coupled differential equations

$$\begin{aligned}\ddot{y}_1 + a_{11}\dot{y}_1 + \ddot{y}_2 + a_{12}\dot{y}_2 &= b_{11}u_1 + b_{12}u_2 + b_{13}\dot{u}_2 \\ \ddot{y}_2 + a_{22}y_2 &= b_{21}u_2.\end{aligned}$$

Taking the Laplace transform and arranging in matrix notation gives

$$\left(\begin{array}{c|c} s^2 + a_{11}s & s^2 + a_{12}s \\ \hline 0 & s^2 + a_{22} \end{array} \right) \begin{pmatrix} Y_1(s) \\ Y_2(s) \end{pmatrix} = \left(\begin{array}{c|c} b_{11} & (b_{13}s + b_{12}) \\ \hline 0 & b_{21} \end{array} \right) \begin{pmatrix} U_1(s) \\ U_2(s) \end{pmatrix}.$$

To find the relevant transfer functions, we need to isolate the outputs on the left hand side by inverting the matrix on the left. Using the inverse formula for a 2×2 matrix derived in Appendix f

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}}{ad - bc}$$

we get

$$\begin{aligned}\begin{pmatrix} Y_1(s) \\ Y_2(s) \end{pmatrix} &= \left(\begin{array}{c|c} s^2 + a_{11}s & s^2 + a_{12}s \\ \hline 0 & s^2 + a_{22} \end{array} \right)^{-1} \left(\begin{array}{c|c} b_{11} & b_{13}s + b_{12} \\ \hline 0 & b_{21} \end{array} \right) \begin{pmatrix} U_1(s) \\ U_2(s) \end{pmatrix} \\ &= \frac{\left(\begin{array}{c|c} s^2 + a_{22} & -s^2 - a_{12}s \\ \hline 0 & s^2 + a_{11}s \end{array} \right)}{(s^2 + a_{11}s)(s^2 + a_{22})} \left(\begin{array}{c|c} b_{11} & b_{13}s + b_{12} \\ \hline 0 & b_{21} \end{array} \right) \begin{pmatrix} U_1(s) \\ U_2(s) \end{pmatrix} \\ &= \frac{\left(\begin{array}{c|c} b_{11}(s^2 + a_{22}) & (s^2 + a_{22})(b_{13}s + b_{12}) - b_{21}(s^2 + a_{12}s) \\ \hline 0 & b_{21}(s^2 + a_{11}s) \end{array} \right)}{(s^2 + a_{11}s)(s^2 + a_{22})} \begin{pmatrix} U_1(s) \\ U_2(s) \end{pmatrix} \\ &= \left(\begin{array}{c|c} \frac{b_{11}}{s^2 + a_{11}s} & \frac{b_{13}s^3 + (b_{12} - b_{21})s^2 + (b_{13}a_{22} - b_{21}a_{12})s + a_{22}b_{12}}{(s^2 + a_{11}s)(s^2 + a_{22})} \\ \hline 0 & \frac{b_{21}}{s^2 + a_{22}} \end{array} \right) \begin{pmatrix} U_1(s) \\ U_2(s) \end{pmatrix}.\end{aligned}$$

Therefore, the transfer function from u_1 to y_1 is $\frac{b_{11}}{s^2 + a_{11}s}$, the transfer function from u_2 to y_1 is $\frac{b_{13}s^3 + (b_{12} - b_{21})s^2 + (b_{13}a_{22} - b_{21}a_{12})s + a_{22}b_{12}}{(s^2 + a_{11}s)(s^2 + a_{22})}$, the transfer function from u_1 to y_2 is 0, and the transfer function from u_2 to y_2 is $\frac{b_{21}}{s^2 + a_{22}}$.

CHAPTER 5. TRANSFER FUNCTION MODELS

For many of the case studies that we will look at in this book, there is a single input, but two or more measurements or outputs. For example, suppose that the system of interest is described by the following set of coupled differential equations

$$\begin{aligned}\ddot{y}_1 &= u - 0.1\dot{y}_1 + 2y_2 \\ \ddot{y}_2 &= 3u + 4\dot{y}_1 - 200y_2 - 20\dot{y}_2.\end{aligned}$$

Taking the Laplace transform and rearranging the equations gives

$$\begin{aligned}(s^2 + 0.1s)Y_1(s) - 2Y_2(s) &= U(s) \\ (s^2 + 20s + 200)Y_2(s) - 4sY_1(s) &= 3U(s).\end{aligned}\tag{5.3}$$

In matrix format we have

$$\left(\begin{array}{c|c} s^2 + 0.1s & -2 \\ \hline -4s & s^2 + 20s + 200 \end{array} \right) \begin{pmatrix} Y_1(s) \\ Y_2(s) \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix} U(s).$$

Using the 2×2 matrix inversion formula $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$ results in

$$\begin{aligned}\begin{pmatrix} Y_1(s) \\ Y_2(s) \end{pmatrix} &= \frac{\left(\begin{array}{c|c} s^2 + 20s + 200 & 2 \\ \hline 4s & s^2 + 0.1s \end{array} \right)}{(s^2 + 0.1s)(s^2 + 20s + 200) - 8s} \begin{pmatrix} 1 \\ 3 \end{pmatrix} U(s) \\ &= \begin{pmatrix} \frac{s^2 + 20s + 200}{s^4 + 20.1s^3 + 202s^2 + 12s} \\ \frac{3s^2 + 4.3s}{s^4 + 20.1s^3 + 202s^2 + 12s} \end{pmatrix} U(s).\end{aligned}\tag{5.4}$$

This book does not cover design techniques for MIMO systems represented in transfer function form. However, for a certain class of physical systems we can simplify the model as a cascade of fast and slow subsystems. Note that the poles of the transfer functions in Equation (5.4) are at 0, -0.06 , and $-10.02 \pm j10.02$. Therefore, there are two slow poles at 0 and -0.06 , and two fast poles at $-10.02 \pm j10.02$.

Returning to Equation 5.3 we can rearrange those equations as

$$\begin{aligned}(s^2 + 0.1s)Y_1(s) &= U(s) + 2Y_2(s) \\ (s^2 + 20s + 200)Y_2(s) &= 3U(s) + 4sY_1(s),\end{aligned}$$

and then solve for Y_1 and Y_2 as

$$Y_1(s) = \frac{1}{s^2 + 0.1s} U(s) + \frac{2}{s^2 + 0.1s} Y_2(s)\tag{5.5}$$

$$Y_2(s) = \frac{3}{s^2 + 20s + 200} U(s) + \frac{4s}{s^2 + 20s + 200} Y_1(s),\tag{5.6}$$

where Equation (5.5) with poles at 0 and -0.1 approximate the slow subsystem, and Equation (5.6) with poles at $-10 \pm j10$ approximate the fast subsystem.

The basic idea is to approximate the system as a cascade of the fast subsystem driven by the input U with the fast state $Y_2(s)$ as the output, followed by the slow subsystem with the fast state Y_2 as the input and the slow state Y_1 as the output as shown in Figure 5.2, where

$$D_1(s) \triangleq \frac{1}{s^2 + 0.1s} U(s)$$

$$D_2(s) \triangleq \frac{4s}{s^2 + 20s + 200} Y_1(s)$$

are considered as unknown disturbances that arise from modeling errors.

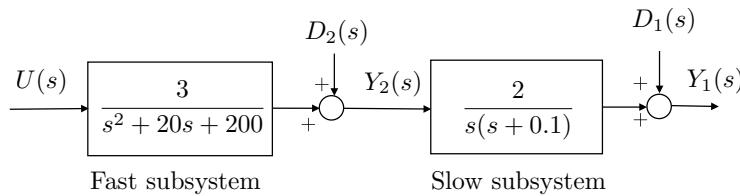


Figure 5.2: System approximated by cascade of fast and slow subsystems.

5.4 Design Study B. Inverted Pendulum



Homework Problem B.5

For the inverted pendulum:

- (a) Start with the linearized equations and use the Laplace transform to convert the equations of motion to the s-domain.
- (b) Find the transfer functions from the input $\tilde{F}(s)$ to the outputs $\tilde{Z}(s)$ and $\tilde{\Theta}(s)$. Assume that the damping constant is negligible. How does this simplify the transfer functions? Is this a reasonable assumption?
- (c) From the simplified $\tilde{Z}(s)/\tilde{F}(s)$ and $\tilde{\Theta}(s)/\tilde{F}(s)$ transfer functions, compute the $\tilde{Z}(s)/\tilde{\Theta}(s)$ transfer function. Draw a block diagram of the system as a series of two transfer functions from $\tilde{F}(s)$ to $\tilde{\Theta}(s)$ and from $\tilde{\Theta}(s)$ to $\tilde{Z}(s)$.
- (d) Describe how this transfer-function cascade makes sense physically. How does force influence the pendulum angle? How does the pendulum angle influence the cart position?

CHAPTER 5. TRANSFER FUNCTION MODELS

Solution

From HW B.5, the linearized equations of motion are given by

$$\begin{pmatrix} (m_1 + m_2) & m_1 \frac{\ell}{2} \\ m_1 \frac{\ell}{2} & m_1 \frac{\ell^2}{4} \end{pmatrix} \begin{pmatrix} \ddot{z} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} -b\dot{z} + \tilde{F} \\ m_1 g \frac{\ell}{2} \tilde{\theta} \end{pmatrix}.$$

The second equation in this matrix formulation can be simplified by dividing both sides of the second equation by $m_1 \frac{\ell}{2}$ to give

$$\begin{pmatrix} (m_1 + m_2) & m_1 \frac{\ell}{2} \\ 1 & \frac{\ell}{2} \end{pmatrix} \begin{pmatrix} \ddot{z} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} -b\dot{z} + \tilde{F} \\ g\tilde{\theta} \end{pmatrix}.$$

We can write these equations with states and state derivatives on the left and inputs on the right as

$$\begin{aligned} (m_1 + m_2)\ddot{z} + m_1 \frac{\ell}{2}\ddot{\theta} + b\dot{z} &= \tilde{F} \\ \ddot{z} + \frac{\ell}{2}\ddot{\theta} - g\tilde{\theta} &= 0. \end{aligned}$$

Taking the Laplace transform gives

$$\begin{aligned} [(m_1 + m_2)s^2 + bs] \tilde{Z}(s) + m_1 \frac{\ell}{2}s^2 \tilde{\Theta}(s) &= \tilde{F}(s) \\ s^2 \tilde{Z}(s) + \left(\frac{\ell}{2}s^2 - g\right) \tilde{\Theta}(s) &= 0. \end{aligned}$$

These equations can be expressed in matrix form as

$$\begin{pmatrix} (m_1 + m_2)s^2 + bs & m_1 \frac{\ell}{2}s^2 \\ s^2 & \frac{\ell}{2}s^2 - g \end{pmatrix} \begin{pmatrix} \tilde{Z}(s) \\ \tilde{\Theta}(s) \end{pmatrix} = \begin{pmatrix} \tilde{F}(s) \\ 0 \end{pmatrix}.$$

Inverting the matrix on the left hand side and solving for the transfer functions gives

$$\begin{aligned} \frac{\tilde{Z}(s)}{\tilde{F}(s)} &= \frac{\frac{\ell}{2}s^2 - g}{m_2 \frac{\ell}{2}s^4 + b\frac{\ell}{2}s^3 - (m_1 + m_2)gs^2 - bgs} \\ \frac{\tilde{\Theta}(s)}{\tilde{F}(s)} &= \frac{-s^2}{m_2 \frac{\ell}{2}s^4 + b\frac{\ell}{2}s^3 - (m_1 + m_2)gs^2 - bgs}. \end{aligned}$$

If we make the assumption that $b \approx 0$, then these transfer functions will simplify further and be even easier to work with from a control design perspective. This is a reasonable and conservative assumption because the damping force $b\dot{z}$ is small relative to the other forces acting on the system. Furthermore, by underestimating the damping in the system, our control design will be conservative because it assumes there is no damping provided by the physics of the

CHAPTER 5. TRANSFER FUNCTION MODELS

system and thus all the damping in the system must come from the feedback control. Assuming $b = 0$, we get

$$\begin{aligned}\frac{\tilde{Z}(s)}{\tilde{F}(s)} &= \frac{\frac{\ell}{2}s^2 - g}{s^2 [m_2 \frac{\ell}{2}s^2 - (m_1 + m_2)g]} \\ \frac{\tilde{\Theta}(s)}{\tilde{F}(s)} &= \frac{-1}{m_2 \frac{\ell}{2}s^2 - (m_1 + m_2)g}.\end{aligned}$$

From a controls perspective, we will be interested in how the pendulum angle θ influences the cart position z and so we want to find the $\tilde{Z}(s)/\tilde{\Theta}(s)$ transfer function. This can be easily calculated by recognizing that

$$\frac{\tilde{Z}(s)}{\tilde{\Theta}(s)} = \frac{\tilde{Z}(s)/\tilde{F}(s)}{\tilde{\Theta}(s)/\tilde{F}(s)}.$$

Accordingly,

$$\frac{\tilde{Z}(s)}{\tilde{\Theta}(s)} = -\frac{\frac{\ell}{2}s^2 - g}{s^2}.$$

The block diagram for the approximate system is shown in Figure 5.3.

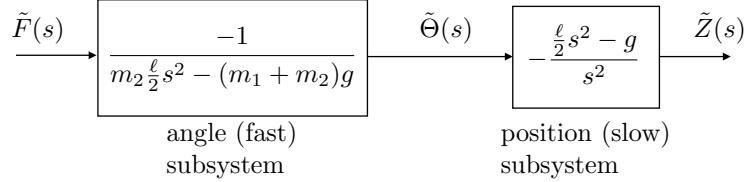


Figure 5.3: The inverted pendulum dynamics are approximated by cascade of fast and slow subsystems. The fast subsystem is the transfer function from the force to the angle, and the slow subsystem is the transfer function from the angle to the position.

This transfer function cascade makes since physically. With the pendulum balanced vertically, a positive force on the cart would cause the pendulum to fall in the negative direction as indicated by the minus sign in the numerator. The pendulum falls in an unstable motion due to the right-half-plane pole in the transfer function. The angle of the rod influences the position of the cart as shown in the second transfer function. If the pendulum were balanced vertically with no input force applied and a small disturbance caused the pendulum to fall in the positive direction, the cart would shoot off in the negative direction. Again, this can be seen from the negative sign in the numerator of the transfer function and the right-half-plane pole in the denominator. Keep in mind that these equations are linearized about the vertical position of the pendulum. When the pendulum is hanging down, the equations are different. How are they different?

The PID and loopshaping techniques that we will discuss in Parts III and V will use the approximate transfer function models of the plant derived above. The state space methods discussed in Part IV will use a state space model, which does not depend on neglecting the coupling between subsystems, and in fact does not depend on a separation into fast and slow subsystems.

5.5 Design Study C. Satellite Attitude Control



Homework Problem C.5

For the satellite attitude control problem:

- (a) Start with the linearized equations for the satellite attitude problem and use the Laplace transform to convert the equations of motion to the s-domain.
- (b) Find the full transfer matrix from the input $\tau(s)$ to the outputs $\Phi(s)$ and $\Theta(s)$. Identify the fast and slow subsystem.
- (c) Find an approximation to the system that is a cascade of a SISO fast system and a SISO slow system, and identify the disturbances that are being ignored.
- (d) Argue that the fast-slow cascade approximation makes sense physically.

Solution

From HW C.3, the linearized equations of motion are given by

$$\begin{pmatrix} \ddot{\theta} \\ \ddot{\phi} \end{pmatrix} = \begin{pmatrix} -\frac{b}{J_s}(\dot{\theta} - \dot{\phi}) - \frac{k}{J_s}(\theta - \phi) + \frac{1}{J_s}\tau \\ -\frac{b}{J_p}(\dot{\phi} - \dot{\theta}) - \frac{k}{J_p}(\phi - \theta) \end{pmatrix}$$

or in other words, the coupled differential equations

$$\begin{aligned} \ddot{\theta} + \frac{b}{J_s}\dot{\theta} + \frac{k}{J_s}\theta &= \frac{b}{J_s}\dot{\phi} + \frac{k}{J_s}\phi + \frac{1}{J_s}\tau \\ \ddot{\phi} + \frac{b}{J_p}\dot{\phi} + \frac{k}{J_p}\phi &= \frac{b}{J_p}\dot{\theta} + \frac{k}{J_p}\theta \end{aligned}$$

Taking the Laplace transform with initial conditions set to zero and rearranging gives

$$(s^2 + \frac{b}{J_s}s + \frac{k}{J_s})\Theta(s) = (\frac{b}{J_s}s + \frac{k}{J_s})\phi(s) + \frac{1}{J_s}\tau(s) \quad (5.7)$$

$$(s^2 + \frac{b}{J_p}s + \frac{k}{J_p})\phi(s) = (\frac{b}{J_p}s + \frac{k}{J_p})\Theta(s). \quad (5.8)$$

CHAPTER 5. TRANSFER FUNCTION MODELS

To find the transfer matrix from τ to $(\Theta, \Phi)^\top$, write Equation (5.7) and (5.8) in matrix form as

$$\left(\begin{array}{c|c} s^2 + \frac{b}{J_s}s + \frac{k}{J_s} & -\frac{b}{J_s}s - \frac{k}{J_s} \\ \hline -\frac{b}{J_p}s - \frac{k}{J_p} & s^2 + \frac{b}{J_p}s + \frac{k}{J_p} \end{array} \right) \begin{pmatrix} \Theta(s) \\ \Phi(s) \end{pmatrix} = \begin{pmatrix} \frac{1}{J_s} \\ 0 \end{pmatrix} \tau(s),$$

and invert the matrix on the left hand side to obtain

$$\begin{pmatrix} \Theta(s) \\ \Phi(s) \end{pmatrix} = \begin{pmatrix} \frac{\frac{1}{J_s}s^2 + \frac{b}{J_sJ_p}s + \frac{k}{J_pJ_s}}{s^4 + \frac{b(J_s+J_p)}{J_pJ_s}s^3 + \frac{k(J_s+J_p)}{J_pJ_s}s^2} \\ \frac{\frac{b}{J_p}s + \frac{k}{J_s}}{s^4 + \frac{b(J_s+J_p)}{J_pJ_s}s^3 + \frac{k(J_s+J_p)}{J_pJ_s}s^2} \end{pmatrix} \tau(s).$$

Plugging in the nominal values from Section C we get that there are two fast poles at $-0.0300 \pm j0.4232$, and two slow poles at 0. Therefore the system is a good candidate for a cascade approximation of fast and slow dynamics.

To find the approximate transfer functions, return to Equations (5.7) and (5.8) and divide by the polynomials on the left hand side to obtain

$$\Theta(s) = \frac{\frac{b}{J_s}s + \frac{k}{J_s}}{s^2 + \frac{b}{J_s}s + \frac{k}{J_s}} \phi(s) + \frac{\frac{1}{J_s}}{s^2 + \frac{b}{J_s}s + \frac{k}{J_s}} \tau(s) \quad (5.9)$$

$$\phi(s) = \frac{\frac{b}{J_p}s + \frac{k}{J_p}}{s^2 + \frac{b}{J_p}s + \frac{k}{J_p}} \Theta(s). \quad (5.10)$$

Note that the roots of the polynomial $s^2 + \frac{b}{J_s}s + \frac{k}{J_s} = 0$ are at $-0.0050 \pm j0.1731$ with magnitude 0.1732, and the roots of the polynomial $s^2 + \frac{b}{J_p}s + \frac{k}{J_p} = 0$ are at $-0.0250 \pm j0.3865$ with magnitude 0.3873. Therefore, Equation (5.9) approximate the slow dynamics and (5.10) approximate the fast dynamics. Therefore, we let τ drive the fast dynamics with output Θ and we let Θ drive the slow dynamics with output Φ to get

$$\begin{aligned} \Theta(s) &= \frac{\frac{1}{J_s}}{s^2 + \frac{b}{J_s}s + \frac{k}{J_s}} \tau(s) + D(s) \\ \Phi(s) &= \frac{\frac{b}{J_p}s + \frac{k}{J_p}}{s^2 + \frac{b}{J_p}s + \frac{k}{J_p}} \Theta(s). \end{aligned}$$

where we assume that $d(t) = \mathcal{L}^{-1}\{D(s)\}$ is an unknown disturbance signal. The block diagram for the approximate system is shown in Figure 5.4

The cascade approximation makes since physically because the torque on the inner pod has an almost immediate effect on the angle θ of the inner pod. The angle θ then effects the angle of the solar panels ϕ through the spring damper system.

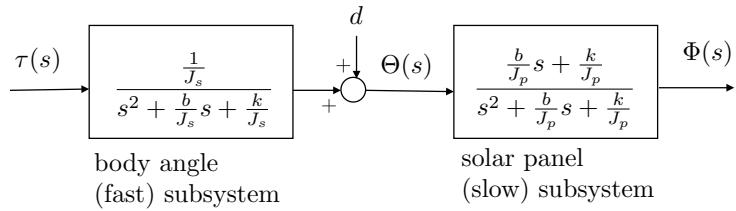


Figure 5.4: The satellite attitude dynamics are approximated by cascade of fast and slow subsystems. The fast subsystem is the transfer function from the torque to the angle θ , and the slow subsystem is the transfer function from the angle θ to the angle ϕ .

Notes and References

The Laplace transform can be found in introductory textbooks on signals and systems like [16] and in introductory textbooks on differential equations. The transfer function of a linear time invariant system is also covered in introductory textbooks on signals and systems, and in most introductory textbooks on control like [4].

Chapter 6

State Space Models

6.1 Theory

The transfer function models discussed in the previous chapter are a frequency domain representation of the system because they describe how the system responds to sinusoidal inputs at specific frequencies. In contrast, differential equation models are time-domain representations of the systems that directly model the time-evolution of the system. The *state space* model is also a time-domain representation that is essentially a reformatting of the differential equations. In essence, the state space model represents the system by an input, an output, and a memory element, that is called the state. In this book, the input will always be denoted as $u(t)$, the output as $y(t)$, and the state as $x(t)$. The state represents all of the memory elements in the system. For example, the state may represent a storage register, or the altitude of an aircraft, or the velocity of a car, or the voltage across a capacitor, or the current through an inductor.

The state space model is composed of two equations. The first equation is called the *state evolution equation* and represents how the memory elements, or state, change as a function of the current state and the inputs to the system. The second equation is called the *output equation* and describes how the current output of the system depends on the current state and the current input. The general state space equations for a continuous time system are written as

$$\dot{x} = f(x, u) \quad (6.1)$$

$$y = h(x, u), \quad (6.2)$$

where Equation (6.1) is the state evolution equation and Equation (6.2) is the output equation. Note that for continuous time systems the state evolution equation is a system of coupled first order nonlinear differential equation. The general state space equations for a discrete time system are written as

$$x[k + 1] = f(x[k], u[k]) \quad (6.3)$$

$$y[k] = h(x[k], u[k]), \quad (6.4)$$

CHAPTER 6. STATE SPACE MODELS

where again Equation (6.3) is the state evolution equation, this time given by a system of coupled first order difference equation, and Equation (6.4) is the output equation.

Any nonlinear n^{th} order differential equations can be transformed into state space form. For example, consider the nonlinear differential equation

$$\ddot{y} + y^2 \dot{y} + \sin(y\dot{y}) + e^y = u,$$

where u is the input and y is the output. Define the state as

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \triangleq \begin{pmatrix} y \\ \dot{y} \\ \ddot{y} \end{pmatrix}.$$

Then the state evolution equation can be derived as

$$\begin{aligned} \dot{x} &= \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} \dot{y} \\ \ddot{y} \\ \dddot{y} \end{pmatrix} = \begin{pmatrix} \dot{y} \\ \ddot{y} \\ -y^2 \dot{y} - \sin(y\dot{y}) - e^y + u \end{pmatrix} \\ &= \begin{pmatrix} x_2 \\ x_3 \\ -x_1^2 x_3 - \sin(x_1 x_2) - e^{x_1} + u \end{pmatrix} \triangleq f(x, u), \end{aligned}$$

and the output equation is

$$y = x_1 \triangleq h(x, u).$$

As another example, consider the coupled differential equations

$$\begin{aligned} \ddot{y}_1 + \dot{y}_1 y_2 + y_2 &= u \\ \ddot{y}_2 + \dot{y}_2 \cos(y_1) + \dot{y}_1 &= 0. \end{aligned}$$

Defining the output as $y = (y_1, y_2)^\top$ and the state as

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \triangleq \begin{pmatrix} y_1 \\ y_2 \\ \dot{y}_1 \\ \dot{y}_2 \end{pmatrix},$$

the state evolution equation can be derived as

$$\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \ddot{y}_1 \\ \ddot{y}_2 \end{pmatrix} = \begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ -\dot{y}_1 y_2 - y_2 + u \\ -\dot{y}_2 \cos(y_1) - \dot{y}_1 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ -x_3 x_2 - x_2 + u \\ -x_4 \cos(x_1) - x_3 \end{pmatrix} \triangleq f(x, u),$$

and the output equation is given by

$$y = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \triangleq h(x, u).$$

6.1.1 Jacobian Linearization Revisited

Jacobian linearization of the system is more straightforward when the equations are in state space form. Given the nonlinear state space equations

$$\dot{x} = f(x, u) \quad (6.5)$$

$$y = h(x, u) \quad (6.6)$$

an equilibrium point is defined as any (x_e, u_e) such that $f(x_e, u_e) = 0$. In other words, equilibrium points are combinations of states and inputs where the state stops evolving in time. To linearize about an equilibrium point, Equations (6.5) and (6.6) can be expanded in a Taylor series about (x_e, u_e) as

$$f(x, u) \approx f(x_e, u_e) + \frac{\partial f}{\partial x}\Big|_e (x - x_e) + \frac{\partial f}{\partial u}\Big|_e (u - u_e) + H.O.T. \quad (6.7)$$

$$h(x, u) \approx h(x_e, u_e) + \frac{\partial h}{\partial x}\Big|_e (x - x_e) + \frac{\partial h}{\partial u}\Big|_e (u - u_e) + H.O.T., \quad (6.8)$$

where the Jacobian matrices are defined as

$$\frac{\partial f}{\partial x} \triangleq \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

$$\frac{\partial f}{\partial u} \triangleq \begin{pmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{pmatrix}$$

$$\frac{\partial h}{\partial x} \triangleq \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial h_p}{\partial x_1} & \cdots & \frac{\partial h_p}{\partial x_n} \end{pmatrix}$$

$$\frac{\partial h}{\partial u} \triangleq \begin{pmatrix} \frac{\partial h_1}{\partial u_1} & \cdots & \frac{\partial h_1}{\partial u_m} \\ \vdots & & \vdots \\ \frac{\partial h_p}{\partial u_1} & \cdots & \frac{\partial h_p}{\partial u_m} \end{pmatrix}.$$

Therefore $\frac{\partial f}{\partial x}$ is an $n \times n$ matrix, $\frac{\partial f}{\partial u}$ is an $n \times m$ matrix, $\frac{\partial h}{\partial x}$ is a $p \times n$ matrix, and $\frac{\partial h}{\partial u}$ is a $p \times m$ matrix.

Defining $\tilde{x} \triangleq x - x_e$ and $\tilde{u} = u - u_e$ and letting $A = \frac{\partial f}{\partial x}\Big|_e$ and $B = \frac{\partial f}{\partial u}\Big|_e$, and noting that $\dot{\tilde{x}} = \dot{x} - \dot{x}_e = \dot{x}$, results in the linearized state evolution equation

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u}.$$

At the equilibria, the output may not necessarily be zeros. If we define the equilibrium output to be $y_e = h(x_e, u_e)$ and the linearized output as $\tilde{y} = y - y_e$,

and define $C \triangleq \frac{\partial h}{\partial x}|_e$ and $D \triangleq \frac{\partial h}{\partial u}|_e$, then using Equation (6.8) in (6.6) gives

$$\begin{aligned}\tilde{y} &= y - y_e \\ &\approx h(x_e, u_e) + C\tilde{x} + D\tilde{u} - h(x_e, u_e) \\ &= C\tilde{x} + D\tilde{u}.\end{aligned}$$

The linearized state space equations are therefore

$$\begin{aligned}\dot{\tilde{x}} &= A\tilde{x} + B\tilde{u} \\ \tilde{y} &= C\tilde{x} + D\tilde{u}.\end{aligned}$$

6.1.2 Converting State Space Models to Transfer Function Models

Linear state space models can be converted to transfer function models using the following technique. Suppose that the linear state space model is given by

$$\dot{x} = Ax + Bu \quad (6.9)$$

$$y = Cx + Du. \quad (6.10)$$

Defining the Laplace transform of a vector to be the Laplace transform of each element, i.e.,

$$\mathcal{L} \left\{ \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \right\} = \begin{pmatrix} \mathcal{L}\{x_1\} \\ \vdots \\ \mathcal{L}\{x_n\} \end{pmatrix},$$

and noting that the Laplace transform is a linear operator, which implies that

$$\mathcal{L}\{Ax\} = A\mathcal{L}\{x\},$$

taking the Laplace transform of Equations (6.9) and (6.10), and setting the initial condition to zeros, gives

$$sX(s) = AX(s) + BU(s) \quad (6.11)$$

$$Y(s) = CX(s) + DU(s). \quad (6.12)$$

Solving (6.11) for $X(s)$ gives

$$\begin{aligned}sX(s) - AX(s) &= BU(s) \\ \implies (sI - A)X(s) &= BU(s) \\ \implies X(s) &= (sI - A)^{-1}BU(s),\end{aligned} \quad (6.13)$$

where I is the $n \times n$ identity matrix. Substituting (6.13) into (6.12) gives

$$\begin{aligned}Y(s) &= C(sI - A)^{-1}BU(s) + DU(s) \\ \implies Y(s) &= [C(sI - A)^{-1}B + D]U(s).\end{aligned}$$

CHAPTER 6. STATE SPACE MODELS

The transfer function from u to y is therefore

$$H(s) = C(sI - A)^{-1}B + D. \quad (6.14)$$

As an example, suppose that the state space equations are given by

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 1 \\ -4 & -3 \end{pmatrix}x + \begin{pmatrix} 0 \\ 2 \end{pmatrix}u \\ y &= \begin{pmatrix} 1 & 0 \end{pmatrix}x + (0)u,\end{aligned}$$

then the transfer function from u to y is given by

$$\begin{aligned}H(s) &= C(sI - A)^{-1}B + D \\ &= (1 \ 0) \left(s \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ -4 & -3 \end{pmatrix} \right)^{-1} \begin{pmatrix} 0 \\ 2 \end{pmatrix} + 0 \\ &= (1 \ 0) \begin{pmatrix} s & -1 \\ 4 & s+3 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \\ &= (1 \ 0) \frac{\begin{pmatrix} s+3 & 1 \\ -4 & s \end{pmatrix}}{s^2 + 3s + 4} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \\ &= \frac{(s+3 \ 1)}{s^2 + 3s + 4} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \\ &= \frac{2}{s^2 + 3s + 4}.\end{aligned}$$

The form of the transfer function in Equation (6.14) allows us to derive a simple relationship between the poles of the transfer function $H(s)$ and the eigenvalues of A . Recall (see Appendix f) that for any square invertible matrix M , we have that

$$M^{-1} = \frac{\text{adj}(M)}{\det(M)},$$

where $\text{adj}(M)$ is the adjugate of M defined as the transpose of the cofactors of M , and $\det(M)$ is the determinant of M . Therefore

$$\begin{aligned}H(s) &= C(sI - A)^{-1}B + D \\ &= \frac{C\text{adj}(sI - A)B}{\det(sI - A)} + D \\ &= \frac{C\text{adj}(sI - A)B + D\det(sI - A)}{\det(sI - A)}.\end{aligned}$$

This expression implies that the zeros of $H(s)$ are given by the roots of the polynomial

$$C\text{adj}(sI - A)B + D\det(sI - A) = 0,$$

and that the poles of $H(s)$ are given by the roots of the polynomial

$$\det(sI - A) = 0. \quad (6.15)$$

Recall from linear algebra (see Appendix F) that Equation (6.15) also defines the eigenvalues of A . Therefore, the poles of $H(s)$ are equivalent to the eigenvalues of A .

In Chapters 11 and 13 we will show two general techniques for converting SISO transfer function models into state space models.

6.2 Design Study A. Single Link Robot Arm



Homework Problem A.6

For the feedback linearized equations of motion for the single link robot arm given in Equation (4.7), define the states as $x = (\theta, \dot{\theta})^\top$, and the input as $\tilde{u} = \tilde{\tau}$, and the measured output as $y = \theta$. Find the linear state space equations in the form

$$\begin{aligned}\dot{x} &= Ax + B\tilde{u} \\ y &= Cx + D\tilde{u}.\end{aligned}$$

Solution

The linear state space equations can be derived in two different ways: (1) directly from the linearized equations of motion, and (2) by linearizing the nonlinear equations of motion.

Starting with the linear state space equation in Equation (4.7) and solving for $\ddot{\theta}$ gives

$$\ddot{\theta} = \frac{3}{m\ell^2} \tilde{\tau} - \frac{3b}{m\ell^2} \dot{\theta}.$$

Therefore,

$$\begin{aligned}\dot{x} &\triangleq \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ \frac{3}{m\ell^2} \tilde{\tau} - \frac{3b}{m\ell^2} \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{x}_2 \\ \frac{3}{m\ell^2} \tilde{\tau} - \frac{3b}{m\ell^2} x_2 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 \\ 0 & -\frac{3b}{m\ell^2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{3}{m\ell^2} \end{pmatrix} \tilde{\tau}.\end{aligned}$$

Assuming that the measured output of the system is $y = \theta$, the linearized output is given by

$$y = \theta = x_1 = (1 \ 0) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (0) \tilde{\tau}.$$

CHAPTER 6. STATE SPACE MODELS

Therefore, the linearized state space equations are given by

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 1 \\ \frac{3g \sin \theta_e}{2\ell} & -\frac{3b}{2\ell} \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{3}{m\ell^2} \end{pmatrix} \tilde{u} \\ y &= (1 \ 0) x.\end{aligned}\quad (6.16)$$

Alternatively, the state space equations can be found directly from the nonlinear equations of motion given in Equation (3.1), by forming the nonlinear state space model as

$$\dot{x} \triangleq \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ \frac{3}{m\ell^2} \tau - \frac{3b}{m\ell^2} \dot{\theta} - \frac{3g}{2\ell} \cos \theta \end{pmatrix} \triangleq f(x, u).$$

Evaluating the Jacobians at the equilibrium $(\theta_e, \dot{\theta}_e, \tau_e) = (0, 0, \frac{mg\ell}{2})$ gives

$$\begin{aligned}A &= \left. \frac{\partial f}{\partial x} \right|_e = \begin{pmatrix} \frac{\partial f_1}{\partial \theta} & \frac{\partial f_1}{\partial \dot{\theta}} \\ \frac{\partial f_2}{\partial \theta} & \frac{\partial f_2}{\partial \dot{\theta}} \end{pmatrix} \Big|_e \\ &= \begin{pmatrix} 0 & 0 \\ \frac{3g}{2\ell} \sin \theta & -\frac{3b}{m\ell^2} \end{pmatrix} \Big|_e \\ &= \begin{pmatrix} 0 & 0 \\ 0 & -\frac{3b}{m\ell^2} \end{pmatrix} \\ B &= \left. \frac{\partial f}{\partial u} \right|_e = \begin{pmatrix} \frac{\partial f_1}{\partial \tau} \\ \frac{\partial f_2}{\partial \tau} \end{pmatrix} \Big|_e \\ &= \begin{pmatrix} 0 \\ \frac{3}{m\ell^2} \end{pmatrix} \Big|_e \\ &= \begin{pmatrix} 0 \\ \frac{3b}{m\ell^2} \end{pmatrix}.\end{aligned}$$

Similarly, the output is given by $y = \theta = x_1 \stackrel{\triangle}{=} h(x, u)$, which implies that

$$\begin{aligned}C &= \left. \frac{\partial h}{\partial x} \right|_e = \begin{pmatrix} \frac{\partial h_1}{\partial \theta} & \frac{\partial h_1}{\partial \dot{\theta}} \end{pmatrix} \Big|_e \\ &= (1 \ 0) \Big|_e \\ &= (1 \ 0) \\ D &= \left. \frac{\partial h}{\partial u} \right|_e = 0.\end{aligned}$$

The linearized state space equations are therefore given by

$$\begin{aligned}\dot{\tilde{x}} &= \begin{pmatrix} 0 & 1 \\ \frac{3g \sin \theta_e}{2\ell} & -\frac{3b}{2\ell} \end{pmatrix} \tilde{x} + \begin{pmatrix} 0 \\ \frac{3}{m\ell^2} \end{pmatrix} \tilde{u} \\ \tilde{y} &= (1 \ 0) \tilde{x},\end{aligned}$$

which is similar to Equation (6.16) but with linearized state and output.

The transfer function can be found from the linearized state space equations using Equation (6.14) as

$$\begin{aligned}
 H(s) &= C(sI - A)^{-1}B + D \\
 &= (1 \ 0) \left(s \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ 0 & -\frac{3b}{m\ell^2} \end{pmatrix} \right)^{-1} \begin{pmatrix} 0 \\ \frac{3}{m\ell^2} \end{pmatrix} \\
 &= (1 \ 0) \begin{pmatrix} s & -1 \\ 0 & s + \frac{3b}{m\ell^2} \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \frac{3}{m\ell^2} \end{pmatrix} \\
 &= \frac{(1 \ 0) \begin{pmatrix} s + \frac{3b}{m\ell^2} & 1 \\ 0 & s \end{pmatrix} \begin{pmatrix} 0 \\ \frac{3}{m\ell^2} \end{pmatrix}}{s^2 + \frac{3b}{m\ell^2}s} \\
 &= \frac{\left(s + \frac{3b}{m\ell^2} \ 1 \right) \begin{pmatrix} 0 \\ \frac{3}{m\ell^2} \end{pmatrix}}{s^2 + \frac{3b}{m\ell^2}s} \\
 &= \frac{\frac{3}{m\ell^2}}{s^2 + \frac{3b}{m\ell^2}s},
 \end{aligned}$$

which is identical to Equation (5.2).

6.3 Design Study B. Inverted Pendulum



Homework Problem B.6

Defining the states as $\tilde{x} = (\tilde{z}, \tilde{\theta}, \dot{\tilde{z}}, \dot{\tilde{\theta}})^\top$, the input as $\tilde{u} = \tilde{F}$, and the measured output as $\tilde{y} = (\tilde{z}, \tilde{\theta})^\top$, find the linear state space equations in the form

$$\begin{aligned}
 \dot{\tilde{x}} &= A\tilde{x} + B\tilde{u} \\
 \tilde{y} &= C\tilde{x} + D\tilde{u}.
 \end{aligned}$$

Solution

The linear state space equations can be derived in two different ways: (1) directly from the linearized equations of motion, and (2) by linearizing the nonlinear equations of motion.

Starting with the linear state space equation in Equation (4.12) and solving

for $(\ddot{\tilde{z}}, \ddot{\tilde{\theta}})^\top$ gives

$$\begin{aligned} \begin{pmatrix} \ddot{\tilde{z}} \\ \ddot{\tilde{\theta}} \end{pmatrix} &= \frac{\begin{pmatrix} m_1 \frac{\ell^2}{4} & -m_1 \frac{\ell}{2} \\ -m_1 \frac{\ell}{2} & (m_1 + m_2) \end{pmatrix}}{m_1^2 \frac{\ell^2}{4} + m_1 m_2 \frac{\ell^2}{4} - m_1^2 \frac{\ell^2}{4}} \begin{pmatrix} -b \dot{\tilde{z}} + \tilde{F} \\ m_1 g \frac{\ell}{2} \tilde{\theta} \end{pmatrix} \\ &= \begin{pmatrix} -\frac{b}{m_2} \dot{\tilde{z}} + \frac{1}{m_2} \tilde{F} - \frac{m_1 g}{m_2} \tilde{\theta} \\ 2 \frac{b}{m_2 \ell} \dot{\tilde{z}} - \frac{2}{m_2 \ell} \tilde{F} + \frac{(m_1 + m_2) g}{m_2 \ell} \tilde{\theta} \end{pmatrix}. \end{aligned}$$

Therefore, defining $\tilde{x} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4)^\top \triangleq (\tilde{z}, \tilde{\theta}, \dot{\tilde{z}}, \dot{\tilde{\theta}})^\top$, and $\tilde{u} = \tilde{F}$ gives

$$\begin{aligned} \dot{\tilde{x}} &\triangleq \begin{pmatrix} \dot{\tilde{x}}_1 \\ \dot{\tilde{x}}_2 \\ \dot{\tilde{x}}_3 \\ \dot{\tilde{x}}_4 \end{pmatrix} = \begin{pmatrix} \dot{\tilde{z}} \\ \dot{\tilde{\theta}} \\ \ddot{\tilde{z}} \\ \ddot{\tilde{\theta}} \end{pmatrix} = \begin{pmatrix} \tilde{x}_3 \\ \tilde{x}_4 \\ -\frac{b}{m_2} \tilde{x}_3 + \frac{1}{m_2} \tilde{u} - \frac{m_1 g}{m_2} \tilde{x}_2 \\ \frac{2b}{m_2 \ell} \tilde{x}_3 - \frac{2}{m_2 \ell} \tilde{u} + \frac{2(m_1 + m_2) g}{m_2 \ell} \tilde{x}_2 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{m_1 g}{m_2} & -\frac{b}{m_2} & 0 \\ 0 & \frac{2(m_1 + m_2) g}{m_2 \ell} & \frac{2b}{m_2 \ell} & 0 \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{m_2} \\ -\frac{m_1 g}{m_2 \ell} \end{pmatrix} \tilde{u}. \end{aligned}$$

Assuming that the measured output of the system is $\tilde{y} = (\tilde{z}, \tilde{\theta})^\top$, the linearized output is given by

$$\tilde{y} = \begin{pmatrix} \tilde{z} \\ \tilde{\theta} \end{pmatrix} = \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tilde{u}.$$

Therefore, the linearized state space equations are given by

$$\begin{aligned} \dot{\tilde{x}} &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{m_1 g}{m_2} & -\frac{b}{m_2} & 0 \\ 0 & \frac{2(m_1 + m_2) g}{m_2 \ell} & \frac{2b}{m_2 \ell} & 0 \end{pmatrix} \tilde{x} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{m_2} \\ -\frac{m_1 g}{m_2 \ell} \end{pmatrix} \tilde{u} \\ \tilde{y} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tilde{x}. \end{aligned} \tag{6.17}$$

Alternatively, the linearized state space equations can be derived from the nonlinear equations of motion. Starting with Equation (3.2) and solving for

$(\ddot{z}, \ddot{\theta})^\top$ gives

$$\begin{aligned} \begin{pmatrix} \ddot{z} \\ \ddot{\theta} \end{pmatrix} &= \frac{\begin{pmatrix} m_1 \frac{\ell^2}{4} & -m_1 \frac{\ell}{2} \cos \theta \\ -m_1 \frac{\ell}{2} \cos \theta & (m_1 + m_2) \end{pmatrix}}{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 \theta} \begin{pmatrix} m_1 \frac{\ell}{2} \dot{\theta}^2 - b\dot{z} + F \\ m_1 g \ell \sin \theta \end{pmatrix} \\ &= \frac{\begin{pmatrix} m_1^2 \frac{\ell^2}{4} \dot{\theta}^2 \sin \theta - b m_1 \frac{\ell^2}{4} \dot{z} + m_1 \frac{\ell^2}{4} F - m_1^2 \frac{\ell^2}{4} g \sin \theta \cos \theta \\ -m_1^2 \frac{\ell^2}{4} \dot{\theta}^2 \sin \theta \cos \theta - b m_1 \frac{\ell^2}{4} \dot{z} \cos \theta - m_1 \frac{\ell}{2} \cos \theta F + (m_1 + m_2) m_1 g \frac{\ell}{2} \sin \theta \end{pmatrix}}{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 \theta}. \end{aligned}$$

Defining $x \triangleq (z, \theta, \dot{z}, \dot{\theta})^\top$, $u = F$, and $y = (z, \theta)^2$ results in the nonlinear state space equations

$$\begin{aligned} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} &= \begin{pmatrix} x_3 \\ x_4 \\ \frac{m_1^2 \frac{\ell^2}{4} x_4^2 \sin x_2 - b m_1 \frac{\ell^2}{4} x_3 + m_1 \frac{\ell^2}{4} u - m_1^2 \frac{\ell^2}{4} g \sin x_2 \cos x_2}{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 x_2} \\ \frac{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 x_2}{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 x_2} \end{pmatrix} \triangleq f(x, u) \\ y &= \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \triangleq h(x, u). \end{aligned}$$

Taking the Jacobians about the equilibrium $x_e = (z_e, 0, 0, 0)^\top$ gives

$$\begin{aligned} A \triangleq \left. \frac{\partial f}{\partial x} \right|_e &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{\partial f_3}{\partial x_2} & \frac{-b m_1 \frac{\ell^2}{4}}{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 x_2} & \frac{2 m_1^2 \frac{\ell}{2} x_4 \sin x_2}{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 x_2} \\ 0 & \frac{\partial f_4}{\partial x_2} & \frac{-b m_1 \frac{\ell}{2} \cos x_2}{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 x_2} & \frac{-2 m_1^2 \frac{\ell}{4} x_4 \sin x_2 \cos x_2}{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 x_2} \end{pmatrix} \Bigg|_e = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{m_1 g}{m_2} & \frac{-b}{m_2} & 0 \\ 0 & \frac{2(m_1 + m_2)g}{m_2 \ell} & \frac{2b}{m_2 \ell} & 0 \end{pmatrix} \\ B \triangleq \left. \frac{\partial f}{\partial u} \right|_e &= \begin{pmatrix} 0 \\ 0 \\ \frac{m_1 \frac{\ell^2}{4}}{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 x_2} \\ -\frac{m_1 \frac{\ell}{2} \cos \theta}{m_1 m_2 \frac{\ell^2}{4} + m_1^2 \frac{\ell^2}{4} \sin^2 x_2} \end{pmatrix} \Bigg|_e = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{m_2} \\ -\frac{m_2}{m_2 \ell} \end{pmatrix} \\ C \triangleq \left. \frac{\partial h}{\partial x} \right|_e &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \\ D \triangleq \left. \frac{\partial h}{\partial u} \right|_e &= \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \end{aligned}$$

resulting in the linearized state space equations given in Equation (6.17).

6.4 Design Study C. Satellite Attitude Control



Homework Problem C.6

Suppose that a star tracker is used to measure θ and a strain gage is used to approximate $\phi - \theta$. Defining the states as $x = (\theta, \phi, \dot{\theta}, \dot{\phi})^\top$, the input as $u = \tau$, and the measured output as $y = (\theta, \phi - \theta)^\top$, find the linear state space equations in the form

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du.\end{aligned}$$

Solution

The equations of motion from HW C.3 are given by

$$\begin{aligned}\ddot{\theta} &= \frac{1}{J_s}\tau - \frac{b}{J_s}\dot{\theta} + \frac{b}{J_s}\dot{\phi} - \frac{k}{J_s}\theta + \frac{k}{J_s}\phi \\ \ddot{\phi} &= \frac{b}{J_p}\dot{\theta} - \frac{b}{J_p}\dot{\phi} + \frac{k}{J_p}\theta - \frac{k}{J_p}\phi.\end{aligned}$$

Suppose that a star tracker is used to measure θ and an on-board strain sensor is used to measure $\psi - \theta$. Defining the state $x \triangleq (x_1, x_2, x_3, x_4)^\top = (\theta, \phi, \dot{\theta}, \dot{\phi})^\top$, the input $u \triangleq \tau$, and the output $y \triangleq (y_1, y_2)^\top = (\theta, \phi - \theta)^\top$, the state space equations are given by

$$\begin{aligned}\dot{x} &= \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ \dot{\phi} \\ \ddot{\theta} \\ \ddot{\phi} \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ \frac{1}{J_s}u - \frac{b}{J_s}x_3 + \frac{b}{J_s}x_4 - \frac{k}{J_s}x_1 + \frac{k}{J_s}x_2 \\ \frac{b}{J_p}x_3 - \frac{b}{J_p}x_4 + \frac{k}{J_p}x_1 - \frac{k}{J_p}x_2 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k}{J_p} & \frac{k}{J_s} & -\frac{b}{J_s} & \frac{b}{J_s} \\ \frac{k}{J_p} & -\frac{k}{J_s} & \frac{b}{J_p} & -\frac{b}{J_p} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{J_s} \\ 0 \end{pmatrix} u.\end{aligned}$$

Similarly

$$\begin{aligned}y &= \begin{pmatrix} \theta \\ \phi - \theta \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 - x_1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} u.\end{aligned}$$

CHAPTER 6. STATE SPACE MODELS

Therefore, the state space model is

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du,\end{aligned}$$

where

$$\begin{aligned}A &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k}{J_s} & \frac{k}{J_s} & -\frac{b}{J_s} & \frac{b}{J_s} \\ \frac{k}{J_p} & -\frac{k}{J_p} & \frac{b}{J_p} & -\frac{b}{J_p} \end{pmatrix} \\ B &= \begin{pmatrix} 0 \\ 0 \\ \frac{1}{J_s} \\ 0 \end{pmatrix} \\ C &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix} \\ D &= \begin{pmatrix} 0 \\ 0 \end{pmatrix}.\end{aligned}$$

Notes and References

There are numerous resources that deriving state space models from differential equations and transfer functions. For example [4, 17, 6, 5].

Part III

PID Control Design

In this part of the book we will introduce feedback control by studying the most commonly used industrial control law, the so-called proportional-integral-derivative control, or PID control. PID controllers are widely used because they are intuitive and easy to understand, they can be designed without any understanding of the underlying physics of the system, and a model of the system is not required, which is especially important in the chemical processing industry. However, stability and performance of the closed loop system can only be guaranteed when using PID control on second order systems. In Part IV on observer based control and Part V on loopshaping, we will extend the basic ideas covered in this Part to higher order systems.

Figure 6.1 shows a basic proportional control strategy. The output y of the physical system is subtracted from the commanded reference output y_r to produce the error $e = y_r - y$. Proportional control determines the input to the system u to be proportional to the error, i.e., $u = k_P e$. The constant gain k_P is called the proportional gain.

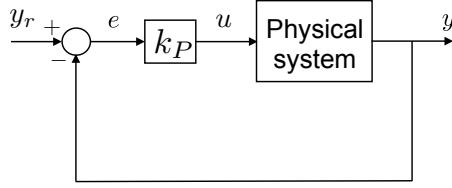


Figure 6.1: Proportional Control. The feedback control is a gain k_p multiplied by the error.

Proportional control can be augmented with integral control to produce PI control, as shown in Figure 6.2. The basic idea is to integrate the error so that the system responds to error accrued in the past. The constant gain k_I is called the integral gain.

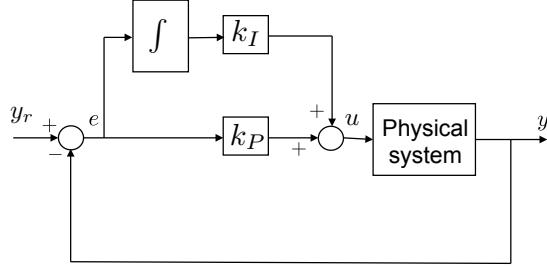


Figure 6.2: Proportional plus Integral (PI) Control. The feedback control is a linear combination of proportional and integral control.

Alternatively, proportional control can be augmented with a derivative control as shown in Figure 6.3. The idea is to respond to how fast the error is

changing, or the derivative of the error. The constant gain k_D is called the derivative gain.

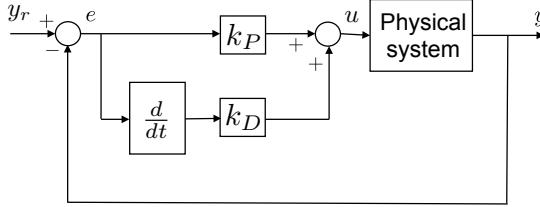


Figure 6.3: Proportional plus Derivative (PD) Control. The feedback control is a linear combination of proportional and derivative control.

When all three elements are used, as shown in Figure 6.4, the resulting controller is called a PID control. In Part III of this book we will explore various aspects of PID control. Many of these concepts can be generalized to other control schemes and so our study of PID control will also help to motivate and understand more advanced concepts.

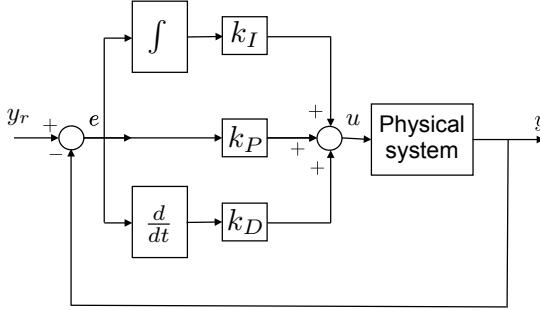


Figure 6.4: Proportional plus Integral plus Derivative (PID) Control. The feedback control is a linear combination of proportional and integral and derivative control.

Intuition for PID control can be gained by considering the step response shown in Figure 6.5. At the beginning of the response, there is significant error between the current output $y(t)$ and the reference command $y_r(t)$. The proportional control term will tend to push the system so as to reduce the gap between $y(t)$ and $y_r(t)$. As $y(t)$ begins to change, a fast response may lead to significant overshoot due to the momentum in the system. Therefore, if it is moving too quickly, we may want to slow down the response, or if it is moving too slowly, we may want to speed up the response. Derivative control is used to effect these changes. After the response has settled into steady state, there may be significant steady state error, as shown in Figure 6.5. Integral control adds up this steady state error and will eventually act to reduce the steady state

error. A good example is an aircraft in a constant cross wind. An integrator can be used to correct the response of the aircraft so that it crabs into the wind to maintain its desired flight path.

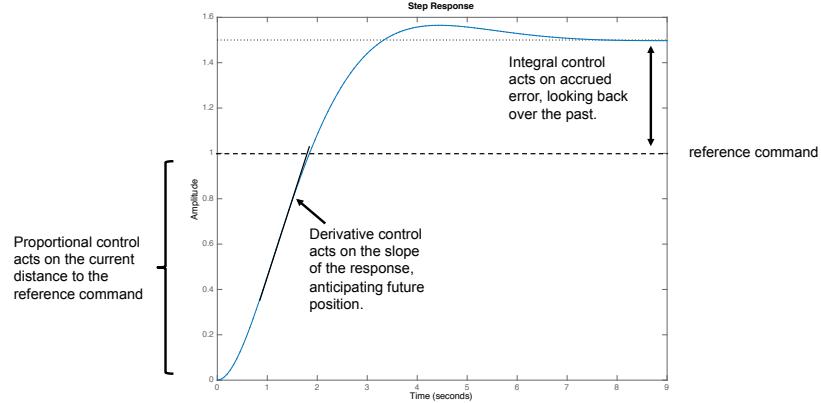


Figure 6.5: Intuitive interpretation of PID control. Proportional control acts on the current error, integral control acts on accrued past error, and derivative control acts on anticipated future error.

Part III of the book is organized as follows. In Chapter 7 we will show how PD control can be used to specify the desired closed loop poles of a second order system. In Chapter 8 we introduce the concepts of rise time, natural frequency, and damping ratio for second order systems, and show how these concepts relate to the location of the closed loop poles. We also show how these quantities can be used to design PD controllers for second order systems and cascades of second order systems. In Chapter 9 we use the final value theorem to show the effect of integrators on the closed loop system. We introduce the notion of *system type* in the context of reference tracking and disturbance rejection. Finally, in Chapter 10 we show how to write computer code to implement PID controllers.

Chapter 7

Pole Placement for Second Order Systems

7.1 Theory

Suppose that the model for the linearized physical system is given by the second order transfer function

$$P(s) = \frac{b_0}{s^2 + a_1 s + a_0}. \quad (7.1)$$

The *open loop poles* of the plant $P(s)$ are defined to be the roots of the *open loop characteristic polynomial*

$$\Delta_{ol}(s) = s^2 + a_1 s + a_0,$$

which are given by

$$p_{ol} = -\frac{a_1}{2} \pm \sqrt{\left(\frac{a_1}{2}\right)^2 - a_0}.$$

Note that the open loop poles are determined by the physical parameters, and that the poles may be in the right half plane. If we use PD control to regulate the output y to the reference command y_r , then the block diagram is shown in Figure 7.1.

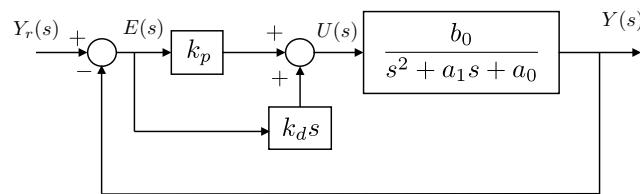


Figure 7.1: PD control of a second order system.

CHAPTER 7. POLE PLACEMENT FOR SECOND ORDER SYSTEMS

The transfer function for the closed loop system can be derived by noting from the block diagram that

$$\begin{aligned}
 Y(s) &= \left(\frac{b_0}{s^2 + a_1 s + a_0} \right) (k_P + k_D s) (Y_r(s) - Y(s)) \\
 \implies (s^2 + a_1 s + a_0) Y(s) &= (b_0 k_P + b_0 k_D s) (Y_r(s) - Y(s)) \\
 \implies [(s^2 + a_1 s + a_0) + (b_0 k_P + b_0 k_D s)] Y(s) &= (b_0 k_P + b_0 k_D s) Y_r(s) \\
 \implies (s^2 + (a_1 + b_0 k_D) s + (a_0 + b_0 k_P)) Y(s) &= (b_0 k_P + b_0 k_D s) Y_r(s) \\
 \implies Y(s) &= \frac{b_0 k_D s + b_0 k_P}{s^2 + (a_1 + b_0 k_D) s + (a_0 + b_0 k_P)} Y_r(s).
 \end{aligned} \tag{7.2}$$

Equation (7.2) constitutes the closed loop transfer function for the system under the influence of PD control. The *closed loop poles* are given by the roots of the *closed loop characteristic polynomial*

$$\Delta_{cl}(s) = s^2 + (a_1 + b_0 k_D) s + (a_0 + b_0 k_P),$$

which are

$$p_{cl} = -\frac{a_1 + b_0 k_D}{2} \pm \sqrt{\left(\frac{(a_1 + b_0 k_D)}{2}\right)^2 - (a_0 + b_0 k_P)}.$$

Note that the closed loop poles can be specified by the designer by selecting k_P and k_D . The basic idea is to select desired closed loop poles $-p_1^d$ and $-p_2^d$ and then to form the desired characteristic polynomial

$$\Delta_{cl}^d(s) = (s + p_1^d)(s + p_2^d) \stackrel{\triangle}{=} s^2 + \alpha_1 s + \alpha_0.$$

By setting $\Delta_{cl}(s) = \Delta_{cl}^d(s)$ we can equate the leading coefficients of each term of the polynomial and solve for the gains k_P and k_D as

$$\begin{aligned}
 k_P &= \frac{\alpha_0 - a_0}{b_0} \\
 k_D &= \frac{\alpha_1 - a_1}{b_0}.
 \end{aligned}$$

One of the disadvantages to the PD architecture shown in Figure 7.1 is the introduction of a zero in the closed loop transfer function. Note that while the open loop system (7.1) does not have a zero, the closed loop system (7.2) has a zero at

$$z_{cl} = -\frac{k_P}{k_D}.$$

The presence of the zero will modify the closed loop response of the system. A simple trick that removes the zero is to use the control structure shown in Figure 7.2. In this particular case, the differentiator acts only on the output y and not on the error e . Notice the change of sign on the signal coming from the differentiator.

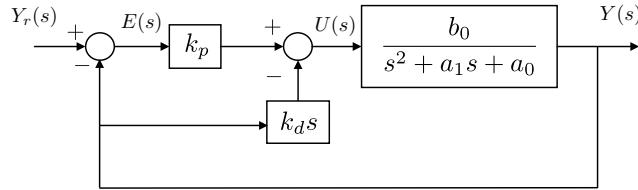


Figure 7.2: PD control of a second order system, where the derivative control only differentiates the output y and not the error e .

In this case the closed loop transfer function is calculated as

$$\begin{aligned}
 Y(s) &= \left(\frac{b_0}{s^2 + a_1 s + a_0} \right) (k_P(Y_r(s) - Y(s)) - k_D s Y(s)) \\
 \implies (s^2 + a_1 s + a_0)Y(s) &= (b_0 k_P(Y_r(s) - Y(s)) - b_0 k_D s Y(s)) \\
 \implies [(s^2 + a_1 s + a_0) + (b_0 k_P + b_0 k_D s)] Y(s) &= b_0 k_P Y_r(s) \\
 \implies (s^2 + (a_1 + b_0 k_D)s + (a_0 + b_0 k_P))Y(s) &= b_0 k_P Y_r(s) \\
 \implies Y(s) &= \frac{b_0 k_P}{s^2 + (a_1 + b_0 k_D)s + (a_0 + b_0 k_P)} Y_r(s).
 \end{aligned} \tag{7.3}$$

Note that the closed loop transfer function 7.3 has the same poles as Equation (7.2), but that the zero has been removed. An added benefit to the structure shown in Figure 7.2 is that when $y_r(t)$ is a step input, the derivative can introduce a large signal spike in $u(t)$. These large spikes are removed by only differentiating $y(t)$ instead of $e(t) = y_r(t) - y(t)$.

7.2 Design Study A. Single Link Robot Arm



Homework Problem A.7

- (a) Given the open loop transfer function found in problem A.6, find the open loop poles of the system, when the equilibrium angle is $\theta_e = 0$.
- (b) Using the PD control architecture shown in Figure 7.2, find the closed loop transfer function from θ_r to θ and find the closed loop poles as a function of k_P and k_D .
- (c) Select k_P and k_D to place the closed loop poles at $p_1 = -3$ and $p_2 = -4$.
- (d) Using the gains from part (c), implement the PD control for the single link robot arm in Simulink and plot the step response.

Hint: Change the s-function that defines the dynamics so that the output is the configuration variable θ . The Simulink block should look like that shown in Figure 7.3. Recall that the torque in the transfer function model

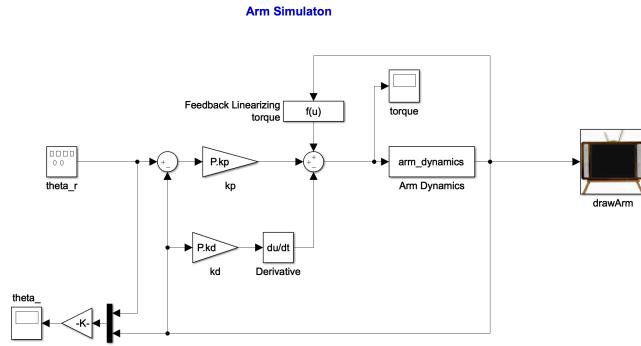


Figure 7.3: Simulink file for Homework A.8

is the feedback linearized torque $\tilde{\tau} = \tau - \tau_e$. Therefore, the torque applied to the robot arm is $\tau = \tau_e + \tilde{\tau}$. The feedback linearizing torque must be added to the torque computed by the PD controller as shown in Figure 7.3.

Solution

The open loop transfer function from homework A.6 is

$$\Theta(s) = \left(\frac{\frac{3}{m\ell^2}}{s^2 + \frac{3b}{m\ell^2}s} \right) \tilde{\tau}(s).$$

Using the parameters from Section A gives

$$\Theta(s) = \left(\frac{66.67}{s^2 + 0.667s} \right) \tilde{\tau}(s).$$

The open loop poles are therefore the roots of the open loop polynomial

$$\Delta_{ol}(s) = s^2 + 0.667s,$$

which are given by

$$p_{ol} = 0, -0.667.$$

Using PD control, the closed loop system is therefore shown in Figure 7.4.

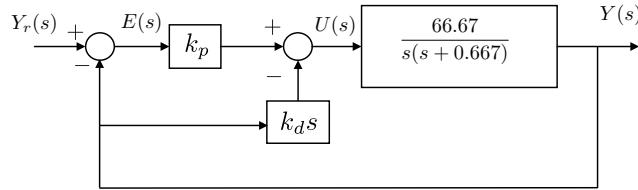


Figure 7.4: PD control of the single link robot arm.

The transfer function of the closed loop system is given by

$$\begin{aligned}
 \Theta(s) &= \left(\frac{66.67}{s^2 + 0.667s} \right) (k_P(\Theta^c(s) - \Theta(s)) - k_D s \Theta(s)) \\
 \implies (s^2 + 0.667s)\Theta(s) &= (66.67k_P(\Theta^c(s) - \Theta(s)) - 66.67k_D s \Theta(s)) \\
 \implies [(s^2 + 0.667s) + (66.67k_D s)]\Theta(s) &= 66.67k_P \Theta^c(s) \\
 \implies (s^2 + (0.667 + 66.67k_D)s + 66.67k_P)\Theta(s) &= 66.67k_P \Theta^c(s) \\
 \implies \Theta(s) &= \frac{66.67k_P}{s^2 + (0.667 + 66.67k_D)s + 66.67k_P} \Theta^c(s).
 \end{aligned}$$

Therefore, the closed loop poles are given by the roots of the closed loop characteristic polynomial

$$\Delta_{cl}(s) = s^2 + (0.667 + 66.67k_D)s + 66.67k_P$$

which are given by

$$p_{cl} = -\frac{(0.667 + 66.67k_D)}{2} \pm \sqrt{\left(\frac{(0.667 + 66.67k_D)}{2}\right)^2 - 66.67k_P}$$

If the desired closed loop poles are at -3 and -4 , then the desired closed loop characteristic polynomial is

$$\begin{aligned}
 \Delta_{cl}^d &= (s + 3)(s + 4) \\
 &= s^2 + 7s + 12.
 \end{aligned}$$

Equating the actual closed loop characteristic polynomial Δ_{cl} with the desired characteristic polynomial Δ_{cl}^d gives

$$s^2 + (0.667 + 66.67k_D)s + 66.67k_P = s^2 + 7s + 12,$$

or by equating each term we get

$$\begin{aligned}
 0.667 + 66.67k_D &= 7 \\
 66.67k_P &= 12.
 \end{aligned}$$

Solving for k_P and k_D gives

$$\begin{aligned}
 k_P &= 0.18 \\
 k_D &= 0.095.
 \end{aligned}$$

See <http://controlbook.byu.edu> for the complete solution.

Notes and References

The concept of using feedback to place the poles of the system at desired locations is a much deeper topic than is covered in this chapter. While the method discussed in this chapter is only applicable to second order systems, it is usually possible to arbitrarily assign the poles for an n^{th} order system. In general, to assign n poles will require n gains. We will discuss pole placement in more detail in Chapter 11 of this book.

Chapter 8

Design Strategies for Second Order Systems

8.1 Theory

In the previous section we saw that for second order systems, PD control could be used to exactly specify the pole location of the closed-loop system. This chapter examines the design problem of how to select the pole locations to achieve desired behavior. In Section 8.1.1 we derive the relationship between a single pole locations and the step response of the system. In Section 8.1.2 we show that the time domain response for a second order system can be understood in terms of natural frequency and damping ratio. Finally in Section 8.1.3 we show the effect that a zero has on the step response of a second order system.

8.1.1 First Order Response

Consider a first order system given by

$$Y(s) = \frac{p}{s + p} U(s),$$

where $p > 0$. The pole location of the first order system is at $-p$. If $U(s) = \frac{A}{s}$ is a step of magnitude A , then

$$Y(s) = \frac{Ap}{s(s + p)}.$$

As shown in Appendix c, the inverse Laplace transform is given by

$$y(t) = \begin{cases} A(1 - e^{-pt}), & t \geq 0 \\ 0, & \text{otherwise} \end{cases}. \quad (8.1)$$

The response is shown in Figure 8.2, which shows the response to steps of size $A = 1$, $A = 2$, and $A = 3$, respectively.

CHAPTER 8. DESIGN STRATEGIES FOR SECOND ORDER SYSTEMS

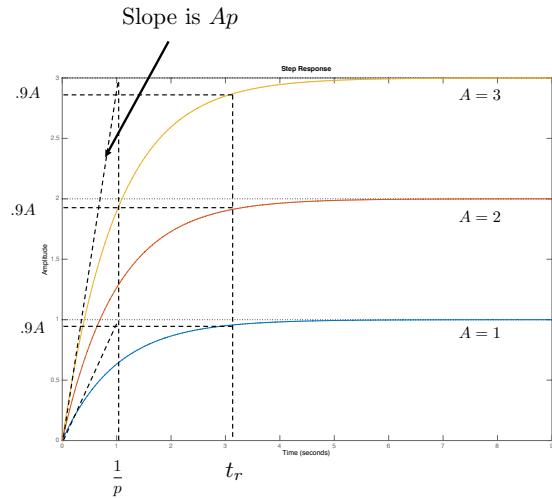


Figure 8.1: Step response of a first order system with pole p . Three step responses are shown, where the step size is $A = 1$, $A = 2$, and $A = 3$, respectively.

If we define the rise time t_r of the system to be the time after the step when the response reaches 90% of its final value, then note from Figure 8.2 that the rise time is independent of the step size A . Also, by differentiating Equation (8.1) at time $t = 0$, it is straightforward to show that the slope of the response at $t = 0$ is equal to Ap . This is also shown graphically in Figure 8.2.

When $A = 1$, the step response for different pole locations is shown in Figure 8.2. Pole locations corresponding to $p = 1$, $p = 2$, and $p = 4$ are shown. Note that as the pole location moves further into the left half of the complex plane, the rise time decreases.

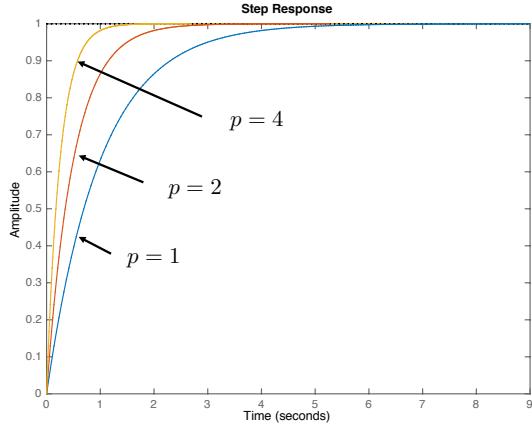


Figure 8.2: Step response of a first order system with poles $p = 1$, $p = 2$, and $p = 4$. The rise time increases as the pole location moves further into the left half of the complex plane.

An important concept for the steady state response of a system is the *DC-gain*, which is defined as follows.

Definition *The DC-gain of a transfer function $H(s)$ is*

$$H(s)|_{DC\text{-gain}} = \lim_{s \rightarrow 0} H(s).$$

An important fact is that if the poles of $H(s)$ are in the open left half plane, then the response of $H(s)$ to a step of size A approaches $A H(s)|_{DC\text{-gain}}$ as $t \rightarrow \infty$. This is true for first order systems, as well general $H(s)$. For example, the DC-gain of the first order system $H(s) = \frac{q}{s+p}$ is $\frac{q}{p}$, as long as $p > 0$.

8.1.2 Second Order Response

Suppose that the closed-loop system has a second order transfer function with two real poles and no zeros:

$$Y(s) = \frac{K}{(s + p_1)(s + p_2)} Y_r(s).$$

To find the output of the system when the input is a step of magnitude A , we find the inverse Laplace transform of

$$Y(s) = \frac{K}{(s + p_1)(s + p_2)} \frac{A}{s}.$$

Using partial fraction expansion we get

$$Y(s) = \frac{\frac{KA}{p_1 p_2}}{s} + \frac{\frac{KA}{(-p_1)(p_2 - p_1)}}{s + p_1} + \frac{\frac{KA}{(-p_2)(p_1 - p_2)}}{s + p_2}.$$

CHAPTER 8. DESIGN STRATEGIES FOR SECOND ORDER SYSTEMS

Taking the inverse Laplace transform gives

$$y(t) = \begin{cases} \frac{KA}{p_1 p_2} + \frac{KA}{(-p_1)(p_2-p_1)} e^{-p_1 t} + \frac{KA}{(-p_2)(p_1-p_2)} e^{-p_2 t}, & t \geq 0 \\ 0, & \text{otherwise} \end{cases}.$$

Note that as $t \rightarrow \infty$ that $y(t) \rightarrow \frac{KA}{p_1 p_2}$ which is equal to the step size A times the DC-gain of the system.

Now suppose that the closed loop system has two complex poles and no zeros. We will write a generic second order system with poles in the left half plane using the notation

$$H(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2},$$

where K is the DC-gain of the system (typically $K = 1$), $\omega_n > 0$ is called the natural frequency, and $\zeta > 0$ is called the damping ratio. The poles are given by the roots of the characteristic polynomial

$$\Delta(s) = s^2 + 2\zeta\omega_n s + \omega_n^2,$$

in other words, the poles are

$$\begin{aligned} p_{1,2} &= -\zeta\omega_n \pm \sqrt{(\zeta\omega_n)^2 - \omega_n^2} \\ &= -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}. \end{aligned}$$

If $0 < \zeta < 1$, then the roots are complex and given by

$$p_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1 - \zeta^2}. \quad (8.2)$$

Using the matlab code

```

1 figure(1), clf
2 zeta = 0.707;
3 for wn = [.5, 1, 2, 5];
4     G = tf(wn^2,[1,2*zeta*wn,wn^2]);
5     step(G)
6     hold on
7 end
8
9 figure(2), clf
10 wn = 1;
11 for zeta=[.2, .4, .707, 1, 2],
12     G = tf(wn^2,[1,2*zeta*wn,wn^2]);
13     step(G)
14     hold on
15 end

```

we obtain the second order response plots shown in Figures 8.3 and 8.4. Note from Figure 8.3 that as the natural frequency increases, the rise time decreases. We should note that a similar phenomena to Figure 8.2 also occurs, where the

CHAPTER 8. DESIGN STRATEGIES FOR SECOND ORDER SYSTEMS

rise time is independent of the size of the step. Note from Figure 8.4 that the damping ratio effects the amount of ringing in the system. For a small damping ratio, e.g., $\zeta = 0.2$ there is a large overshoot and significant ringing in the system. At the other extreme, when ζ is larger than one, the poles are real and the response is highly damped. The sweet spot is when $\zeta = 0.707$ where the rise time is small, but the overshoot is minimal.

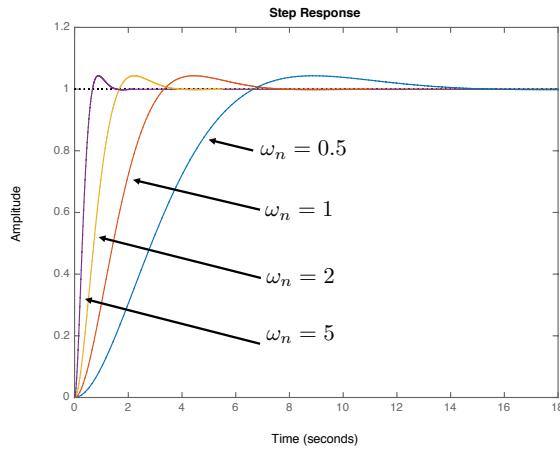


Figure 8.3: Second order response for $\omega_n = 0.5, 1, 2, 5$, and $\zeta = 0.707$.

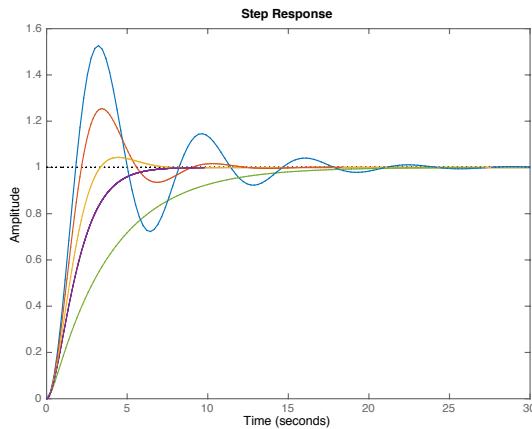


Figure 8.4: Second order response for $\omega_n = 1$, and $\zeta = 0.2, 0.4, 0.6, 0.707, 0.9, 1, 2$.

Note from Equation (8.2) that when $\zeta = 0.707 = \frac{1}{\sqrt{2}}$ that the poles are at

$$p_{1,2} = -\frac{\omega_n}{\sqrt{2}} \pm j \frac{\omega_n}{\sqrt{2}}.$$

In the complex plane, the pole locations are shown in Figure 8.5, where ω_n is the distance from origin to the poles. When $\zeta = \frac{1}{\sqrt{2}}$ the angle θ equals 45 degrees.

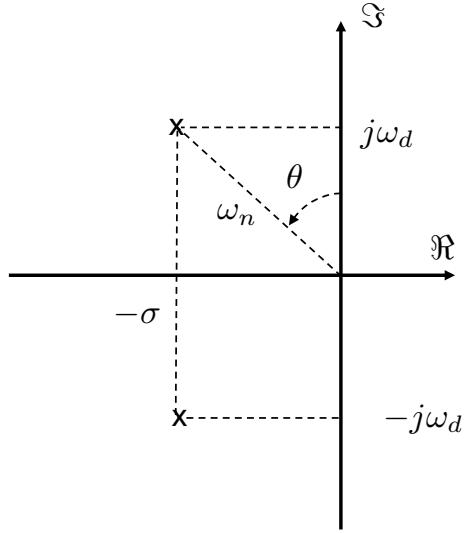


Figure 8.5: Complex conjugate poles in the left half plane.

In rectangular coordinates, the poles are given by

$$p_{1,2} = -\sigma \pm j\omega_d, \quad (8.3)$$

as shown in Figure 8.5. Equations (8.2) and (8.3) give

$$\begin{aligned} \sigma &= \zeta\omega_n \\ \omega_d &= \omega_n\sqrt{1-\zeta^2}. \end{aligned}$$

From Figure 8.5 it is also clear that

$$\sin \theta = \frac{\sigma}{\omega_n} = \zeta.$$

Therefore

$$\theta = \sin^{-1} \zeta.$$

Since the pole locations are at $-\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}$, the poles are both real if $\zeta \geq 1$. When $\zeta = 1$ the poles are both located at $-\omega_n$. When $0 < \zeta < 1$ the poles are imaginary and occur in complex conjugate pairs. As shown in Figure 8.6,

CHAPTER 8. DESIGN STRATEGIES FOR SECOND ORDER SYSTEMS

we say that the system is *over damped* when $\zeta > 1$, we say that the system is *critically damped* when $\zeta = 1$, and we say that the system is *under damped* when $0 < \zeta < 1$.

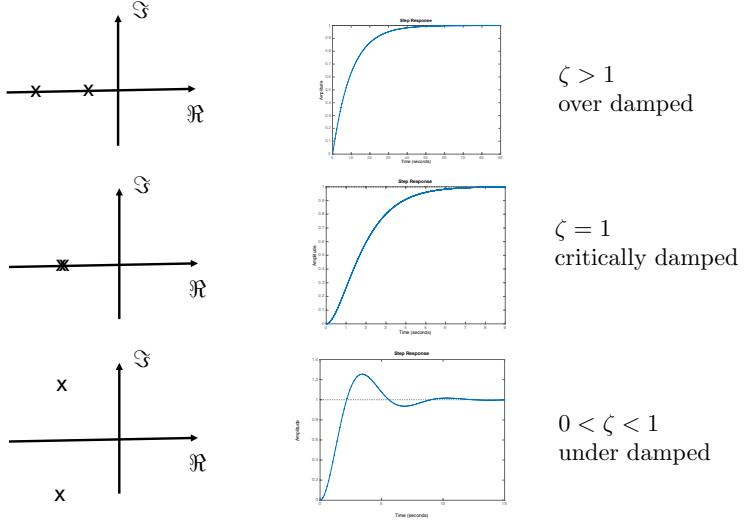


Figure 8.6: Second order response as a function of the damping ratio ζ .

The response of $H(s)$ to a step of size A is the inverse Laplace transform of

$$Y(s) = \frac{A\omega_n^2}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)},$$

which, as shown in Appendix c, when the poles are complex results in

$$y(t) = A \left[1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\sigma t} \cos \left(\omega_d t - \tan^{-1} \left(\frac{\sigma}{\omega_d} \right) \right) \right]. \quad (8.4)$$

A plot of this signal is shown in Figure 8.7.

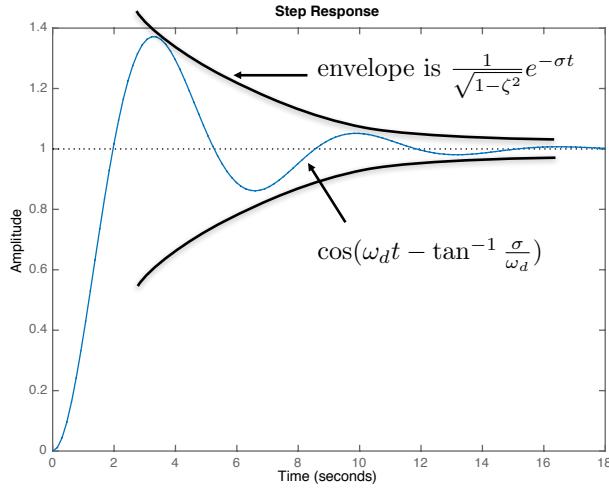


Figure 8.7: Second order response. The steady state value is A . The envelope is $\frac{1}{\sqrt{1-\zeta^2}}e^{-\sigma t}$, and the frequency and phase shift are given by ω_d and $\tan^{-1}\left(\frac{\sigma}{\omega_d}\right)$ respectively.

The rise time is defined to be time that it takes the output to transition from 10% to 90% of its final value A . A suitable approximation is that the rise time is one half of the peak time, which is the time when $y(t)$ reaches its first peak. To find the peak time t_p , differentiate $y(t)$ in Equation (8.4) and then solve for the first instant of time when the derivative is zero. The result is

$$t_p = \frac{\pi}{\omega_d} = \frac{\pi}{\omega_n \sqrt{1 - \zeta^2}}.$$

Therefore, the rise time is approximately

$$t_r \approx \frac{1}{2}t_p = \frac{1}{2} \frac{\pi}{\omega_n \sqrt{1 - \zeta^2}}.$$

When $\zeta = 0.707$ we get

$$t_r \approx \frac{2.2}{\omega_n}. \quad (8.5)$$

Specifications for second order systems are usually given either in terms of the rise time t_r and the damping ratio ζ , or in terms of the natural frequency ω_n and the damping ratio ζ .

8.1.3 Effect of a Zero on the Step Response

In the previous section, we looked at the step response when there were two complex poles and no zeros. In this section we will briefly discuss the effect of

CHAPTER 8. DESIGN STRATEGIES FOR SECOND ORDER SYSTEMS

zeros on the step response of the system. This is a topic that we could treat at a much deeper level.

A transfer function is called *rational* if it is the ratio of two polynomials in s . For example,

$$H(s) = \frac{s+1}{s^3 + 3s^2 + s + 4}$$

is a rational transfer function, whereas

$$H(s) = \frac{(s+1)e^{-s}}{s^3 + 3s^2 + s + 4}$$

is not. Rational transfer function can be written as

$$H(s) = \frac{N(s)}{D(s)},$$

where $N(s)$ is the numerator polynomial and $D(s)$ is the denominator polynomial. The *zeros* of the transfer function are roots of the equation $N(s) = 0$, and the poles of the system are the roots of the equation $D(s) = 0$.

One way to understand the effect of zeros is to consider the effect that zeros have on the partial fraction expansion of a transfer function. For a review of partial fraction expansion, see Appendix C. First consider a two pole system with characteristic equation given by $\Delta(s) = (s+2)(s+3)$. The transfer function with DC-gain equal to one, and without zeros is given by

$$H(s) = \frac{6}{(s+2)(s+3)}.$$

The response to a unit step is given by

$$\begin{aligned} y(t) &= \mathcal{L}^{-1} \left\{ \frac{6}{s(s+2)(s+3)} \right\} \\ &= \mathcal{L}^{-1} \left\{ \frac{1}{s} + \frac{-3}{s+2} + \frac{2}{s+3} \right\} \\ &= 1 - 3e^{-2t} + 2e^{-3t}, \quad t \geq 0. \end{aligned} \tag{8.6}$$

Now consider the step response when a zero has been added at $z = -2.1$ and the gain adjusted so that the DC-gain is still one, i.e., when

$$H(s) = \frac{\frac{6}{2.1}(s+2.1)}{(s+2)(s+3)}.$$

The response to a unit step is given by

$$\begin{aligned} y(t) &= \mathcal{L}^{-1} \left\{ \frac{\frac{6}{2.1}(s+2.1)}{s(s+2)(s+3)} \right\} \\ &= \mathcal{L}^{-1} \left\{ \frac{1}{s} + \frac{-0.1429}{s+2} + \frac{-0.8571}{s+3} \right\} \\ &= 1 - 0.1429e^{-2t} - 0.8571e^{-3t}, \quad t \geq 0. \end{aligned}$$

CHAPTER 8. DESIGN STRATEGIES FOR SECOND ORDER SYSTEMS

Note that $y(t)$ still settles to one, but the response of the pole at -2 has almost been removed. If however, the zeros is far away from the poles, then the effect is much less. Suppose that the zero is at $z = -60$ and the gain is adjusted so that the DC-gain is still one, i.e.,

$$H(s) = \frac{\frac{6}{60}(s + 60)}{(s + 2)(s + 3)}.$$

The response to a unit step is given by

$$\begin{aligned} y(t) &= \mathcal{L}^{-1} \left\{ \frac{\frac{6}{60}(s + 60)}{s(s + 2)(s + 3)} \right\} \\ &= \mathcal{L}^{-1} \left\{ \frac{1}{s} + \frac{-2.9}{s + 2} + \frac{1.9}{s + 3} \right\} \\ &= 1 - 2.9e^{-2t} + 1.9e^{-3t}, \quad t \geq 0. \end{aligned}$$

In this case, the response is similar to Equation (8.6).

Another way to understand the effect of zeros on the system is to recall that an s in the denominator is essentially a differentiator. Therefore

$$H(s) = \frac{\omega_n^2(\tau s + 1)}{s^2 + 2\zeta\omega_n s + \omega_n^2} = \tau \frac{\omega_n^2 s}{s^2 + 2\zeta\omega_n s + \omega_n^2} + \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}.$$

Therefore the step response is given by

$$\begin{aligned} y(t) &= \mathcal{L}^{-1} \left\{ \tau \frac{\omega_n^2 s}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)} + \frac{\omega_n^2}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)} \right\} \\ &= \tau \frac{d}{dt} \mathcal{L}^{-1} \left\{ \frac{\omega_n^2}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)} \right\} + \mathcal{L}^{-1} \left\{ \frac{\omega_n^2}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)} \right\}. \end{aligned}$$

Therefore, the effect of the zero is to add τ times the derivative of the step response to what the step response would have been without the zero. Figure 8.8 shows the step response for different values of τ , where it can be seen that adding a zero has a strong effect on the rise time and the damping in the response.

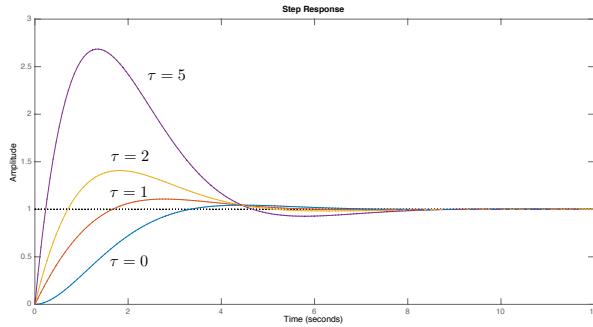


Figure 8.8: Step response of a second order system to a single zero.

8.1.4 Successive Loop Closure

The natural frequency ω_n and the damping ratio ζ are only defined for second order systems, and therefore, the PD design technique based on ω_n and ζ are only valid for second order systems. However, these concepts can be extended to higher order systems through a technique called successive loop closure, or inner-loop, outer-loop design. When we derived the Laplace transform in Chapter 5 we saw that for many systems, the transfer functions can be arranged in cascade form as shown in Figure 8.9 where the input to the system is u_1 and the output is y_2 , but where y_1 and u_2 are intermediary values that represent the natural physical coupling between internal variables. Many physical systems can be represented by a cascade of systems. When the system is represented as a cascade,

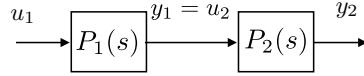


Figure 8.9: Open-loop transfer function modeled as a cascade of two transfer functions.

the control strategy can be designed using *successive loop closure*. The basic idea is to close feedback loops in succession around the open-loop plant dynamics rather than designing a single control system. To illustrate this approach, consider the open-loop system shown in Figure 8.9. The open-loop dynamics are given by the product of two transfer functions in series: $P(s) = P_1(s)P_2(s)$. We assume that each of the transfer functions has an output (y_1, y_2) that can be measured and used for feedback. Typically, each of the transfer functions, $P_1(s)$, and $P_2(s)$, is of relatively low order – usually first or second order. In this case, we are interested in controlling the output y_2 . Instead of closing a single feedback loop around y_2 , we will close feedback loops around y_1 , and y_2 in succession as shown in Figure 8.10.

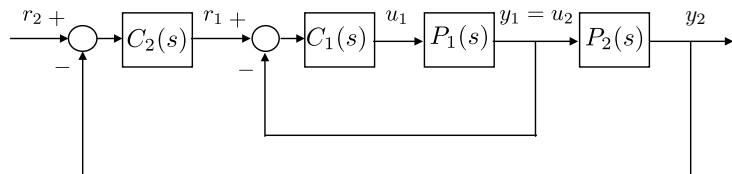


Figure 8.10: Successive loop closure design.

We will design the compensators $C_1(s)$, and $C_2(s)$ in succession. A necessary condition in the design process is that the inner loop must be much faster than each successive loop. As a rule of thumb, if the rise time of the inner loop is t_{r_1} , the rise time of the next loop should be 5 to 10 times longer, i.e., $t_{r_2} > W_1 t_{r_1}$, where W_1 is a design parameter, usually on the order of 5 – 10. In part V of this book we will talk about the frequency response of the system. At that time,

we will provide a deeper explanation which can be understood in terms of the bandwidth of the each successive loop.

Examining the inner loop shown in Figure 8.10, the goal is to design a closed-loop system from r_1 to y_1 having a rise time of t_{r_1} . If the rise time of the inner loop is significantly faster than the rise time of the outer loop, and if the DC-gain of the inner loop is k_{DC_1} , then relative to the outer loop the inner loop can be effectively modeled as the DC-gain. This is depicted schematically in Figure 8.11. With the inner-loop transfer function modeled as its DC-gain, design of the outer loop is simplified because it only includes the plant transfer function $P_2(s)$ and the compensator $C_2(s)$ and the DC-gain k_{DC_1} . Because each

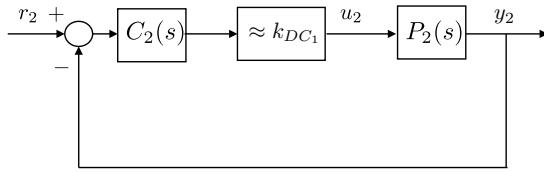


Figure 8.11: Successive loop closure design with inner loop modeled as a unity gain.

of the plant models $P_1(s)$, and $P_2(s)$ are first or second order, conventional PID compensators can be employed. The loopshaping techniques discussed in Part V can also be used for successive loop closure of higher order systems.

8.1.5 Input Saturation and Limits of Performance

A natural question is to ask if the rise time can be made arbitrarily small. Unfortunately, saturation limits on the actuators place a lower bound on the achievable rise time. All physical systems have saturation limit on their actuators. For example, in the robot arm problem, the torque that can be applied to the arm is limited by the motor that is used to apply that torque. Force and torque will always be limited by physical constraints like current limits. Similarly, for an airplane, the rudder command is limited by how far the rudder can physically move, which is typically on the order of ± 40 degrees. A natural way to model input constraints is to add a saturation block preceding the physical plant. Figure 8.12 shows a saturation block in conjunction with a PD control scheme. Mathematically, the saturation block is given by

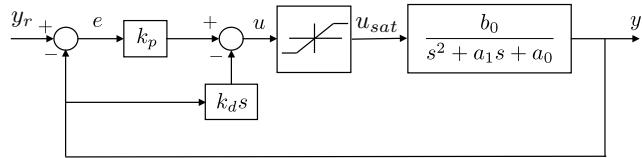


Figure 8.12: Control system with input saturation. The input saturation places a lower bound on the achievable rise time.

$$u_{sat} = \begin{cases} u_{max} & \text{if } u \geq u_{max} \\ u, & \text{if } -u_{max} \leq u \leq u_{max} \\ -u_{max} & \text{otherwise} \end{cases}$$

where the input to the block is u , the output is u_{sat} , and the saturation limit is u_{max} . Note that the saturation block is non-linear. Therefore, one design strategy is to select the feedback gains so that the system remains linear by keeping the input out of saturation.

To see how this might be done, consider the second-order system shown in Figure 8.12 with proportional feedback on the output error and derivative feedback on the output. When the input is not saturated, the closed-loop transfer function is

$$\frac{y}{y_r} = \frac{b_0 k_p}{s^2 + (a_1 + b_0 k_d)s + (a_0 + b_0 k_p)}. \quad (8.7)$$

We can see that the closed-loop poles of the system are defined by the selection of the control gains k_p and k_d . Note also that the actuator effort u can be expressed as $u = k_p e - k_d \dot{y}$. When \dot{y} is zero or small, the size of the actuator effort u is primarily governed by the size of the control error e and the control gain k_p . If the system is stable, the largest control effort in response to a step input will occur immediately after the step when $\dot{y} = 0$. If the system is to be kept just out of saturation, then immediately after the step we have $|u| = u_{max} = k_p e_{max}$. Rearranging this expression, we find that the proportional control gain can be determined from the maximum anticipated output error and the saturation limits of the actuator as

$$k_p = \pm \frac{u_{max}}{e_{max}}, \quad (8.8)$$

where u_{max} is the maximum control effort the system can provide, and e_{max} is the step error that results from a step input of nominal size, and the sign of k_p is determined by the physics of the system.

Suppose that the desired closed loop characteristic polynomial is given by

$$\Delta_{cl}^d(s) = s^2 + 2\zeta\omega_n s + \omega_n^2,$$

then by comparing with the actual closed loop characteristic polynomial

$$\Delta_{cl}(s) = s^2 + (a_1 + b_0 k_d)s + (a_0 + b_0 k_p)$$

we have that

$$\begin{aligned} \omega_n &= \sqrt{a_0 + b_0 k_p} \\ &\leq \sqrt{a_0 + b_0 \frac{u_{max}}{e_{max}}}, \end{aligned}$$

which is an upper limit on the natural frequency of the closed-loop system, ensuring that saturation of the actuator is avoided. The corresponding rise time constraint is

$$t_r \geq \frac{2.2}{\omega_n} = \frac{2.2}{\sqrt{a_0 + b_0 \frac{u_{max}}{e_{max}}}},$$

which is the smallest possible rise time for an error step of e_{\max} .

If the input has an equilibrium value, then the saturation constraint used in the above calculation must be modified to account for the equilibrium. For example, suppose that $u = u_e + \tilde{u}$ where u_e is the equilibrium value and \tilde{u} is the output of the PID controller. Also suppose that the input saturation constraint is given by $u_{\min} \leq u \leq u_{\max}$. The objective is to find a value for \tilde{u}_{\max} that $|\tilde{u}| \leq \tilde{u}_{\max}$ guarantees that $u_{\min} \leq u \leq u_{\max}$. We have that

$$\begin{aligned} u_{\min} &\leq u \leq u_{\max} \\ \implies u_{\min} &\leq u_e + \tilde{u} \leq u_{\max} \\ \implies u_{\min} - u_e &\leq \tilde{u} \leq u_{\max} - u_e. \end{aligned}$$

The desired value for \tilde{u}_{\max} is therefore the minimum (in absolute value) of the right and left sides of this expression. The situation is shown in Figure 8.13, where it is clear that the value for \tilde{u}_{\max} can be written as

$$\tilde{u}_{\max} = \begin{cases} |u_{\max} - u_e|, & \text{if } u_e \geq \frac{u_{\max} + u_{\min}}{2} \\ |u_e - u_{\min}|, & \text{if } u_e \leq \frac{u_{\max} + u_{\min}}{2}. \end{cases}$$

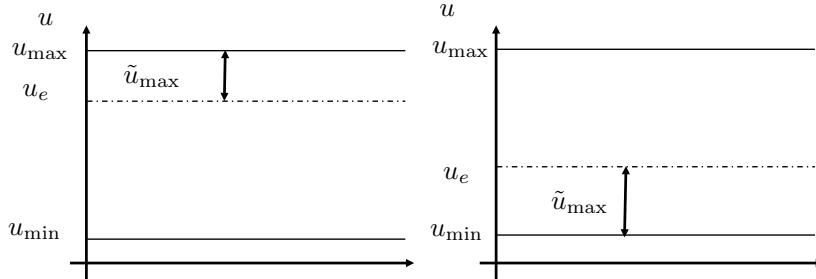


Figure 8.13: The saturation constraint for \tilde{u} depends on the value of the equilibrium input u_e .

The take-away from this discussion is that the rise time is limited by the input saturation constraint and cannot be made arbitrarily small. In practice, the control parameters are tuned so that the closed loop system has the fastest possible rise time without driving the plant input into saturation.

8.2 Design Study A. Single Link Robot Arm



Homework Problem A.8

For the single link robot arm, do the following:

- (a) Suppose that the design requirements are that the rise time is $t_r \approx 0.8$ seconds, with a damping ratio of $\zeta = 0.707$. Find the desired closed loop characteristic polynomial $\Delta_{cl}^d(s)$, and the associated pole locations. Find the proportional and derivative gains k_P and k_D to achieve these specifications, and modify the Simulink simulation from HW A.8 to verify that the step response satisfies the requirements.
- (b) Suppose that the input torque for the robot arm is limited to $|\tau| \leq \tau_{\max} = 1$ Nm. Modify the Simulink diagram to include a saturation block on the torque τ . Using the rise time t_r and damping ratio ζ as tuning parameters, tune the PD control law so that the input just saturates when a step of size 50 degrees is placed on $\tilde{\theta}^r$.

Solution

A rise time of $t_r \approx 0.8$ seconds, implies that the natural frequency is

$$\omega_n = 2.2/0.8 = 2.75.$$

Therefore, the desired characteristic polynomial is

$$\Delta_{cl} = s^2 + 2\zeta\omega_n s + \omega_n^2 = s^2 + 3.885s + 7.5625. \quad (8.9)$$

From HW A.8 the actual closed loop characteristic polynomial is

$$\Delta_{cl}(s) = s^2 + (0.667 + 66.67k_D)s + 66.67k_P. \quad (8.10)$$

Equating Equations (8.9) and (8.10) and solving for the gains gives

$$k_P = 0.1134$$

$$k_D = 0.0483.$$

See <http://controlbook.byu.edu> for the complete solution.

8.3 Design Study B. Inverted Pendulum



Homework Problem B.8

For the inverted pendulum, do the following:

- (a) Using the principle of successive loop closure, draw a block diagram that uses PD control for both inner loop control and outer loop control. The input to the outer loop controller is the desired cart position z^d and the output of the controller is the desired pendulum angle θ^d . The input to the inner loop controller is the desired pendulum angle θ^d and the output is the force F on the cart.

- (b) Focusing on the inner loop, find the PD gains k_{P_θ} and k_{D_θ} so that the rise time of the inner loop is $t_{r_\theta} = 0.5$ seconds, and the damping ratio is $\zeta_\theta = 0.707$.
- (c) Find the DC gain k_{DC_θ} of the inner loop.
- (d) Replacing the inner loop by its DC-gain, find the PD gains k_{P_z} and k_{D_z} so that the rise time of the outer loop is $t_{r_z} = 10t_{r_\theta}$ and the damping ratio is $\zeta_z = 0.707$.
- (e) Implement the successive loop closure design for the inverted pendulum in Simulink where the commanded cart position is given by a square wave with magnitude 0.5 meters and frequency 0.01 Hz.
- (f) Suppose that the size of the input force on the cart is limited to $F_{\max} = 5$ N. Modify the Simulink diagram to include a saturation block on the force F . Using the rise time of the outer loop, tune the PD control law to get the fastest possible response without input saturation when a step of size 1 meter is placed on \tilde{z}^r .

Solution

The block diagram for the inner loop is shown in Figure 8.14.

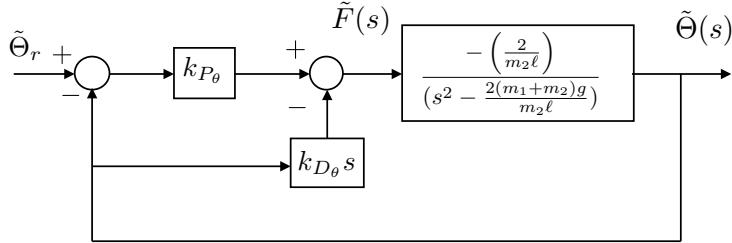


Figure 8.14: Block diagram for inner loop of inverted pendulum control

The closed loop transfer function from $\tilde{\Theta}^d$ to $\tilde{\Theta}$ is given by

$$\tilde{\Theta}(s) = \frac{-\frac{2k_{P_\theta}}{m_2\ell}}{s^2 - \frac{2k_{D_\theta}}{m_2\ell}s - \left(\frac{2(m_1+m_2)g}{m_2\ell} + \frac{2k_{P_\theta}}{m_2\ell}\right)} \tilde{\Theta}^d(s).$$

Therefore the closed loop characteristic equation is

$$\Delta_{cl}(s) = s^2 - \frac{2k_{D_\theta}}{m_2\ell}s - \left(\frac{2(m_1+m_2)g}{m_2\ell} + \frac{2k_{P_\theta}}{m_2\ell}\right).$$

The desired closed loop characteristic equation is

$$\Delta_{cl}^d(s) = s^2 + 2\zeta_\theta\omega_{n_\theta}s + \omega_{n_\theta}^2,$$

where

$$\begin{aligned}\omega_{n_\theta} &= \frac{2.2}{t_{r_\theta}} = 4.4 \\ \zeta_\theta &= 0.707.\end{aligned}$$

Therefore

$$\begin{aligned}k_{P_\theta} &= -(m_1 + m_2)g - \frac{m_2\ell\omega_{n_\theta}^2}{2} = -17.09 \\ k_{D_\theta} &= -\zeta_\theta\omega_{n_\theta}m_2\ell = -1.5554.\end{aligned}$$

The DC gain of the inner loop is given by

$$k_{DC_\theta} = \frac{k_{P_\theta}}{(m_1 + m_2)g + k_{P_\theta}} = 3.531.$$

Replacing the inner loop by its DC gain, the block diagram for the outer loop is shown in Figure 8.15.

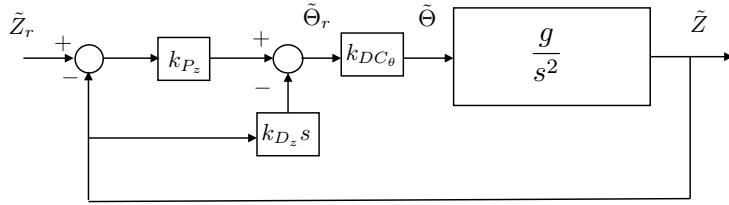


Figure 8.15: Block diagram for outer loop of inverted pendulum control

The closed loop transfer function from z^d to z is given by

$$Z(s) = \frac{gk_{DC_\theta}k_{P_z}}{s^2 + gk_{DC_\theta}k_{D_z}s + gk_{DC_\theta}k_{P_z}} Z^d(s).$$

Note that the DC gain for the outer loop is equal to one. The closed loop characteristic equation is therefore

$$\Delta_{cl}(s) = s^2 + gk_{DC_\theta}k_{D_z}s + gk_{DC_\theta}k_{P_z}.$$

The desired characteristic equation is

$$\Delta_{cl}^d(s) = s^2 + 2\zeta_z\omega_{n_z}s + \omega_{n_z}^2,$$

where

$$\begin{aligned}t_{r_z} &= 10t_{r_\theta} = 5 \\ \omega_{n_z} &= \frac{2.2}{t_{r_z}} = 0.44 \\ \zeta_z &= 0.707.\end{aligned}$$

The PD gains are therefore

$$k_{P_z} = \frac{\omega_{n_z}^2}{gk_{DC_z}} = 0.0056$$

$$k_{D_z} = \frac{2\zeta_z \omega_{n_z}}{gk_{DC_\theta}} = 0.1134.$$

See <http://controlbook.byu.edu> for the complete solution.

8.4 Design Study C. Satellite Attitude Control



Homework Problem C.8

- (a) For simple satellite system, using the principle of successive loop closure, draw a block diagram that uses PD control for the inner and the outer loop control. The input to the outer loop controller is the desired angle of the solar panel ϕ_r and the output is the desired angle of the satellite θ_r . The input to the inner loop controller is θ_r and the output is the torque on the satellite τ .
- (b) Focusing on the inner loop, find the PD gains k_{P_θ} and k_{D_θ} so that the rise time of the inner loop is $t_{r_\theta} = 2$ seconds, and the damping ratio is $\zeta_\theta = 0.707$.
- (c) Find the DC gain k_{DC_θ} of the inner loop.
- (d) Replacing the inner loop by its DC-gain, find the PD gains k_{P_ϕ} and k_{D_ϕ} so that the rise time of the outer loop is $t_{r_\phi} = 10t_{r_\theta}$.
- (e) Find the DC gain k_{DC_ϕ} of the output loop. If the DC gain is not equal to unity, then ϕ_r will need to be premultiplied by $1/k_{DC_\phi}$.
- (f) Implement the successive loop closure design for the satellite system in Simulink where the commanded solar panel angle is given by a square wave with magnitude 15 degrees and frequency 0.015 Hz.
- (g) Suppose that the size of the input torque on the satellite is limited to $\tau_{\max} = 5$ Nm. Modify the Simulink diagram to include a saturation block on the torque τ . Using the rise time of the outer loop as a tuning parameter, tune the PD control law to get the fastest possible response without input saturation when a step of size 30 degrees is placed on ϕ^r .

Solution

The block diagram for the inner loop is shown in Figure 8.16.

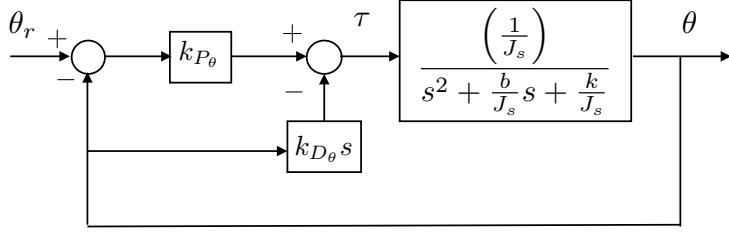


Figure 8.16: Block diagram for inner loop of satellite control

The closed loop transfer function from Θ^d to Θ is given by

$$\Theta(s) = \frac{\frac{k_{P_\theta}}{J_s}}{s^2 + \left(\frac{b+k_{D_\theta}}{J_s}\right)s + \left(\frac{k+k_{P_\theta}}{J_s}\right)} \Theta^d(s).$$

Therefore the closed loop characteristic equation is

$$\Delta_{cl}(s) = s^2 + \left(\frac{b+k_{D_\theta}}{J_s}\right)s + \left(\frac{k+k_{P_\theta}}{J_s}\right).$$

The desired closed loop characteristic equation is

$$\Delta_{cl}^d(s) = s^2 + 2\zeta_\theta \omega_{n_\theta} s + \omega_{n_\theta}^2,$$

where

$$\begin{aligned} \omega_{n_\theta} &= \frac{2.2}{t_{r_\theta}} = 1.1 \\ \zeta_\theta &= 0.707. \end{aligned}$$

Therefore

$$\begin{aligned} k_{P_\theta} &= \omega_{n_\theta}^2 J_s - k = 5.9 \\ k_{D_\theta} &= 2\zeta_\theta \omega_{n_\theta} J_s - b = 7.727. \end{aligned}$$

The DC gain of the inner loop is given by

$$k_{DC_\theta} = \frac{k_{P_\theta}}{k + k_{P_\theta}} = 0.9752.$$

Replacing the inner loop by its DC gain, the block diagram for the outer loop is shown in Figure 8.17.

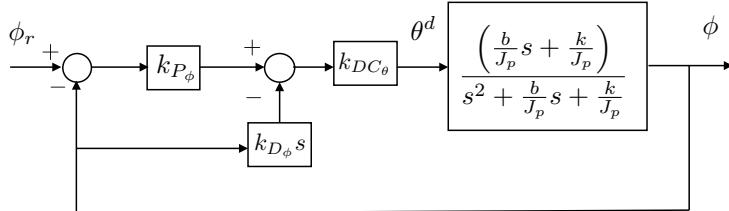


Figure 8.17: Block diagram for outer loop of satellite attitude control

CHAPTER 8. DESIGN STRATEGIES FOR SECOND ORDER SYSTEMS

To find the closed loop transfer function from ϕ^d to ϕ , follow the loop backwards from ϕ to obtain

$$\Phi(s) = \left(\frac{\frac{b}{J_p}s + \frac{k}{J_p}}{s^2 + \frac{b}{J_p}s + \frac{k}{J_p}} \right) [k_{DC_\theta} k_{P_\theta} (\Phi_r - \Phi) - k_{DC_\theta} k_{D_\theta} s \Phi].$$

After some manipulation we get

$$\begin{aligned} [(J_p + bk_{DC_\theta} k_{D_\theta})s^2 + (b + bk_{DC_\theta} k_{P_\theta} + kk_{DC_\theta} k_{D_\theta})s + (k + kk_{DC_\theta} k_{P_\theta})] \Phi \\ = [bk_{DC_\theta} k_{P_\theta} s + kk_{DC_\theta} k_{P_\theta}] \Phi_r, \end{aligned} \quad (8.11)$$

which results in the monic transfer function

$$\Phi(s) = \frac{\left(\frac{bk_{DC_\theta} k_{P_\theta}}{J_p + bk_{DC_\theta} k_{D_\theta}} \right) s + \left(\frac{kk_{DC_\theta} k_{P_\theta}}{J_p + bk_{DC_\theta} k_{D_\theta}} \right)}{s^2 + \left(\frac{b + bk_{DC_\theta} k_{P_\theta} + kk_{DC_\theta} k_{D_\theta}}{J_p + bk_{DC_\theta} k_{D_\theta}} \right) s + \left(\frac{k + kk_{DC_\theta} k_{P_\theta}}{J_p + bk_{DC_\theta} k_{D_\theta}} \right)}.$$

The desired closed loop characteristic equation for the outer loop is

$$\Delta_{cl}^d(s) = s^2 + 2\zeta_\phi \omega_{n_\phi} s + \omega_{n_\phi}^2,$$

where

$$t_{r_\phi} = 10t_{r_\theta} = 20$$

$$\omega_{n_\phi} = \frac{2.2}{t_{r_\phi}} = 0.11$$

$$\zeta_\phi = 0.707.$$

Therefore the gains k_{P_ϕ} and k_{P_θ} satisfy

$$\begin{aligned} \frac{k + kk_{DC_\theta} k_{P_\theta}}{J_p + bk_{DC_\theta} k_{D_\theta}} &= \omega_{n_\phi}^2 \\ \frac{b + bk_{DC_\theta} k_{P_\theta} + kk_{DC_\theta} k_{D_\theta}}{J_p + bk_{DC_\theta} k_{D_\theta}} &= 2\zeta_\phi \omega_{n_\phi}. \end{aligned}$$

Expressing these equations in matrix form gives

$$\begin{pmatrix} kk_{DC_\theta} & -J_p bk_{DC_\theta} \\ bk_{DC_\theta} & kk_{DC_\theta} - 2bk_{DC_\theta} \zeta_\phi \omega_{n_\phi} \end{pmatrix} \begin{pmatrix} k_{P_\phi} \\ k_{D_\phi} \end{pmatrix} = \begin{pmatrix} -k + J_p \omega_{n_\phi}^2 \\ -b + 2J_p \zeta_\phi \omega_{n_\phi} \end{pmatrix},$$

implying that

$$\begin{pmatrix} k_{P_\phi} \\ k_{D_\phi} \end{pmatrix} = \begin{pmatrix} kk_{DC_\theta} & -J_p bk_{DC_\theta} \\ bk_{DC_\theta} & kk_{DC_\theta} - 2bk_{DC_\theta} \zeta_\phi \omega_{n_\phi} \end{pmatrix}^{-1} \begin{pmatrix} -k + J_p \omega_{n_\phi}^2 \\ -b + 2J_p \zeta_\phi \omega_{n_\phi} \end{pmatrix} = \begin{pmatrix} -0.6168 \\ 0.9778 \end{pmatrix}$$

The DC gain of the inner loop is given by

$$k_{DC_\phi} = \frac{kk_{DC_\theta} k_{P_\phi}}{k + kk_{DC_\theta} k_{P_\phi}} = -1.5093.$$

Since the DC gain of the outer loop is not equal to one, ϕ_r in Figure 8.17 must be pre-multiplied by $1/k_{DC_\phi}$.

See <http://controlbook.byu.edu> for the complete solution.

CHAPTER 8. DESIGN STRATEGIES FOR SECOND ORDER SYSTEMS

Notes and References

Chapter 9

System Type and Integrators

9.1 Theory

In the previous chapter we explored design techniques for second order system where the feedback controller was a PD-type control. Since second order systems have two degrees of freedom, their transient response can be completely determined by the proportional and derivative gains. However, from the simulation results it can easily be observed that if there is a disturbance on the system, or if controller does not know the system parameters exactly, then there will be a steady state error in the output. The steady state error can be effectively removed using an integrator in the control law. In this chapter we will explore these ideas in more detail and explain when and how integral control can be used to enhance steady state tracking and to reduce the effect of disturbances.

9.1.1 Final Value Theorem

The key theoretical tool to understanding the steady state error and the effect of integrators is the final value theorem, which we state and prove below.

Final Value Theorem.

Suppose that $z(t)$ and $Z(s)$ are Laplace transform pairs, and suppose that $\lim_{t \rightarrow \infty} z(t)$ is finite, then

$$\lim_{t \rightarrow \infty} z(t) = \lim_{s \rightarrow 0} sZ(s).$$

Proof.

From the fundamental theorem of calculus we have that for $t \geq 0$

$$z(t) - z(0) = \int_0^t dz.$$

Therefore

$$\begin{aligned}
 \lim_{t \rightarrow \infty} z(t) &= z(0) + \lim_{t \rightarrow \infty} \int_0^t \frac{dz}{d\tau} d\tau \\
 &= z(0) + \lim_{t \rightarrow \infty} \int_0^t \dot{z}(\tau) d\tau \\
 &= z(0) + \lim_{t \rightarrow \infty} \int_0^t \lim_{s \rightarrow 0} \dot{z}(\tau) e^{-s\tau} d\tau \\
 &= z(0) + \lim_{s \rightarrow 0} \lim_{t \rightarrow \infty} \int_0^t \dot{z}(\tau) e^{-s\tau} d\tau \\
 &= z(0) + \lim_{s \rightarrow 0} \int_0^\infty \dot{z}(\tau) e^{-s\tau} d\tau \\
 &= z(0) + \lim_{s \rightarrow 0} (sZ(s) - z(0)) \\
 &= \lim_{s \rightarrow 0} sZ(s),
 \end{aligned}$$

which completes the proof.

9.1.2 System Type for Reference Tracking

Consider the closed loop system shown in Figure 9.1. The transfer function

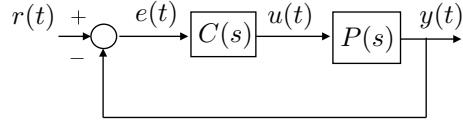


Figure 9.1: Block diagram for reference tracking.

from the reference r to the error $e = r - y$ is computed as

$$\begin{aligned}
 E(s) &= R(s) - P(s)C(s)E(s) \\
 \implies E(s) &= \frac{1}{1 + P(s)C(s)}R(s)
 \end{aligned}$$

Step Input.

Suppose that $r(t)$ is a step of size one, then $R(s) = \frac{1}{s}$ and

$$E(s) = \frac{1}{1 + P(s)C(s)} \frac{1}{s}.$$

Assuming that the closed loop system is stable and therefore that $\lim_{t \rightarrow \infty} e(t)$

CHAPTER 9. SYSTEM TYPE AND INTEGRATORS

is finite, then the final value theorem gives

$$\begin{aligned}\lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} sE(s) \\ &= \lim_{s \rightarrow 0} s \frac{1}{1 + P(s)C(s)} \frac{1}{s} \\ &= \lim_{s \rightarrow 0} \frac{1}{1 + P(s)C(s)} \\ &= \frac{1}{1 + \lim_{s \rightarrow 0} P(s)C(s)}.\end{aligned}$$

Let

$$M_p \triangleq \lim_{s \rightarrow 0} P(s)C(s),$$

then

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{1 + M_p}.$$

Now suppose that

$$P(s)C(s) = \frac{K(s + z_1)(s + z_2) \cdots (s + z_m)}{(s + p_1)(s + p_2) \cdots (s + p_n)},$$

then

$$M_p = \lim_{s \rightarrow 0} P(s)C(s) = \frac{Kz_1z_2 \cdots z_m}{p_1p_2 \cdots p_n}.$$

In this case, the steady state error is $\frac{1}{1+M_p}$ which is finite. Note also that the DC-gain of the closed loop system is

$$k_{DC} = \frac{1}{1 + M_p}.$$

Now suppose that $P(s)C(s)$ contains a pole at the origin, or in other words, one free integrator, i.e.,

$$P(s)C(s) = \frac{K(s + z_1)(s + z_2) \cdots (s + z_m)}{s(s + p_1)(s + p_2) \cdots (s + p_n)}.$$

In this case we have

$$M_p = \lim_{s \rightarrow 0} P(s)C(s) = \infty,$$

and the steady state error is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{1 + \infty} = 0.$$

Therefore, one free integrator in $P(s)C(s)$ implies that the steady state error to a step is zero. Note that multiple free integrators, i.e.,

$$P(s)C(s) = \frac{K(s + z_1)(s + z_2) \cdots (s + z_m)}{s^q(s + p_1)(s + p_2) \cdots (s + p_n)},$$

gives that same result for $q \geq 1$.

Ramp Input.

Now suppose that $r(t)$ is a unit ramp, or in other words that $R(s) = \frac{1}{s^2}$. In this case the error is given by

$$E(s) = \frac{1}{1 + P(s)C(s)} \frac{1}{s^2}.$$

Again assuming that the closed loop system is stable, the final value theorem gives

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} sE(s) \\ &= \lim_{s \rightarrow 0} s \frac{1}{1 + P(s)C(s)} \frac{1}{s^2} \\ &= \lim_{s \rightarrow 0} \frac{1}{s + sP(s)C(s)} \\ &= \frac{1}{\lim_{s \rightarrow 0} sP(s)C(s)}. \end{aligned}$$

Defining

$$M_v = \lim_{s \rightarrow 0} sP(s)C(s),$$

gives

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{M_v}.$$

If $P(s)C(s)$ has no free integrators, i.e.,

$$P(s)C(s) = \frac{K(s + z_1)(s + z_2) \cdots (s + z_m)}{(s + p_1)(s + p_2) \cdots (s + p_n)}$$

then

$$M_v = \lim_{s \rightarrow 0} s \frac{K(s + z_1)(s + z_2) \cdots (s + z_m)}{(s + p_1)(s + p_2) \cdots (s + p_n)} = 0.$$

Therefore

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{0} = \infty.$$

If on the other hand $P(s)C(s)$ has one free integrator, i.e.,

$$P(s)C(s) = \frac{K(s + z_1)(s + z_2) \cdots (s + z_m)}{s(s + p_1)(s + p_2) \cdots (s + p_n)}$$

then

$$M_v = \lim_{s \rightarrow 0} s \frac{K(s + z_1)(s + z_2) \cdots (s + z_m)}{s(s + p_1)(s + p_2) \cdots (s + p_n)} = \frac{Kz_1z_2 \cdots z_m}{p_1p_2 \cdots p_n},$$

therefore the steady state error is finite. If $P(s)C(s)$ has two or more free integrators, then

$$M_v = \lim_{s \rightarrow 0} s \frac{K(s + z_1)(s + z_2) \cdots (s + z_m)}{s^q(s + p_1)(s + p_2) \cdots (s + p_n)} = \infty,$$

when $q \geq 2$, therefore

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{\infty} = 0.$$

Therefore, the number of free integrators in $P(s)C(s)$ is the key parameter for reference tracking with zero steady state error.

For the reference tracking problem, we say that the system is *type q* if the steady state error to input $R(s) = \frac{1}{s^{q+1}}$ is finite and if the steady state error to input $R(s) = \frac{1}{s^p}$ is zero for $1 \leq p \leq q$. In general, for the feedback system shown in Figure 9.1, the steady state tracking error when $R(s) = \frac{1}{s^{q+1}}$ is given by

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} sE(s) = \lim_{s \rightarrow 0} \left(\frac{1}{1 + P(s)C(s)} \right) \left(\frac{1}{s^q} \right).$$

Table 9.1 summarizes the tracking error for different inputs as a function of system type, where

$$\begin{aligned} M_p &= \lim_{s \rightarrow 0} P(s)C(s) \\ M_v &= \lim_{s \rightarrow 0} sP(s)C(s) \\ M_a &= \lim_{s \rightarrow 0} s^2 P(s)C(s). \end{aligned}$$

Table 9.1: Reference tracking verse system type.

System Type	Step ($\frac{1}{s}$)	ramp ($\frac{1}{s^2}$)	parabola ($\frac{1}{s^3}$)	...
0	$\frac{1}{1+M_p}$	∞	∞	...
1	0	$\frac{1}{M_v}$	∞	...
2	0	0	$\frac{1}{M_a}$...
3	0	0	0	...
\vdots	\vdots	\vdots	\vdots	

As an example, consider the feedback system shown in Figure 9.2. The open

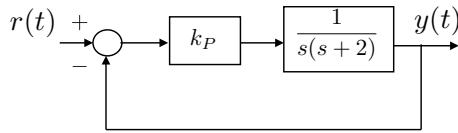


Figure 9.2: The system type for reference tracking for this system is type 1.

loop system

$$P(s)C(s) = \frac{k_P}{s(s+2)}$$

CHAPTER 9. SYSTEM TYPE AND INTEGRATORS

has one free integrator (pole at zero) and therefore the system is type 1. From Table 9.1 the tracking error when the input is a step is zero, and the tracking error when the input is a ramp of slope one is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{M_v} = \frac{1}{\lim_{s \rightarrow 0} sP(s)C(s)} = \frac{1}{\frac{k_P}{2}} = \frac{2}{k_P}.$$

The tracking error when the input is a parabola, or higher order polynomial, is ∞ , meaning that $y(t)$ and $r(t)$ diverge as $t \rightarrow \infty$.

As another example, consider the system shown in Figure 9.3.

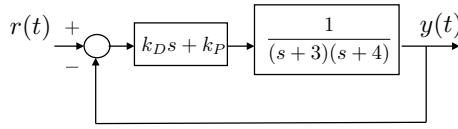


Figure 9.3: The system type for reference tracking for this system is type 0.

In this case, the open loop system

$$P(s)C(s) = \frac{k_Ds + k_P}{(s + 3)(s + 4)}$$

does not have any free integrators (poles at zero) and therefore the system is type 0. From Table 9.1 the tracking error when the input is a step is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{1 + M_p} = \frac{1}{1 + \lim_{s \rightarrow 0} P(s)C(s)} = \frac{1}{1 + \frac{k_P}{12}} = \frac{12}{12 + k_P}.$$

The tracking error when the input is a ramp , or higher order polynomial, is ∞ , meaning that $y(t)$ and $r(t)$ diverge as $t \rightarrow \infty$.

If on the other hand, a PID controller is used as shown in Figure 9.4 where

$$U(s) = \left(k_P + \frac{k_I}{s} + k_Ds \right) E(s) = \left(\frac{k_Ds^2 + k_Ps + k_I}{s} \right) E(s),$$

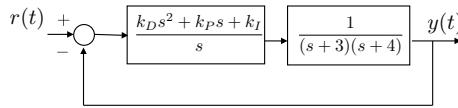


Figure 9.4: The system type for reference tracking for this system is type 1.

then the open loop system

$$P(s)C(s) = \frac{k_Ds^2 + k_Ps + k_I}{s(s + 3)(s + 4)}$$

has one free integrator (pole at zero) and therefore the system becomes type 1 by design. From Table 9.1 the tracking error when the input is a step is zero, and the tracking error when the input is a ramp is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{M_v} = \frac{1}{\lim_{s \rightarrow 0} sP(s)C(s)} = \frac{1}{\frac{k_I}{12}} = \frac{12}{k_I}.$$

The tracking error when the input is a parabola, or higher order polynomial, is ∞ , meaning that $y(t)$ and $r(t)$ diverge as $t \rightarrow \infty$.

9.1.3 System Type for Input Disturbances

Consider the general feedback loop shown in Figure 9.5. There are in general four inputs to the system, namely the reference input $r(t)$, the input disturbance $d_{in}(t)$, the output disturbance $d_{out}(t)$, and the sensor noise $n(t)$. In general $r(t)$, $d_{in}(t)$, and $d_{out}(t)$ are signals with low frequency content, and the sensor noise $n(t)$ has high frequency content. The negative sign on the disturbance and noise terms is for notational convenience.

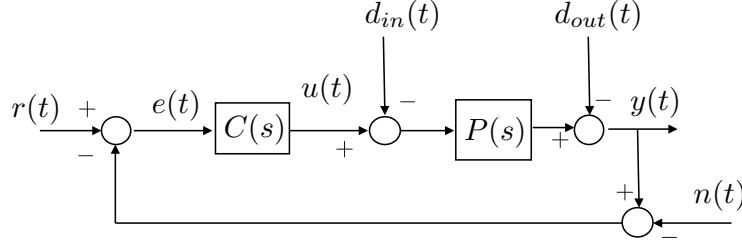


Figure 9.5: General feedback loop with reference input $r(t)$, input disturbance $d_{in}(t)$, output disturbance $d_{out}(t)$, and sensor noise $n(t)$.

We can compute the transfer function from each input to the error signal by following the signal flow to obtain

$$E(s) = R(s) + N(s) + D_{out}(s) + P(s)D_{in}(s) - P(s)C(s)E(s).$$

Solving for $E(s)$ gives

$$\begin{aligned} E(s) &= \frac{1}{1 + P(s)C(s)}R(s) + \frac{1}{1 + P(s)C(s)}N(s) \\ &\quad + \frac{1}{1 + P(s)C(s)}D_{out}(s) + \frac{P(s)}{1 + P(s)C(s)}D_{in}(s). \end{aligned}$$

Since the transfer function from $N(s)$ and $D_{out}(s)$ to $E(s)$ is identical to the transfer function from $R(s)$ to $E(s)$, the system type as defined in the previous section for reference inputs, also characterizes the steady state response to output disturbances and noise. However, the transfer function from the input

disturbance to the error is different, and so we need to further analyze system type with respect to input disturbances.

Suppose that the input disturbance is given by $D_{in}(s) = \frac{1}{s^{q+1}}$, then by the final value theorem, the steady state error due to the input disturbance is

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} s \frac{P(s)}{1 + P(s)C(s)} D_{in}(s) = \lim_{s \rightarrow 0} \left(\frac{P(s)}{1 + P(s)C(s)} \right) \left(\frac{1}{s^q} \right).$$

The system type with respect to input disturbance is the value of q such that the steady state error is finite, and the steady state error is zero for $D_{in}(s) = \frac{1}{s^p}$ for $1 \leq p \leq q$.

For example, consider the system shown in Figure 9.6. Recall from the discussion above, that the system type with respect to the reference input is type 1.

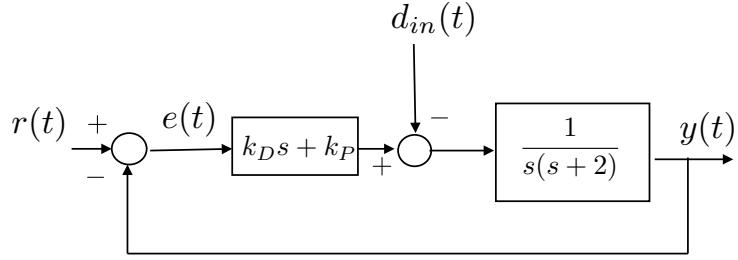


Figure 9.6: The system type for input disturbances for this system is type 0.

For the input disturbance we have

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \left(\frac{P(s)}{1 + P(s)C(s)} \right) \left(\frac{1}{s^q} \right) \\ &= \lim_{s \rightarrow 0} \left(\frac{\frac{1}{s(s+2)}}{1 + \left(\frac{1}{s(s+2)} \right) (k_D s + k_P)} \right) \left(\frac{1}{s^q} \right) \\ &= \lim_{s \rightarrow 0} \left(\frac{1}{s(s+2) + (k_D s + k_P)} \right) \left(\frac{1}{s^q} \right) \\ &= \lim_{s \rightarrow 0} \left(\frac{1}{k_P} \right) \left(\frac{1}{s^q} \right) \end{aligned}$$

which is finite when $q = 0$. Therefore, the system type with respect to input disturbances is type 0 and the steady state error to a step input disturbance is $\frac{1}{k_P}$ whereas the steady state error with respect to a ramp and higher order polynomials is infinite.

If on the other hand, we add an integrator as in Figure 9.7, then

$$\begin{aligned}
 \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \left(\frac{P(s)}{1 + P(s)C(s)} \right) \left(\frac{1}{s^q} \right) \\
 &= \lim_{s \rightarrow 0} \left(\frac{\frac{1}{s(s+2)}}{1 + \left(\frac{1}{s(s+2)} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right)} \right) \left(\frac{1}{s^q} \right) \\
 &= \lim_{s \rightarrow 0} \left(\frac{s}{s^2(s+2) + (k_D s^2 + k_P s + k_I)} \right) \left(\frac{1}{s^q} \right) \\
 &= \lim_{s \rightarrow 0} \left(\frac{1}{k_I} \right) \left(\frac{1}{s^{q-1}} \right)
 \end{aligned}$$

which is finite when $q = 1$. Therefore, the system type with respect to input disturbances is type 1 and the steady state error to a step on the input disturbance is zero and the steady state error with respect to a ramp on the input disturbance is $\frac{1}{k_I}$ whereas the steady state error with respect to a parabola and higher order polynomials on the input disturbance is infinite.

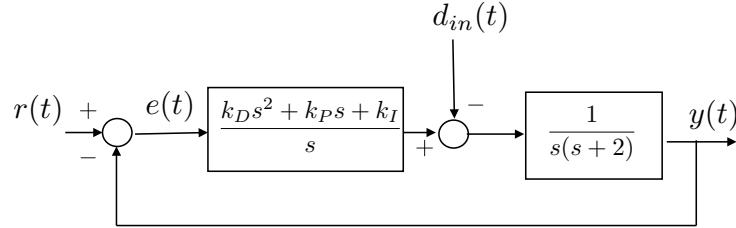


Figure 9.7: The system type for input disturbances for this system is type 1.

Therefore, the system type with respect to input disturbances is related to the number of free integrators in the controller $C(s)$ and not in the open loop system $P(s)C(s)$. As a consequence, while an integrator may not be necessary for steady state tracking, it is often necessary to reject constant input disturbances.

9.2 Design Study A. Single Link Robot Arm



Homework Problem A.9

- (a) When the controller for the single link robot arm is PD control, what is the system type? Characterize the steady state error when the reference input is a step, a ramp, and a parabola. How does this change if you add an integrator?

- (b) Consider the case where a constant disturbance acts at the input to the plant (for example gravity in this case). What is the steady state error to a constant input disturbance when the integrator is not present, and when it is present?

Solution

The closed loop system is shown in Figure 9.8.

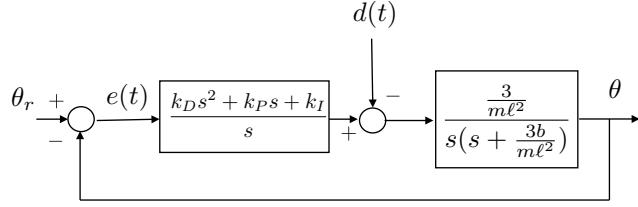


Figure 9.8: Closed loop system for problem HW A.9.

Without the integrator, the open-loop transfer function is given by

$$P(s)C(s) = \left(\frac{\frac{3}{m\ell^2}}{s(s + \frac{3b}{m\ell^2})} \right) (k_D s + k_P).$$

The system has one free integrator and is therefore type 1, which, from Table 9.1 implies that the tracking error when the input is a step is zero, and the tracking error when the input is a ramp is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{M_v} = \frac{1}{\lim_{s \rightarrow 0} s P(s) C(s)} = \frac{1}{\frac{k_P}{b}} = \frac{b}{k_P}.$$

The tracking error when the input is a parabola, or higher order polynomial, is ∞ , meaning that $\theta(t)$ and $\theta_r(t)$ diverge as $t \rightarrow \infty$.

With the integrator, the open loop transfer function is

$$P(s)C(s) = \left(\frac{\frac{3}{m\ell^2}}{s(s + \frac{3b}{m\ell^2})} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right).$$

which has two free integrators and is therefore type 2. Therefore, from Table 9.1 the tracking error when the input is either a step or a ramp is zero, and the tracking error when the input is a parabola is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{M_a} = \frac{1}{\lim_{s \rightarrow 0} s^2 P(s) C(s)} = \frac{1}{\frac{k_I}{b}} = \frac{b}{k_I}.$$

The tracking error when the input is t^3 or a higher order polynomial, is ∞ .

CHAPTER 9. SYSTEM TYPE AND INTEGRATORS

For the input disturbance, the transfer function from $D(s)$ to $E(s)$ is given by

$$E(s) = \frac{P(s)}{1 + P(s)C(s)}D(s).$$

Without the integrator, and when $D(s) = \frac{A}{s^{q+1}}$, the steady state error is given by

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{P}{1 + PC} \frac{A}{s^q} \\ &= \lim_{s \rightarrow 0} \left(\frac{\left(\frac{3}{m\ell^2} \right)}{1 + \left(\frac{3}{s(s + \frac{3b}{m\ell^2})} \right)} \right) \left(\frac{A}{s^q} \right) \\ &= \lim_{s \rightarrow 0} \left(\frac{\frac{3}{m\ell^2}}{s(s + \frac{3b}{m\ell^2}) + \frac{3}{m\ell^2}(k_D s + k_P)} \right) \left(\frac{A}{s^q} \right) \\ &= \frac{A \frac{3}{m\ell^2}}{\frac{3k_P}{m\ell^2}} \\ &= \frac{A}{k_P}, \end{aligned}$$

if $q = 0$. Therefore, to an input disturbance the system is type 0. The steady state error when a constant step of size A is placed on $d(t)$ is $\frac{A}{k_P}$.

With the integrator, and when $D(s) = \frac{A}{s^{q+1}}$, the steady state error is given by

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{P}{1 + PC} \frac{A}{s^q} \\ &= \lim_{s \rightarrow 0} \left(\frac{\left(\frac{3}{m\ell^2} \right)}{1 + \left(\frac{3}{s(s + \frac{3b}{m\ell^2})} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right)} \right) \left(\frac{A}{s^q} \right) \\ &= \lim_{s \rightarrow 0} \left(\frac{\frac{3}{m\ell^2}}{s^2(s + \frac{3b}{m\ell^2}) + \frac{3}{m\ell^2}(k_D s^2 + k_P s + k_I)} \right) \left(\frac{A}{s^q} \right) \\ &= \frac{A \frac{3}{m\ell^2}}{\frac{3k_I}{m\ell^2}} \\ &= \frac{A}{k_I}, \end{aligned}$$

if $q = 1$. Therefore, to an input disturbance the system is type 1. The steady state error when $d(t)$ is a constant step is zero, and the steady state error when $d(t)$ is a ramp of slope of size A is $\frac{A}{k_I}$.

9.3 Design Study B. Inverted Pendulum



Homework Problem B.9

- (a) When the inner loop controller for the inverted pendulum is PD control, what is the system type with respect to tracking of the inner loop? Characterize the steady state error when $\tilde{\theta}^r$ is a step, a ramp, and a parabola. What is the system type with respect to an input disturbance?
- (b) When the outer loop controller for the inverted pendulum is PD control, what is the system type of the outer loop with respect to tracking? Characterize the steady state error when z^r is a step, a ramp, and a parabola. How does this change if you add an integrator? What is the system type with respect to an input disturbance for both PD and PID control?

Solution

The block diagram for the inner loop is shown in Figure 9.9.

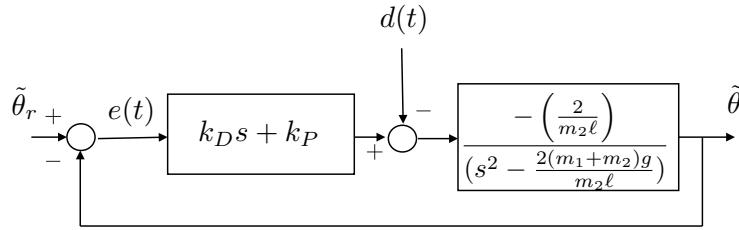


Figure 9.9: Inner loop system for problem HW B.9.

The open loop system is given by

$$P(s)C(s) = \left(\frac{-\frac{2}{m_2 \ell}}{s^2 - \frac{2(m_1+m_2)g}{m_2 \ell}} \right) (k_D s + k_P).$$

Since there are no free integrators, the system is type 0, and from Table 9.1 the tracking error when the input is a step is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{1 + M_p} = \frac{1}{1 + \lim_{s \rightarrow 0} s P(s) C(s)} = \frac{1}{1 + \frac{k_P}{(m_1+m_2)g}}.$$

The tracking error when the input is a ramp, or higher order polynomial, is ∞ .

CHAPTER 9. SYSTEM TYPE AND INTEGRATORS

For the disturbance input, the steady state error to a step on $d(t)$ is

$$\begin{aligned}
\lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} s \frac{P(s)}{1 + P(s)C(s)} \frac{1}{s} \\
&= \lim_{s \rightarrow 0} \frac{\left(\frac{-\frac{2}{m_2 \ell}}{s^2 - \frac{2(m_1+m_2)g}{m_2 \ell}} \right)}{1 + \left(\frac{-\frac{2}{m_2 \ell}}{s^2 - \frac{2(m_1+m_2)g}{m_2 \ell}} \right) (k_D s + k_P)} \\
&= \lim_{s \rightarrow 0} \frac{\left(-\frac{2}{m_2 \ell} \right)}{\left(s^2 - \frac{2(m_1+m_2)g}{m_2 \ell} \right) + \left(-\frac{2}{m_2 \ell} \right) (k_D s + k_P)} \\
&= \frac{\left(-\frac{2}{m_2 \ell} \right)}{\left(-\frac{2(m_1+m_2)g}{m_2 \ell} \right) + \left(-\frac{2k_P}{m_2 \ell} \right)} \\
&= \frac{1}{(m_1 + m_2)g + k_P}.
\end{aligned}$$

The system is type 0 with respect to the input disturbance.

The block diagram for the outer loop is shown in Figure 9.10.

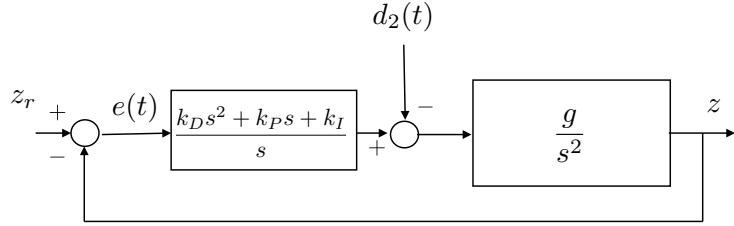


Figure 9.10: Outer loop system for problem HW B.9.

The open loop system is given by

$$P(s)C(s) = \left(\frac{g}{s^2} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right).$$

When $k_I = 0$ there are two free integrators in $P(s)C(s)$ and the system is type 2, and from Table 9.1 the tracking error when the input is a parabola is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{M_a} = \frac{1}{\lim_{s \rightarrow 0} s^2 P(s)C(s)} = \frac{1}{k_P g}.$$

When $k_I \neq 0$, there are three free integrators in $P(s)C(s)$ and the system is type 3, with zero tracking error for a step, ramp, and parabola.

For the disturbance input, the steady state error when $D(s) = \frac{1}{s^{q+1}}$ is

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} s \frac{P(s)}{1 + P(s)C(s)} \frac{1}{s^{q+1}} = \lim_{s \rightarrow 0} \frac{\left(\frac{g}{s^2} \right)}{1 + \left(\frac{g}{s^2} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right)} \frac{1}{s^q}.$$

CHAPTER 9. SYSTEM TYPE AND INTEGRATORS

Without the integrator, i.e., when $k_I = 0$ we have

$$\begin{aligned}\lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{g}{s^2 + g(k_D s + k_P)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{g}{g(k_P)} \frac{1}{s^q}\end{aligned}$$

which is finite when $q = 0$. Therefore, the system is type 0 with respect to the input disturbance and the steady state error when $d_2(t)$ is a unit step is $1/k_P$.

When $k_I \neq 0$, we have

$$\begin{aligned}\lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{gs}{s^3 + g(k_D s^2 + k_P s + k_I)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{g}{g(k_I)} \frac{1}{s^{q-1}}\end{aligned}$$

which is finite when $q = 1$. Therefore, the system is type 1 with respect to the input disturbance and the steady state error when $d_2(t)$ is a unit ramp is $1/k_I$.

9.4 Design Study C. Satellite Attitude Control



Homework Problem C.9

- (a) When the inner loop controller of the satellite system is PD control, what is the system type of the inner loop with respect to the reference input, and with respect to the disturbance input? Characterize the steady state error when the reference input is a step, a ramp, and a parabola. Characterize the steady state error when the input disturbance is a step, a ramp, and a parabola.
- (b) With PD control for the outer loop, what is the system type of the outer loop with respect to the reference input and with respect to the disturbance input? Characterize the steady state error when the reference input and disturbance input is a step, a ramp, and a parabola. How does this change if you add an integrator?

Solution

The block diagram for the inner loop is shown in Figure 9.11.

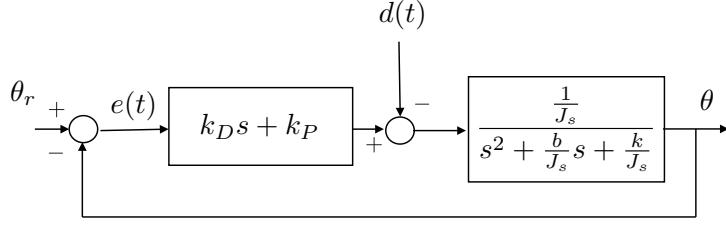


Figure 9.11: Inner loop system for problem HW C.9.

The open loop system is given by

$$P(s)C(s) = \left(\frac{\frac{1}{J_s}}{s^2 + \frac{b}{J_s} s + \frac{k}{J_s}} \right) (k_D s + k_P).$$

Since there are no free integrators, the system is type 0, and from Table 9.1 the tracking error when the input is a step is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{1 + M_p} = \frac{1}{1 + \lim_{s \rightarrow 0} s P(s) C(s)} = \frac{1}{1 + \frac{k_P}{k}}.$$

The tracking error when the input is a ramp, or higher order polynomial, is ∞ .

For a disturbance input of $D(s) = 1/s^{q+1}$, the steady state error is

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} s \frac{P(s)}{1 + P(s)C(s)} \frac{1}{s^{q+1}} \\ &= \lim_{s \rightarrow 0} \frac{\left(\frac{\frac{1}{J_s}}{s^2 + \frac{b}{J_s} s + \frac{k}{J_s}} \right)}{1 + \left(\frac{\frac{1}{J_s}}{s^2 + \frac{b}{J_s} s + \frac{k}{J_s}} \right) (k_D s + k_P)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{\frac{1}{J_s}}{\left(s^2 + \frac{b}{J_s} s + \frac{k}{J_s} \right) + \frac{1}{J_s} (k_D s + k_P)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{\frac{1}{J_s}}{\frac{k}{J_s} + \frac{k_P}{J_s}} \frac{1}{s^q} \end{aligned}$$

which is finite and equal to $1/(k + k_P)$ when $q = 0$. Therefore, the system is type 0 with respect to the input disturbance, and the steady state error to a step on $d(t)$ is $1/(k + k_P)$, and is infinite to a ramp and higher order polynomials.

The block diagram for the outer loop is shown in Figure 9.12.

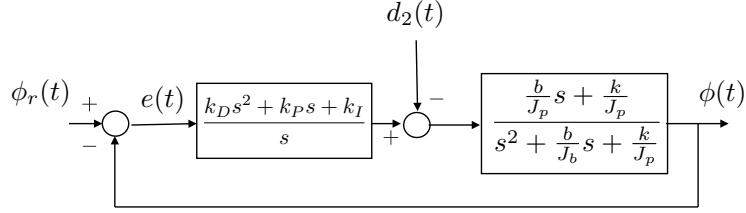


Figure 9.12: Outer loop system for problem HW C.9.

The open loop system is given by

$$P(s)C(s) = \left(\frac{\frac{b}{J_p} s + \frac{k}{J_p}}{s^2 + \frac{b}{J_p} s + \frac{k}{J_p}} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right).$$

When $k_I = 0$ there are no free integrator in $P(s)C(s)$ and the system is type 0, and from Table 9.1 the tracking error when the input is a step is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{1 + M_p} = \frac{1}{1 + \lim_{s \rightarrow 0} P(s)C(s)} = \frac{1}{1 + k_P}.$$

When the reference is a ramp or higher order polynomial, the tracking error is infinite. When $k_I \neq 0$, there is one free integrators in $P(s)C(s)$ and the system is type 1. From Table 9.1 the tracking error when the input is a ramp is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{M_v} = \frac{1}{\lim_{s \rightarrow 0} s P(s)C(s)} = \frac{1}{k_I}.$$

The tracking error when ϕ_r is a step is zero, and it is infinite when ϕ_r is a parabola or higher order polynomial.

For the disturbance input, the steady state error when $D(s) = \frac{1}{s^{q+1}}$ is

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} s \frac{P(s)}{1 + P(s)C(s)} \frac{1}{s^{q+1}} = \lim_{s \rightarrow 0} \frac{\left(\frac{\frac{b}{J_p} s + \frac{k}{J_p}}{s^2 + \frac{b}{J_p} s + \frac{k}{J_p}} \right)}{1 + \left(\frac{\frac{b}{J_p} s + \frac{k}{J_p}}{s^2 + \frac{b}{J_p} s + \frac{k}{J_p}} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right)} \frac{1}{s^q}.$$

Without the integrator, i.e., when $k_I = 0$ we have

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{\frac{b}{J_p} s + \frac{k}{J_p}}{(s^2 + \frac{b}{J_p} s + \frac{k}{J_p}) + (\frac{b}{J_p} s + \frac{k}{J_p})(k_D s + k_P)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{1}{1 + k_P} \frac{1}{s^q} \end{aligned}$$

which is finite when $q = 0$. Therefore, the system is type 0 with respect to the input disturbance and the steady state error when $d_2(t)$ is a unit step is $1/(1 + k_P)$.

CHAPTER 9. SYSTEM TYPE AND INTEGRATORS

When $k_I \neq 0$, we have

$$\begin{aligned}\lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{s \left(\frac{b}{J_p} s + \frac{k}{J_p} \right)}{s \left(s^2 + \frac{b}{J_p} s + \frac{k}{J_p} \right) + \left(\frac{b}{J_p} s + \frac{k}{J_p} \right) (k_D s^2 + k_P s + k_I)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{1}{k_I} \frac{1}{s^{q-1}}\end{aligned}$$

which is finite when $q = 1$. Therefore, the system is type 1 with respect to the input disturbance and the steady state error when $d_2(t)$ is a unit ramp is $1/k_I$. It is zero when d_2 is a step, and it is infinite when d_2 is a parabola or higher order polynomial.

Notes and References

The PID structure analyzed in this chapter uses a differentiator on the error as show in Figure 7.1 instead of differentiating the output variable as show in Figure 7.2. It is straight forward to show using the final value theorem that the system type does not change under the two configurations. However, the value of the steady state error does change and is not necessarily given by the simple formulas in Table 9.1.

Chapter 10

Digital Implementation of PID Controllers

10.1 Theory

In this book we work primarily with continuous time systems and continuous time controllers. However, almost all modern control strategies are implemented digitally on a micro-controller or a micro-processor, typically implemented using C-code. In this chapter, we describe how PID controllers can be implemented in a sampled data system using computer code.

A general PID control signal is given by

$$u(t) = k_P e(t) + k_I \int_{-\infty}^t e(\tau) d\tau + k_D \frac{de}{dt}(t),$$

where $e(t) = y_r(t) - y(t)$ is the error between the desired reference output $y_r(t)$ and the current output $y(t)$. Define $u_P(t) = e(t)$, $u_I(t) = \int_{-\infty}^t e(\tau) d\tau$, and $u_D(t) = \frac{de}{dt}(t)$. The PID controller is therefore given by $u(t) = k_p u_P(t) + k_I u_I(t) + k_D u_D(t)$. In the following paragraphs we will derive discrete sampled-time equivalents for $u_P(t)$, $u_I(t)$, and $u_D(t)$.

First consider a sampled time implementation of $u_P(t)$. When $t = nT_s$, where T_s is the sample period and n is the sample time, we have $u_P(nT_s) = e(nT_s)$, or using discrete time notation

$$u_P[n] = e[n].$$

To derive the sample-time implementation of an integrator, note that

$$\begin{aligned} u_I(nT_s) &= \int_{-\infty}^{nT_s} e(\tau)d\tau \\ &= \int_{-\infty}^{(n-1)T_s} e(\tau)d\tau + \int_{(n-1)T_s}^{T_s} e(\tau)d\tau \\ &= u_I((n-1)T_s) + \int_{(n-1)T_s}^{T_s} e(\tau)d\tau. \end{aligned}$$

As shown in Figure 10.1 a trapezoidal approximation of the integral between $(n-1)T_s$ and nT_s is given by

$$\int_{(n-1)T_s}^{T_s} e(\tau)d\tau \approx \frac{T_s}{2} (e(nT_s) + e((n-1)T_s)).$$

Therefore

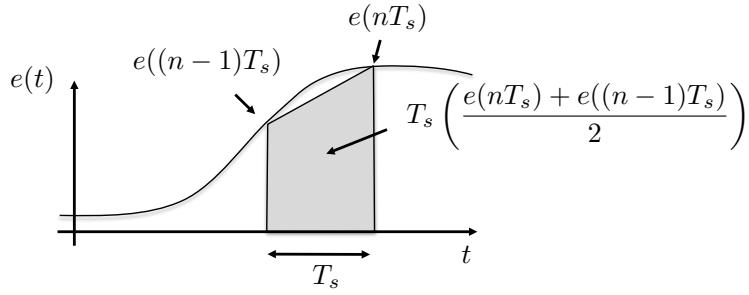


Figure 10.1: Trapezoidal rule for digital integration.

$$u_I(nT_s) \approx u_I((n-1)T_s) + \frac{T_s}{2} (e(nT_s) + e((n-1)T_s)).$$

Taking the z-transform we get

$$U_I(z) = z^{-1}U_I(z) + \frac{T_s}{2}(E(z) + z^{-1}E(z)).$$

Solving for $U_I(z)$ gives

$$U_I(z) = \frac{T_s}{2} \left(\frac{1+z^{-1}}{1-z^{-1}} \right) E(z). \quad (10.1)$$

Since in the Laplace domain we have

$$U_I(s) = \frac{1}{s}E(s),$$

we have derived the so-called Tustin approximation where s in the Laplace domain is replaced with the z -transform approximation

$$s \mapsto \frac{2}{T_s} \left(\frac{1 - z^{-1}}{1 + z^{-1}} \right), \quad (10.2)$$

where T_s is the sample period [18]. Taking the inverse z -transform of Equation (10.1) gives the discrete time implementation of an integrator as

$$u_I[n] = u_I[n - 1] + \frac{T_s}{2} (e[n] + e[n - 1]). \quad (10.3)$$

In the Laplace domain, a pure differentiator is given by $u_D(s) = sE(s)$. However, since a pure differentiator is not causal, the standard approach is to use the band-limited, or dirty derivative

$$U_D(s) = \frac{s}{\sigma s + 1} E(s),$$

where σ is small, and $\frac{1}{\sigma}$ defines the bandwidth of the differentiator. In practical terms, the dirty derivative differentiates signals with frequency content less than $\frac{1}{\sigma}$ radians per second. Using the the Tustin approximation (10.2), the dirty derivative in the z -domain becomes

$$\begin{aligned} u_D(z) &= \frac{\frac{2}{T_s} \left(\frac{1 - z^{-1}}{1 + z^{-1}} \right)}{\frac{2\sigma}{T_s} \left(\frac{1 - z^{-1}}{1 + z^{-1}} \right) + 1} E(z) \\ &= \frac{\left(\frac{2}{2\sigma + T_s} \right) (1 - z^{-1})}{1 - \left(\frac{2\sigma - T_s}{2\sigma + T_s} \right) z^{-1}} E(z). \end{aligned}$$

Transforming to the time domain, we have

$$u_D[n] = \left(\frac{2\sigma - T_s}{2\sigma + T_s} \right) u_D[n - 1] + \left(\frac{2}{2\sigma + T_s} \right) (e[n] - e[n - 1]). \quad (10.4)$$

10.1.1 Integrator Anti-Windup

A potential problem with a straight-forward implementation of PID controllers is integrator wind up. When the error $y_r - y$ is large and a large error persists for an extended period of time, the value of the integrator, can become large, or “wind up.” A large integrator will cause u , to saturate, which will cause the system to push with maximum effort in the direction needed to correct the error. Since the value of the integrator will continue to wind up until the error signal changes sign, the control signal may not come out of saturation until well after the error has changed sign, which can cause a large overshoot and may potentially destabilize the system.

Since integrator wind up can destabilize the system, it is important to add an anti-wind-up scheme. A number of different anti-wind-up schemes are possible. In this section we will discuss two schemes that are commonly used in practice. The first schemes is based on the fact that integrators are typically used to correct for steady state error and that overshoot is caused by integrating error during the transients of the step response. Therefore, the integrator can be turned off when the derivative of the error is large. This anti-windup scheme is implemented by an if-statement conditioned on the size of \dot{e} .

The second anti-windup scheme is to subtract from the integrator exactly the amount needed to keep u at the saturation bound. In particular, let

$$u_{\text{unsat}} = k_P e + k_D u_D + k_I u_I$$

denote the unsaturated control value before applying the anti-windup scheme. After the saturation block we have that

$$u = \text{sat}(u_{\text{unsat}}) = \text{sat}(k_P e + k_D u_D + k_I u_I).$$

The idea of anti-windup is to modify the output of the integrator from u_I to u_I^+ so that

$$k_P e + k_D u_D + k_I u_I^+ = \text{sat}(k_P e + k_D u_D + k_I u_I) = u.$$

Subtracting u_{unsat} from u gives

$$\begin{aligned} u - u_{\text{unsat}} &= (k_P e + k_D u_D + k_I u_I^+) - (k_P e + k_D u_D + k_I u_I) \\ &= k_I(u_I^+ - u_I). \end{aligned}$$

Solving for u_I^+ gives

$$u_I^+ = u_I + \frac{1}{k_I}(u - u_{\text{unsat}}),$$

which is the updated value of the integrator that ensures that the control signal remains just out of saturation. Note that if the control value is not in saturation then $u = u_{\text{unsat}}$ and the integrator value remains unchanged.

10.1.2 Matlab Implementation

Matlab code that implements a general PID loop is shown below.

```

1 function u = pidloop(y_r,y,flag,kp,ki,kd,limit,Ts,sigma,vbar)
2     persistent integrator;
3     persistent differentiator;
4     persistent error_d1;
5     % reset (initialize) persistent variables when flag==1
6     if flag==1,
7         integrator = 0;
8         differentiator = 0;
9         error_d1 = 0; % _d1 means delayed by one time step
10    end
11    % compute the current error

```

CHAPTER 10. DIGITAL IMPLEMENTATION OF PID CONTROLLERS

```

12     error = y_r - y;
13     % update differentiator
14     a1 = (2*sigma-Ts)/(2*sigma+Ts);
15     a2 = 2/(2*sigma+Ts);
16     differentiator = a1*differentiator+a2*(error-error_d1);
17     % update integrator
18     if abs(differentiator) < vbar,
19         integrator = integrator + (Ts/2)*(error + error_d1);
20     end
21     % unsaturated PID control
22     u_unsat = kp*error + ki*integrator + kd*differentiator;
23     % saturated PID control
24     u = sat(u_unsat, limit);
25     % anti-windup: update integrator to keep u out of saturation
26     if ki<0 % only implement when integrator is turned on
27         integrator = integrator + 1/ki * (u - u_unsat);
28     end
29     % update the delayed error for next time through loop
30     error_d1 = error;
31 end % end pidloop
32
33 % function to saturate the output
34 function out = sat(in, limit)
35     if in > limit,          out = limit;
36     elseif in < -limit;    out = -limit;
37     else                   out = in;
38 end
39 end % end sat

```

The inputs on Line 1 are the reference output y_r ; the current output y ; a flag used to reset the persistent variables; the PID gains k_P , k_I , and k_D ; the limit of the saturation command; the sample time T_s ; the time constant σ of the differentiator, and the anti-windup limit $limit2$ on the derivative. Lines 2–10 define the persistent variables that persist between function calls, and initializes these variables when $flag==1$. The error is computed on Line 12. Lines 13–16 implement the digital differentiator given in Equation (10.4). Lines 17–20 implement the digital integrator given in Equation (10.3) together with the first anti-windup scheme described above. In particular, the integrator is only turned on when the output of the differentiator is below $limit2$. The unsaturated PID controller is computed on Line 22, and the corresponding saturated control is computed on Line 24. The anti-windup scheme that subtracts area from the integrator so that the resulting controller is just at the limit of saturation is given in Lines 26–28. Line 30 updates the persistent variable `error_d1`.

Note that the Matlab code given above assumes that differentiator uses the error signal $u_D(t) = \frac{de}{dt}$. If on the other hand, the architecture shown in Figure 7.2 is to be used, where $u_D(t) = -\frac{dy}{dt}$, then Line 16 can be replaced with

```
1     differentiator = a1*differentiator + a2*(y - y_d1);
```

where an additional persistent variable `y_d1` representing the delayed output

y must also be added. In some cases the derivative signal \dot{y} is available from a sensor and can be used instead of approximating the differentiator.

10.1.3 Gain Selection for PID control

Gain selection for PID control usually proceeds in the following manner. The proportional and derivative gains k_P and k_D are first selected according to the methods discussed in Chapter 8. The integral gain k_I is then tuned by starting with zero and then slowly increasing the gain until the steady state error is removed. The parameters associated with the anti-windup scheme, especially \bar{v} , are also adjusted during this phase. The proportional and derivative gains can then be adjusted if necessary by retuning ω_n and ζ . In the case of successive loop closure, integrators are only added on the outer loop.

The addition of an integrator has a destabilizing effect on the feedback system. This effect is most easily understood by using the root locus method described in Appendix e, where it can be seen that for second order systems, a large k_I will always result in closed loop poles in the right half plane. The root locus can also be used to aid in the selection of the integral gain k_I , where the value of k_I is selected so that integrator has minimal impact on the closed loop poles designed using PD control. A more effective method for selecting integral gains will be discussed in Chapter 12.

10.2 Design Study A. Single Link Robot Arm



Homework Problem A.10

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Modify the system dynamics file so that the parameters vary by up to 20% each time they are run (uncertainty parameter = 0.2).
- (b) Rearrange the block diagram so that the controller is implemented as an m-function implemented at the sample rate of $T_s = 0.01$. The controller should only assume knowledge of the angle θ and the reference angle θ_r .
- (c) Implement the PID controller designed in Problems A.8 using an m-function called `arm_ctrl.m`. Use the dirty derivative gain of $\tau = 0.05$. Tune the integrator to remove the steady state error caused by the uncertain parameters.

Solution

See <http://controlbook.byu.edu> for the complete solution.

The Matlab code for the controller is shown below.

CHAPTER 10. DIGITAL IMPLEMENTATION OF PID CONTROLLERS

```

1 function tau=arm_ctrl(in,P)
2     theta_c = in(1);
3     theta   = in(2);
4     t       = in(3);
5
6     % set persistent flag to initialize integrator and
7     % differentiator at the start of the simulation
8     persistent flag
9     if t<P.Ts,
10         flag = 1;
11     else
12         flag = 0;
13     end
14
15     % compute equilibrium torque tau_e
16     tau_e = P.m*P.g*(P.ell/2)*cos(theta);
17     % compute the linearized torque using PID
18     tau_tilde = PID_th(theta_c,theta,flag,P.kp,P.ki,P.kd, ...
19                         P.tau_max,P.Ts,P.sigma);
20     % compute total torque
21     tau = tau_e + tau_tilde;
22
23 end
24
25 %_____
26 % PID control for angle theta
27 function u = PID_th(theta_c,theta,flag,kp,ki,kd,limit,Ts,sigma)
28     % declare persistent variables
29     persistent integrator
30     persistent thetadot
31     persistent error_d1
32     persistent theta_d1
33     % reset persistent variables at start of simulation
34     if flag==1,
35         integrator = 0;
36         thetadot = 0;
37         error_d1 = 0;
38         theta_d1 = 0;
39     end
40
41     % compute the error
42     error = theta_c-theta;
43     % update derivative of y
44     thetadot = P.beta*thetadot + (1-P.beta)*((theta-theta_d1)/P.Ts);
45     % update integral of error
46     if abs(thetadot)<0.05,
47         integrator = integrator + (Ts/2)*(error+error_d1);
48     end
49     % update delayed variables for next time through the loop
50     error_d1 = error;
51     theta_d1 = theta;
52
53     % compute the pid control signal
54     u_unsat = kp*error + ki*integrator - kd*thetadot;
55     u = sat(u_unsat,limit);
56
57     % integrator anti-windup

```

```

58     if ki≠0,
59         integrator = integrator + Ts/ki*(u-u_unsat);
60     end
61 end
62
63 function out = sat(in,limit)
64     if in > limit,      out = limit;
65     elseif in < -limit,   out = -limit;
66     else                 out = in;
67     end
68 end

```

10.3 Design Study B. Inverted Pendulum



Homework Problem B.10

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Modify the system dynamics file so that the parameters m_1 , m_2 , ℓ and b vary by up to 20% of their nominal value each time they are run (uncertainty parameter = 0.2).
- (b) Rearrange the block diagram so that the controller is implemented as an m-function implemented at the sample rate of $T_s = 0.01$. The controller should only assume knowledge of the position z and the angle θ , as well as the reference position z_r .
- (e) Implement the nested PID loops designed in Problems B.8 using an m-function called `pendulum_ctrl.m`. Use the dirty derivative gain of $\tau = 0.05$. Tune the integrator to remove the steady state error caused by the uncertain parameters.

Solution

See <http://controlbook.byu.edu> for the complete solution.

The Matlab code for the controller is shown below.

```

1 function F=pendulum_ctrl(in,P)
2     z_r    = in(1);
3     z     = in(2);
4     theta = in(3);
5     t     = in(4);
6
7     % set persistent flag to initialize integrators and
8     % differentiators at the start of the simulation
9     persistent flag
10    if t<P.Ts

```

CHAPTER 10. DIGITAL IMPLEMENTATION OF PID CONTROLLERS

```

11         flag = 1;
12     else
13         flag = 0;
14     end
15
16     % compute the desired angled angle using the outer loop control
17     theta_r = PID_z(z_r,z,flag,P.kp_z,P.ki_z,P.kd_z, ...
18                       30*pi/180,P.Ts,P.sigma);
19     % compute the force using the inner loop
20     F       = PD_th(theta_r,theta,flag,P.kp_th,P.kd_th, ...
21                       P.F_max,P.Ts,P.sigma);
22
23 end
24
25 %
26 % PID control for position
27 function u = PID_z(z_c,z,flag,kp,ki,kd,limit,Ts,sigma)
28     % declare persistent variables
29     persistent integrator
30     persistent zdot
31     persistent error_d1
32     persistent z_d1
33     % reset persistent variables at start of simulation
34     if flag==1
35         integrator = 0;
36         zdot      = 0;
37         error_d1   = 0;
38         z_d1      = 0;
39     end
40
41     % compute the error
42     error = z_c-z;
43     % update derivative of z
44     zdot = (2*sigma-Ts)/(2*sigma+Ts)*zdot...
45             + 2/(2*sigma+Ts)*(z-z_d1);
46     % update integral of error
47     if abs(zdot)<.1
48         integrator = integrator + (Ts/2)*(error+error_d1);
49     end
50     % update delayed variables for next time through the loop
51     error_d1 = error;
52     z_d1     = z;
53
54     % compute the pid control signal
55     u_unsat = kp*error + ki*integrator - kd*zdot;
56     u = sat(u_unsat,limit);
57
58     % integrator anti-windup
59     if ki<0
60         integrator = integrator + Ts/ki*(u-u_unsat);
61     end
62 end
63
64 %
65 %
66 % PID control for angle theta
67 function u = PD_th(theta_c,theta,flag,kp,kd,limit,Ts,sigma)

```

```

68 % declare persistent variables
69 persistent thetadot
70 persistent theta_d1
71 % reset persistent variables at start of simulation
72 if flag==1
73     thetadot = 0;
74     theta_d1 = 0;
75 end
76
77 % compute the error
78 error = theta_c-theta;
79 % update derivative of y
80 thetadot = (2*sigma-Ts)/(2*sigma+Ts)*thetadot...
81         + 2/(2*sigma+Ts)*(theta-theta_d1);
82 % update delayed variables for next time through the loop
83 error_d1 = error;
84 theta_d1 = theta;
85
86 % compute the pid control signal
87 u_unsat = kp*error - kd*thetadot;
88 u = sat(u_unsat,limit);
89
90 end
91
92 %
93 % saturation function
94 function out = sat(in,limit)
95     if in > limit,      out = limit;
96     elseif in < -limit, out = -limit;
97     else                out = in;
98 end
99 end

```

10.4 Design Study C. Satellite Attitude Control



Homework Problem C.10

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Modify the system dynamics file so that the parameters J_s , J_p , k and b vary by up to 20% of their nominal value each time they are run (uncertainty parameter = 0.2).
- (b) Rearrange the block diagram so that the controller is implemented as an m-function implemented at the sample rate of $T_s = 0.01$. Assume that the controller only has knowledge of the angles ϕ and θ as well as the reference angle ϕ_r .
- (c) Implement the nested PID loops designed in Problems C.8 using an m-function called `satellite_ctrl.m`. Use the dirty derivative gain of $\tau = 0.05$.

Tune the integrator to remove the steady state error caused by the uncertain parameters.

Solution

See <http://controlbook.byu.edu> for the complete solution.

The Matlab code for the controller is shown below.

```

1  function tau=satellite_ctrl(in,P)
2      phi_r    = in(1);
3      phi     = in(2);
4      theta   = in(3);
5      t       = in(4);
6
7      % set persistent flag to initialize integrators and
8      % differentiators at the start of the simulation
9      persistent flag
10     if t<P.Ts
11         flag = 1;
12     else
13         flag = 0;
14     end
15
16     % compute the desired angled angle using the outer loop control
17     phi_r = phi_r/P.k_DC_phi;
18     theta_r = PID_phi(phi_r,phi,flag,P.kp_phi,P.ki_phi,P.kd_phi, ...
19                         P.Ts,P.sigma);
20     % compute the force using the inner loop
21     tau     = PD_th(theta_r,theta,flag,P.kp_th,P.kd_th, ...
22                         P.taumax,P.Ts,P.sigma);
23
24 end
25
26 %_____
27 % PID control for position
28 function u = PID_phi(phi_r,phi,flag,kp,ki,kd,Ts,sigma)
29     % declare persistent variables
30     persistent integrator
31     persistent error_dl
32     persistent phidot
33     persistent phi_dl
34     % reset persistent variables at start of simulation
35     if flag==1
36         integrator = 0;
37         error_dl   = 0;
38         phidot     = 0;
39         phi_dl     = 0;
40     end
41
42     % compute the error
43     error = phi_r-phi;
44
45     % update derivative of phi
46     phidot = (2*sigma-Ts)/(2*sigma+Ts)*phidot...
47                 + 2/(2*sigma+Ts)*(phi-phi_dl);

```

```

48      % update delayed variables for next time through the loop
49      phi_d1 = phi;
50
51      % update integral of error
52      integrator = integrator + (Ts/2)*(error+error_d1);
53      % update delayed variables for next time through the loop
54      error_d1 = error;
55
56      % compute the pid control signal
57      u = kp*error + ki*integrator -kd*phidot;
58
59  end
60
61
62 %_____
63 % PID control for angle theta
64 function u = PD_th(theta_r,theta,flag,kp,kd,limit,Ts,sigma)
65     % declare persistent variables
66     persistent thetadot
67     persistent theta_d1
68     % reset persistent variables at start of simulation
69     if flag==1
70         thetadot    = 0;
71         theta_d1   = 0;
72     end
73
74     % compute the error
75     error = theta_r-theta;
76     % update derivative of y
77     thetadot = (2*sigma-Ts)/(2*sigma+Ts)*thetadot...
78             + 2/(2*sigma+Ts)*(theta-theta_d1);
79     % update delayed variables for next time through the loop
80     theta_d1 = theta;
81
82     % compute the pid control signal
83     u_unsat = kp*error - kd*thetadot;
84     u = sat(u_unsat,limit);
85
86 end
87
88 %_____
89 % saturation function
90 function out = sat(in,limit)
91     if     in > limit,          out = limit;
92     elseif in < -limit,        out = -limit;
93     else                      out = in;
94     end
95 end

```

Notes and References

A standard reference for digital implementation of PID controllers is [18]. Simple anti-wind-up schemes are discussed in [18, 4].

Part IV

Observer Based Control Design

In Part III we focused on designing feedback control systems for second order systems. Since second order system have two poles, we saw that it is possible to use feedback control with two gains to place the closed loop poles at any desired location. In Chapter 11 of this book we will see that for an n^{th} order system it is possible to arbitrarily place the closed loop poles, but that this will require n feedback gains. In that sense, the full state feedback controllers that we will derive in Chapter 11 are a generalization of PD control to higher order systems. This generalization is most easily accomplished using the state space model of the system derived in Chapter 6. In fact, the state space model leads to efficient numerical routines for designing and implementing the feedback controller.

In Chapter 12 we show how integral control can be added to the state feedback controllers designed in Chapter 11. The method introduced in Chapter 12 has significant advantage over the integrator tuning method discussed in Chapter 10 in the sense that tuning the integral gain will not impact the other closed loop poles, and therefore allows for greater design flexibility.

We saw in Chapter 10 that in order to implement PD/PID control, the derivative of the output needs to be reconstructed using a digital differentiator. The generalization of this idea to state feedback control is that the full state of the system must be reconstructed. The tool that we will use to reconstruct the system state is called an *observer*. The observer design problem will be discussed in Chapter 13.

It turns out that observers can be used in a variety of different contexts outside of feedback control. In particular, observers can be used to solve target tracking problems and to estimate the system parameters of complex dynamic systems. In the setting of feedback control where unknown input and output disturbances act on the system, observers can be used to estimate the unknown disturbances. In Chapter 14 we will show how to design the observer to estimate both the system state and the unknown constant input disturbance. We will also show that estimating the input disturbance is mathematically and functionally similar to adding an integrator.

The design tools developed in Part IV of this book use state space models and require the use of linear algebra and matrix analysis. A brief review of linear algebra concepts that will be used in Part IV are given in Appendix f.

Chapter 11

Full State Feedback

11.1 Theory

In this chapter we will show how to use full state feedback to stabilize a linear time-invariant system represented in state space form. Recall from Section 6.1.2, that for state space equations given by

$$\dot{x} = Ax + Bu \quad (11.1)$$

$$y = Cx, \quad (11.2)$$

the poles of the system are the eigenvalues of A , or in other words, the roots of the open loop characteristic equation

$$\Delta_{ol}(s) = \det(sI - A) = 0.$$

The matrix A is called the state-interaction matrix because it specifies how the states interact with each other. The basic idea behind full state feedback is to use the feedback signal to change the interaction matrix so that the poles are in specified locations. A full state feedback controller is given by

$$u = -Kx + \nu, \quad (11.3)$$

where $K \in \mathbb{R}^{m \times n}$ is the feedback gains, and ν will be a signal that is specified later. Using (11.3) in Equation (11.1) results in the closed-loop state space equations

$$\dot{x} = Ax + B(-Kx + \nu) = (A - BK)x + B\nu \quad (11.4)$$

$$y = Cx, \quad (11.5)$$

where the interaction matrix is now $A - BK$, and so the closed loop poles will be given by the eigenvalues of $A - BK$, or in other words, the roots of the closed loop characteristic equation

$$\Delta_{cl}(s) = \det(sI - (A - BK)) = 0.$$

A natural question that will be addressed in the remainder of this section, is how to select K so that the closed loop characteristic equation is equal to a desired closed-loop characteristic $\Delta_{cl}^d(s)$, where the designer selects the roots of $\Delta_{cl}^d(s) = 0$ to achieve desired closed loop performance.

Section 6.1.2 showed that state space model given by Equations (11.1)–(11.2) is equivalent to the transfer function model given by

$$Y(s) = C(sI - A)^{-1}BU(s). \quad (11.6)$$

While the mapping from state-space model to transfer function model is unique, the reverse is not true. In fact, there are an infinite number of state space models that correspond to a particular transfer function model. Accordingly, we say that a state space model is a *realization* of the transfer function model, if they produce the same input-output behavior when the initial conditions are zero.

In particular, define the change of variables

$$z = P^{-1}x, \quad (11.7)$$

where P is an invertible matrix. Then

$$\dot{z} = P^{-1}\dot{x} = P^{-1}(Ax + Bu) = P^{-1}APz + P^{-1}Bu \quad (11.8)$$

$$y = Cx = CPz \quad (11.9)$$

is an alternative state space representation of system (11.1)–(11.2). To show that these two representations have the same input-output behavior, or in other words, the same transfer function, note from Equation (11.6) that the transfer function of system (11.8)–(11.9) is

$$\begin{aligned} Y(s) &= (CP(sI - P^{-1}AP)^{-1}P^{-1}B)U(s) \\ &= (CP(sP^{-1}P - P^{-1}AP)^{-1}P^{-1}B)U(s) \\ &= (CP(P^{-1}(sI - A)P)^{-1}P^{-1}B)U(s) \\ &= (CPP^{-1}(sI - A)^{-1}PP^{-1}B)U(s) \\ &= (C(sI - A)^{-1}B)U(s), \end{aligned}$$

which is identical to Equation (11.6). Therefore Equations (11.1)–(11.2) and Equations (11.8)–(11.9) are realizations of the same system.

11.1.1 State Space Realization in Control Canonic Form

In this section, we will derive one specific state space realization for single-input single-output transfer functions, where the state feedback problem is particularly simple. The realization that we derive below is called the *control canonic form* of the system.

Consider the general SISO monic transfer function model given by

$$Y(s) = \frac{b_ms^m + b_{m-1}s^{m-1} + \cdots + b_1s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1s + a_0}U(s), \quad (11.10)$$

where $m < n$. The block diagram of the system is shown in Figure 11.1.

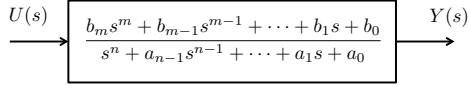


Figure 11.1: General transfer function for a SISO system.

The first step in deriving the control canonic state space equations is to artificially decompose the transfer function as shown in Figure 11.2, where the numerator polynomial and the denominator polynomial appear in separate cascaded blocks, and where $Z(s)$ is an artificial intermediate variable.

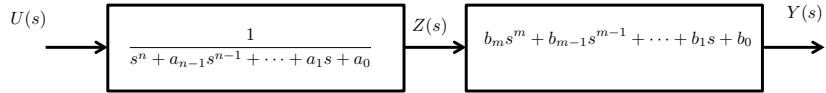


Figure 11.2: General transfer function for a SISO system decomposed into state dynamics and output dynamics.

In the s -domain, the equations corresponding to Figure 11.2 can be written as

$$Z(s) = \frac{1}{s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} U(s)$$

$$Y(s) = (b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0) Z(s).$$

Taking the inverse Laplace transform of these equations gives

$$\frac{d^n z}{dt^n} = u(t) - a_{n-1} \frac{d^{n-1} z}{dt^{n-1}} - \dots - a_1 \dot{z} - a_0 z(t) \quad (11.11)$$

$$y(t) = b_m \frac{d^m z}{dt^m} + b_{m-1} \frac{d^{m-1} z}{dt^{m-1}} + \dots + b_1 \dot{z} + b_0 z(t). \quad (11.12)$$

The next step in the derivation of the state space equations is to draw the analog computer implementation of Equations (11.11) and (11.12), which is shown in Figure 11.3.

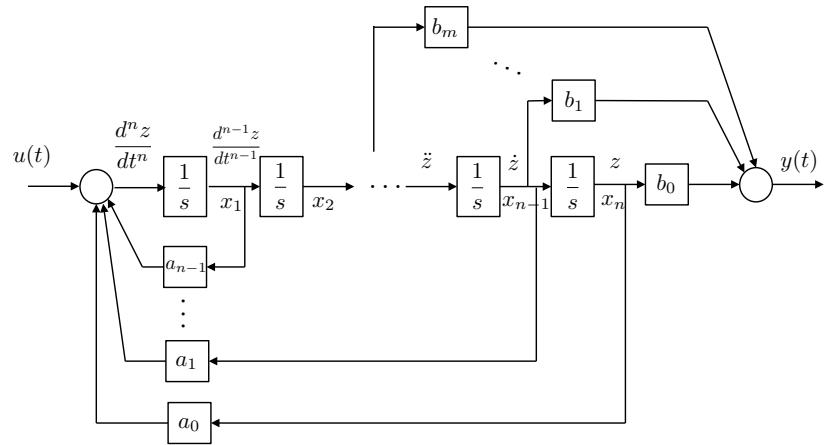


Figure 11.3: Block diagram showing the analog computer implementation of Equations (11.11) and (11.12).

Labeling the output of the integrators as the state variables x_1, \dots, x_n , as shown in Figure 11.3, we can write the state space equations as

$$\begin{aligned} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{pmatrix} &= \begin{pmatrix} -a_{n-1} & -a_{n-2} & \cdots & -a_1 & -a_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} u \quad (11.13) \\ y &= (0 \ \dots \ 0 \ b_m \ \dots \ b_1 \ b_0) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + 0 \cdot u. \end{aligned}$$

We will define the control canonic realization as

$$\begin{aligned} \dot{x}_c &= A_c x_c + B_c u \\ y &= C_c x_c + D u, \end{aligned}$$

where

$$A_c \triangleq \begin{pmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_1 & -a_0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix} \quad (11.14)$$

$$B_c \triangleq \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad (11.15)$$

$$C_c \triangleq (0 \ \dots \ 0 \ b_m \ \dots \ b_1 \ b_0). \quad (11.16)$$

Note that the negative of the coefficients of the denominator of (11.10) constitute the first row of A_c matrix, and that the remainder of A_c has ones on the lower off diagonal and zeros in all other elements. Matrices with this structure are said to be in *companion form*. The B_c vector has a single one as its first elements, with the remaining elements equal to zero. The coefficients of the numerator of Equation (11.10) are the last p -elements of C_c .

As an example, suppose that the transfer function model of the system is given by

$$Y(s) = \frac{9s + 20}{s^3 + 6s^2 - 11s + 8} U(s),$$

then the control canonic realization of the system is given by

$$\begin{aligned} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} &= \begin{pmatrix} -6 & 11 & -8 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u \\ y &= (0 \ 9 \ 20) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \end{aligned} \quad (11.17)$$

11.1.2 Full State Feedback using Control Canonic Form

The design of a state feedback controller is particularly easy when the state space equations are in control canonic form. The key insight is that there is a direct relationship between matrices in companion form, as shown in Equation (11.14) and their characteristic polynomial

$$\Delta_{ol}(s) = s^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0.$$

CHAPTER 11. FULL STATE FEEDBACK

Suppose that we let

$$u = -K_c x_c + \nu = -\begin{pmatrix} k_1 & k_2 & \dots & k_{n-1} & k_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} + \nu. \quad (11.18)$$

Substituting into Equation (11.13) and rearranging gives

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{pmatrix} = \begin{pmatrix} -(a_{n-1} + k_1) & -(a_{n-2} + k_2) & \dots & -(a_1 + k_{n-1}) & -(a_0 + k_n) \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \vdots & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \nu.$$

Since the state interaction matrix is still in companion form, the characteristic polynomial of the closed loop system is given by

$$\Delta_{cl}(s) = s^n + (a_{n-1} + k_1)s^{n-1} + \dots + (a_1 + k_{n-1})s + (a_0 + k_n).$$

If the desired characteristic polynomial is given by

$$\Delta_{cl}^d(s) = s^n + \alpha_{n-1}s^{n-1} + \dots + \alpha_1s + \alpha_0, \quad (11.19)$$

then we have that

$$\begin{aligned} a_{n-1} + k_1 &= \alpha_{n-1} \\ a_{n-2} + k_2 &= \alpha_{n-2} \\ &\vdots \\ a_1 + k_{n-1} &= \alpha_1 \\ a_0 + k_n &= \alpha_0. \end{aligned}$$

Defining the row vectors

$$\begin{aligned} \mathbf{a}_A &= (a_{n-1}, a_{n-2}, \dots, a_1, a_0) \\ \boldsymbol{\alpha} &= (\alpha_{n-1}, \alpha_{n-2}, \dots, \alpha_1, \alpha_0) \\ K_c &= (k_1, k_2, \dots, k_{n-1}, k_n), \end{aligned}$$

we have that the feedback gains satisfy

$$K_c = \boldsymbol{\alpha} - \mathbf{a}_A, \quad (11.20)$$

where the subscript “c” is used to emphasize that the gains are when the state equations are in control canonic form.

As an example, consider the state equations given in Equation (11.17). The open loop characteristic polynomial is given by

$$\Delta_{ol}(s) = s^3 + 6s^2 - 11s + 8,$$

with roots at -7.5885 , and $0.7942 \pm j0.6507$. Suppose that the desired closed loop characteristic polynomial is

$$\Delta_{cl}^d(s) = s^3 + 5s^2 + 12s + 8,$$

with roots at -1 and $-2 \pm j2$, then $\mathbf{a}_A = (6, -11, 8)$, and $\boldsymbol{\alpha} = (5, 12, 8)$. The feedback gains that place the poles at the desired locations are therefore given by

$$K_c = \boldsymbol{\alpha} - \mathbf{a}_A = (5, 12, 8) - (6, -11, 8) = (-1, 23, 0).$$

11.1.3 Full State Feedback: General Case

The formula given in Equation (11.20) assumes that the state space equations are in control canonic form. It turns out that the closed loop eigenvalues cannot always be arbitrarily assigned to desired locations. When they can, the system $\dot{x} = Ax + Bu$ is said to be *controllable*. In this section we will derive an expression for the state feedback gains, and we will simultaneously derive a simple test for when the system is controllable.

Recall from Equations (11.7)–(11.9) that state space equations can be transformed using the change of variables in Equation (11.7) without modifying the input-output properties of the system. This implies that the change of variables does not change the characteristic polynomial. In fact, we can show this fact explicitly as follows:

$$\begin{aligned} \det(sI - P^{-1}AP) &= \det(sP^{-1}P - P^{-1}AP) \\ &= \det(P^{-1}(sI - A)P) \\ &= \det(P^{-1})\det(sI - A)\det(P) \\ &= \det(sI - A). \end{aligned}$$

Therefore, to derive full state feedback gains for the general state space equations

$$\dot{x} = Ax + Bu, \quad (11.21)$$

the idea is to find a state transformation matrix P so that the transformed system is in control canonic form, i.e., find P so that

$$x_c = P^{-1}x, \quad (11.22)$$

and

$$\dot{x}_c = P^{-1}APx_c + P^{-1}Bu = A_cx_c + B_cu, \quad (11.23)$$

where A_c and B_c have the form given by Equations (11.14) and (11.15), respectively. Therefore, we need to find P so that $P^{-1}B = B_c$, and $P^{-1}AP =$

CHAPTER 11. FULL STATE FEEDBACK

A_c , which are satisfied if and only if $B = PB_c$ and $AP = PA_c$. Let $P = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n)$, where $\mathbf{p}_i \in \mathbf{R}^n$ is the i^{th} column of P . The requirement that $B = PB_c$ implies that

$$B = (\mathbf{p}_1 \quad \mathbf{p}_2 \quad \dots \quad \mathbf{p}_n) \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \mathbf{p}_1.$$

Therefore, $\mathbf{p}_1 = B$. The requirement that $PA_c = AP$ implies that

$$(\mathbf{p}_1 \quad \mathbf{p}_2 \quad \dots \quad \mathbf{p}_n) \begin{pmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_1 & -a_0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \vdots & 1 & 0 \end{pmatrix} = A(\mathbf{p}_1 \quad \mathbf{p}_2 \quad \dots \quad \mathbf{p}_n).$$

After multiplication, the first $n - 1$ columns can be equated to get

$$-a_{n-1}\mathbf{p}_1 + \mathbf{p}_2 = A\mathbf{p}_1 \quad (11.24)$$

$$-a_{n-2}\mathbf{p}_1 + \mathbf{p}_3 = A\mathbf{p}_2 \quad (11.25)$$

$$\vdots \quad (11.26)$$

$$-a_1\mathbf{p}_1 + \mathbf{p}_n = A\mathbf{p}_{n-1}. \quad (11.27)$$

Equation (11.24), and the previous conclusion that $\mathbf{p}_1 = B$, implies that

$$\mathbf{p}_2 = a_{n-1}B + AB.$$

Using this results in Equation (11.25) gives

$$\mathbf{p}_3 = a_{n-2}B + a_{n-1}AB + A^2B.$$

We can continue to iteratively solve for each column of P until the $(n - 1)^{th}$ column, which from Equation (11.27) is

$$\mathbf{p}_n = a_1B + a_2AB + \dots + a_{n-2}A^{n-2}B + A^{n-1}B.$$

Notice that each column in P is a linear combination of the elements $B, AB, A^2B, \dots, A^{n-1}B$. Accordingly, defining

$$\mathcal{C}_{A,B} = (B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B), \quad (11.28)$$

gives

$$\mathbf{p}_1 = \mathcal{C}_{A,B} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{p}_2 = \mathcal{C}_{A,B} \begin{pmatrix} a_{n-1} \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \quad \dots \quad \mathbf{p}_n = \mathcal{C}_{A,B} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ 1 \end{pmatrix}.$$

Defining the matrix

$$\mathcal{A}_A = \begin{pmatrix} 1 & a_{n-1} & a_{n-2} & \dots & a_2 & a_1 \\ 0 & 1 & a_{n-1} & \vdots & a_3 & a_2 \\ 0 & 0 & 1 & \vdots & a_4 & a_3 \\ \vdots & & & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & 1 & a_{n-1} \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix}, \quad (11.29)$$

gives that the transformation matrix that converts the system into control canonic form is

$$P = \mathcal{C}_{A,B}\mathcal{A}_A.$$

Notice that since \mathcal{A}_A is an upper triangular matrix with ones on the diagonal, that $\det(\mathcal{A}_A) = 1$, implying that \mathcal{A}_A^{-1} always exists. It is clear that the matrix P is invertible with

$$P^{-1} = \mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1}$$

if and only if $\mathcal{C}_{A,B}^{-1}$ exists. The matrix $\mathcal{C}_{A,B}$ in Equation (11.28) is called the *controllability matrix*, and a single input system is said to be *controllable* if \mathcal{C} is invertible. Therefore, when the original system (11.21) is controllable, there exists an invertible state transformation matrix P so that the change of variables in Equation (11.22) is well defined and that results in the transformed equations being in control canonic form via Equations (11.23).

In control canonic form, the state feedback equation is

$$u = -K_c x_c + \nu = -(\boldsymbol{\alpha} - \mathbf{a}_A)x_c + \nu.$$

Using Equation (11.22) the state feedback control can be expressed in the original states x as

$$\begin{aligned} u &= -(\boldsymbol{\alpha} - \mathbf{a}_A)P^{-1}x + \nu \\ &= -(\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1}x + \nu \\ &= -Kx + \nu, \end{aligned} \quad (11.30)$$

where

$$K \triangleq (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1}. \quad (11.31)$$

Equation (11.31) is called *Ackermann's formula* for the feedback gains. Since $\mathcal{C}_{A,B}$ is only square in the single-input case, Ackermann's formula only holds for single input systems.

Using the full state feedback control in Equation (11.30) in the state equations (11.1)-(11.2) results in the closed loop system

$$\begin{aligned} \dot{x} &= (A - BK)x + B\nu \\ y &= Cx. \end{aligned}$$

CHAPTER 11. FULL STATE FEEDBACK

The resulting transfer function from ν to y is given by

$$Y(s) = (C(sI - (A - BK))^{-1}B) \nu(s). \quad (11.32)$$

A block diagram of the resulting closed loop system is shown in Figure 11.4 where the double line associated with x is meant to imply that the state x is not really available. The block diagram also shows the new input ν . Since the objective is for the output y to track the reference input r , we let $\nu(t) = k_r r(t)$, as shown in Figure 11.4. The feedforward gain k_r is selected so that the DC-gain from r to y is equal to one.

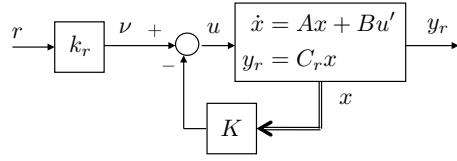


Figure 11.4: Full state feedback with reference input r .

From Equation (11.32), the transfer function from R to Y is given by

$$Y(s) = (C(sI - (A - BK))^{-1}Bk_r) R(s), \quad (11.33)$$

and the DC-gain is

$$k_{DC} = \lim_{s \rightarrow 0} [C(sI - (A - BK))^{-1}Bk_r] = -C(A - BK)^{-1}Bk_r. \quad (11.34)$$

Therefore, the DC-gain is equal to one if

$$k_r = -\frac{1}{C(A - BK)^{-1}B}.$$

The matrix $A - BK$ will be invertible when there are no eigenvalues at zero. Since K is selected to place the eigenvalues in the open left half plane, the matrix will always be invertible, and k_r exists. Note that in many applications, the output measures by the sensors is not the same as the desired reference output. In other words, the state space equations may be given by

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y_r &= C_r x \\ y_m &= C_m x, \end{aligned}$$

where y_r is the reference output and y_m is the measured output. In this case, the reference gain k_r is selected to make the DC-gain from r to y_r equal to one, or in other words

$$k_r = -\frac{1}{C_r(A - BK)^{-1}B}. \quad (11.35)$$

Chapter 13 will describe observers that use the measured output y_r to reconstruct the state x .

The feedforward gain computed in Equation (11.35) is for single input systems. More generally, if the size of the reference output y_r is equal to the size of the input vector $u \in \mathbb{R}^m$, then $C_r \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times m}$, which implies that the transfer matrix in Equation (11.33) is square and has dimension $m \times m$. Similarly, the gain $K_r \in \mathbb{R}^{m \times m}$ will also be square, and from Equation (11.34) will be equal to

$$k_r = -[C_r(A - BK)^{-1}B]^{-1}.$$

11.1.4 State Feedback for Linearized Systems

Suppose that the state space model is from a linearized system, i.e, suppose that $x \in \mathbb{R}^n$ is the state and that $x_e \in \mathbb{R}^n$ is the equilibrium state, and that $\tilde{x} = x - x_e$ is the state linearized around the equilibrium. Similarly, suppose that $u \in \mathbb{R}^m$ is the input, that $u_e \in \mathbb{R}^m$ is the input at equilibrium, and that $\tilde{u} = u - u_e$ is the input linearized around equilibrium, and that $y_r \in \mathbf{R}^p$ is the reference output, $y_{re} = C_r x_e \in \mathbf{R}^p$ is the reference output at equilibrium, and $\tilde{y}_r = y_r - y_{re}$ is the reference output linearized about equilibrium, then the linearized state space model is given by

$$\begin{aligned}\dot{\tilde{x}} &= A\tilde{x} + B\tilde{u} \\ \tilde{y}_r &= C_r\tilde{x}.\end{aligned}$$

Now suppose that the state feedback design process is used to find the state feedback controller

$$\tilde{u} = -K\tilde{x} + k_r\tilde{r} \quad (11.36)$$

where $\tilde{r} = r - y_{re}$ is the reference input around the equilibrium output, and where $\text{eig}(A - BK)$ are in the open left half plane, and the DC-gain from \tilde{r} to \tilde{y}_r is one. The controller input to the plant is therefore

$$u = u_e - K(x - x_e) + k_r(r - y_{re}). \quad (11.37)$$

11.2 Summary of Design Process - State Feedback

The state feedback design process can be summarized as follows.

Given the plant

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y_r &= C_r x\end{aligned}$$

and the desired pole locations p_1, p_2, \dots, p_n , find the gains K and k_r so that the feedback controller

$$u = -Kx + k_r r$$

places the poles at the desired location and results in DC-gain equal to one.

Step 1. Check to see if the system is controllable by computing the controllability matrix

$$\mathcal{C}_{A,B} = [B, AB, \dots, A^{n-1}B]$$

and checking to see if $\text{rank}(\mathcal{C}) = n$.

Step 2. Find the open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A) = s^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0,$$

and construct the row vector

$$\mathbf{a}_A = (a_{n-1}, a_{n-2}, \dots, a_1, a_0)$$

and the matrix

$$\mathcal{A}_A = \begin{pmatrix} 1 & a_{n-1} & a_{n-2} & \cdots & a_2 & a_1 \\ 0 & 1 & a_{n-1} & \cdots & a_3 & a_2 \\ \vdots & & \ddots & & & \vdots \\ 0 & 0 & \cdots & 0 & 1 & a_{n-1} \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$$

Step 3. Select the desired closed loop polynomial

$$\Delta_{cl}^d(s) = (s - p_1)(s - p_2) \cdots (s - p_n) = s^2 + \alpha_{n-1}s^{n-1} + \dots + \alpha_1s + \alpha_0,$$

and construct the row vector

$$\boldsymbol{\alpha} = (\alpha_{n-1}, \alpha_{n-2}, \dots, \alpha_1, \alpha_0).$$

Step 4. Compute the desired gains as

$$K = (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1}$$

$$k_r = \frac{-1}{C_r(A - BK)^{-1}B}.$$

If Matlab is available, these steps can be implemented using the following script.

```

1 % compute the controllability matrix
2 CC = ctrb(A,B);
3 % check for controllability
4 if rank(CC) == size(A,1),
5     disp('The system is controllable')
6 else
7     disp('The system is not controllable')
8 end
9 % place the poles at p1,p2,...,pn
10 K = place(A,B,p1,p2,p3,p4);
11 % compute the reference gain
12 kr = -1/(Cr*inv(A-B*K)*B);

```

11.2.1 Simple example

Given the system

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}x + \begin{pmatrix} 0 \\ 4 \end{pmatrix}u \\ y_r &= \begin{pmatrix} 5 & 0 \end{pmatrix}x\end{aligned}$$

find the feedback gain K to place the closed loop poles at $-1 \pm j$, and the reference gain k_r so that the DC-gain is one.

Step 1. The controllability matrix is given by

$$\mathcal{C}_{A,B} = [B, AB] = \begin{pmatrix} 0 & 4 \\ 4 & 12 \end{pmatrix}.$$

The determinant is $\det(\mathcal{C}) = -16 \neq 0$, therefore the system is controllable.

Step 2. The open loop characteristic polynomial is

$$\Delta_{ol}(s) = \det(sI - A) = \det \begin{pmatrix} s & -1 \\ -2 & s-3 \end{pmatrix} = s^2 - 3s - 2,$$

which implies that

$$\begin{aligned}\mathbf{a}_A &= (-3, -2) \\ \mathcal{A}_A &= \begin{pmatrix} 1 & -3 \\ 0 & 1 \end{pmatrix}\end{aligned}$$

Step 3. The desired closed loop polynomial is

$$\Delta_{cl}^d(s) = (s + 1 - j)(s + 1 + j) = s^2 + 2s + 2,$$

which implies that

$$\boldsymbol{\alpha} = (2, 2).$$

Step 4. The gains are therefore given as

$$\begin{aligned}K &= (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1} \\ &= ((2, 2) - (-3, -2)) \begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -\frac{3}{4} & \frac{1}{4} \\ \frac{1}{4} & 0 \end{pmatrix} \\ &= (5 \quad 4) \begin{pmatrix} 0 & \frac{1}{4} \\ \frac{1}{4} & 0 \end{pmatrix} \\ &= (1 \quad \frac{5}{4})\end{aligned}$$

$$\begin{aligned}
 k_r &= \frac{-1}{C_r(A - BK)^{-1}B} \\
 &= \frac{-1}{(5 \ 0) \left(\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} - \begin{pmatrix} 0 \\ 4 \end{pmatrix} \left(1 \ \frac{5}{4} \right) \right)^{-1} \begin{pmatrix} 0 \\ 4 \end{pmatrix}} \\
 &= \frac{-1}{(5 \ 0) \left(\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 4 & 5 \end{pmatrix} \right)^{-1} \begin{pmatrix} 0 \\ 4 \end{pmatrix}} \\
 &= \frac{-1}{(5 \ 0) \begin{pmatrix} 0 & 1 \\ -2 & -2 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 4 \end{pmatrix}} \\
 &= \frac{-1}{-10} = \frac{1}{10}.
 \end{aligned}$$

11.3 Design Study A. Single Link Robot Arm



Homework Problem A.11

The objective of this problem is to implement state feedback controller using the full state. Start with the simulation files developed in Homework A.10.

- (a) Select the closed loop poles as the roots of the equations $s^2 + 2\zeta\omega_n + \omega_n^2 = 0$ where ω_n , and ζ were found in Homework A.8.
- (b) Add the state space matrices A , B , C , D derived in Homework A.6 to your param file.
- (c) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}_{A,B}) = n$.
- (d) Find the feedback gain K so that the eigenvalues of $(A - BK)$ are equal to desired closed loop poles. Find the reference gain k_r so that the DC-gain from θ_r to θ is equal to one. Note that $K = (k_p, k_d)$ where k_p and k_d are the proportional and derivative gains found in Homework A.8. Why?
- (e) Modify the control code to implement the state feedback controller. To construct the state $x = (\theta, \dot{\theta})^\top$ use a digital differentiator to estimate $\dot{\theta}$.

Solution

From HW A.6, the state space equations for the single link robot arm are given by

$$\begin{aligned}
 \dot{x} &= \begin{pmatrix} 0 & 1.0000 \\ 0 & -0.667 \end{pmatrix} x + \begin{pmatrix} 0 \\ 66.667 \end{pmatrix} u \\
 y &= (1 \ 0) x.
 \end{aligned}$$

CHAPTER 11. FULL STATE FEEDBACK

Step 1. The controllability matrix is therefore

$$\mathcal{C}_{A,B} = [B, AB] = \begin{pmatrix} 0 & 66.6667 \\ 66.6667 & -44.44440 \end{pmatrix}.$$

The determinant is $\det(\mathcal{C}_{A,B}) = -4444 \neq 0$, therefore the system is controllable.

Step 2. The open loop characteristic polynomial is

$$\Delta_{ol}(s) = \det(sI - A) = \det \begin{pmatrix} s & -1 \\ 0 & s + 0.667 \end{pmatrix} = s^2 + 0.667s,$$

which implies that

$$\begin{aligned} \mathbf{a}_A &= (-0.667, 0) \\ \mathcal{A}_A &= \begin{pmatrix} 1 & -0.667 \\ 0 & 1 \end{pmatrix} \end{aligned}$$

Step 3. The desired closed loop polynomial is

$$\Delta_{cl}^d(s) = s^2 + 2\zeta\omega_n s + \omega_n^2 = s^2 + 6.3621s + 20.2445$$

which implies that

$$\boldsymbol{\alpha} = (6.3621, 20.2445).$$

Step 4. The gains are therefore given as

$$\begin{aligned} K &= (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1} \\ &= (0.3037 \quad 0.0854) \end{aligned}$$

$$\begin{aligned} k_r &= \frac{-1}{C(A - BK)^{-1}B} \\ &= 0.3037. \end{aligned}$$

Alternatively, we could have used the following Matlab script.

```

1 % sample rate
2 P.Ts = 0.01;
3
4 % dirty derivative gain
5 P.sigma = 0.05;
6 P.beta = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts); % dirty derivative gain
7
8 % equilibrium torque
9 P.theta_e = 0*pi/180;
10 P.tau_e = P.m*P.g*P.ell/2*cos(P.theta_e);
11

```

CHAPTER 11. FULL STATE FEEDBACK

```

12 %
13 % state space design
14 A = [...
15     0, 1;...
16     0, -3*P.b/P.m/(P.ell^2);...
17 ];
18 B = [0; 3/P.m/(P.ell^2) ];
19 C = [...
20     1, 0;...
21 ];
22
23 % tuning parameters
24 %tr = 0.8; % part (a)
25 % tuned to get fastest possible rise time before saturation.
26 tr = 0.4;
27 zeta = 0.707;
28
29 % desired closed loop polynomial
30 wn = 2.2/tr;
31 % gains for pole locations
32 des_char_poly = [1,2*zeta*wn,wn^2];
33 des_poles = roots(des_char_poly);
34
35 % is the system controllable?
36 if rank ctrb(A,B) ≠ 2,
37     disp('System Not Controllable');
38 else
39     P.K = place(A,B,des_poles);
40     P.kr = -1/(C*inv(A-B*P.K)*B);
41 end

```

The Matlab code for the controller is given by

```

1 function tau=arm_ctrl(in,P)
2     theta_c = in(1);
3     theta    = in(2);
4     t        = in(3);
5
6     % use a digital differentiator to find thetadot
7     persistent thetadot
8     persistent theta_d1
9     % reset persistent variables at start of simulation
10    if t<P.Ts
11        thetadot    = 0;
12        theta_d1    = 0;
13    end
14    thetadot = P.beta*thetadot + (1-P.beta)*((theta-theta_d1)/P.Ts);
15    theta_d1 = theta;
16
17    % construct the state
18    x = [theta; thetadot];
19    % compute equilibrium torque tau_e
20    theta_e = 0.0;
21    tau_e = P.m*P.g*(P.ell/2)*cos(theta_e);
22    % compute the state feedback controller
23    tau_tilde = - P.K*x + P.kr*theta_c;

```

```

24 % compute total torque
25 tau = sat( tau_e + tau_tilde, P.tau_max);
26 end
27
28 function out = sat(in,limit)
29 if in > limit, out = limit;
30 elseif in < -limit, out = -limit;
31 else out = in;
32 end
33 end

```

11.4 Design Study B. Inverted Pendulum



Homework Problem B.11

The objective of this problem is to implement state feedback controller using the full state. Start with the simulation files developed in Homework B.10.

- (a) Using the values for ω_{n_z} , ζ_z , ω_{n_θ} , and ζ_θ selected in Homework B.8, find the desired closed loop poles.
- (b) Add the state space matrices A , B , C , D derived in Homework B.6 to your param file.
- (c) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}) = n$.
- (d) Find the feedback gain K so that the eigenvalues of $(A - BK)$ are equal to desired closed loop poles. Find the reference gain k_r so that the DC-gain from z_r to z is equal to one.
- (e) Implement the state feedback scheme and tune the closed loop poles to get good response. You should be able to get much faster response using state space methods.

Solution

From HW B.6, the state space equations for the inverted pendulum are given by

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0 & -2.4500 & -0.0500 & 0 \\ 0 & 24.5000 & 0.1000 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 1 \\ -2 \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} x,\end{aligned}\tag{11.38}$$

where Equation (11.38) represents the measured outputs. The reference output is the position z , which implies that

$$y_r = (1 \ 0 \ 0 \ 0) x.$$

Step 1. The controllability matrix is therefore

$$\mathcal{C}_{A,B} = [B, AB, A^2B, A^3B] = \begin{pmatrix} 0 & 1.0000 & -0.0500 & 4.9025 \\ 0 & -2.0000 & 0.1000 & -49.0050 \\ 1.0000 & -0.0500 & 4.9025 & -0.4901 \\ -2.0000 & 0.1000 & -49.0050 & 2.9403 \end{pmatrix}.$$

The determinant is $\det(\mathcal{C}_{A,B}) = -1536 \neq 0$, implying that the system is controllable.

Step 2. The open loop characteristic polynomial is

$$\Delta_{ol}(s) = \det(sI - A) = s^4 + 0.0500s^3 - 24.5000s^2 - 0.9800s$$

which implies that

$$\mathbf{a}_A = (0.05, -24.5, 0.98, 0)$$

$$\mathcal{A}_A = \begin{pmatrix} 1 & 0.05 & -24.5, 0.98 \\ 0 & 1 & 0.05 & -24.5 \\ 0 & 0 & 1 & 0.05 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Step 3. The desired closed loop polynomial is

$$\Delta_{cl}^d(s) = (s^2 + 2\zeta_\theta\omega_{n_\theta}s + \omega_{n_\theta}^2)(s^2 + 2\zeta_z\omega_{n_z}s + \omega_{n_z}^2)$$

$$= s^4 + 4.4280s^3 + 9.5248s^2 + 9.2280s + 4.0000$$

which implies that

$$\boldsymbol{\alpha} = (4.4280, 9.5248, 9.2280, 4.0000).$$

Step 4. The gains are therefore given as

$$K = (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1}$$

$$= (-0.2041 - 17.1144 - 0.5208 - 2.4494)$$

The feedforward reference gain $k_r = -1/C_r(A - BK)^{-1}B$ is computed using $C_r = (1, 0, 0, 0)$, which gives

$$k_r = \frac{-1}{C_r(A - BK)^{-1}B}$$

$$= -0.2041.$$

CHAPTER 11. FULL STATE FEEDBACK

Alternatively, we could have used the following Matlab script.

```

1 % inverted Pendulum parameter file
2
3 % dirty derivative parameters
4 P.sigma = 0.05; % cutoff freq for dirty derivative
5 P.beta = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts); % dirty derivative gain
6
7 % saturation limits
8 P.F_max = 5; % Max Force, N
9 theta_max = 30.0*pi/180.0; % Max theta, rads
10
11 %%%%%%%%%%%%%%
12 % state feedback control
13 %%%%%%%%%%%%%%
14 % tuning parameters
15 tr_z = 1.5; % rise time for position
16 tr_theta = 0.5; % rise time for angle
17 zeta_z = 0.707; % damping ratio position
18 zeta_th = 0.707; % damping ratio angle
19
20 % state space equations
21 A = [...
22     0, 0, 1, 0;...
23     0, 0, 0, 1;...
24     0, -P.m1*P.g/P.m2, -P.b/P.m2, 0;...
25     0, (P.m1+P.m2)*P.g/P.m2/P.ell, P.b/P.m2/P.ell, 0;...
26 ];
27 B = [0; 0; 1/P.m2; -1/P.m2/P.ell];
28 C = [...
29     1, 0, 0, 0;...
30     0, 1, 0, 0;...
31 ];
32
33 % gain calculation
34 wn_th = 2.2/tr_theta; % natural frequency for angle
35 wn_z = 2.2/tr_z; % natural frequency for position
36 des_char_poly = conv([1, 2*zeta_z*wn_z, wn_z^2], [1, 2*zeta_th*wn_th, wn_th^2]);
37 des_poles = roots(des_char_poly);
38
39 % Compute the gains if the system is controllable
40 if rank(ctrb(A, B)) < 4
41     disp('The system is not controllable')
42 else
43     P.K = place(A, B, des_poles);
44     P.kr = -1.0/(C(1,:)*inv(A-B*K)*B);
45 end
46
47 sprintf('K: (%f, %f, %f, %f)\nkr: %f\n', P.K(1), P.K(2), P.K(3), P.K(4), P.kr)

```

The Matlab code for the controller is given by

```

1 function F=pendulum_ctrl(in,P)
2     z_r    = in(1);
3     z      = in(2);

```

```

4 theta = in(3);
5 t      = in(4);
6
7 % use a digital differentiator to find zdot and thetadot
8 persistent zdot
9 persistent z_d1
10 persistent thetadot
11 persistent theta_d1
12 % reset persistent variables at start of simulation
13 if t<P.Ts,
14     zdot      = 0;
15     z_d1      = 0;
16     thetadot  = 0;
17     theta_d1  = 0;
18 end
19 zdot = P.beta*zdot + (1-P.beta)*(z-z_d1)/P.Ts;
20 thetadot = P.beta*thetadot + (1-P.beta)*(theta-theta_d1)/P.Ts;
21 z_d1 = z;
22 theta_d1 = theta;
23
24 % construct the state
25 x = [z; theta; zdot; thetadot];
26 % compute the state feedback controller
27 F = sat( -P.K*x + P.kr*z_r, P.F_max);
28 end
29
30 %
31 % saturation function
32 function out = sat(in,limit)
33     if in > limit,       out = limit;
34     elseif in < -limit,   out = -limit;
35     else                 out = in;
36     end
37 end

```

11.5 Design Study C. Satellite Attitude Control



Homework Problem C.11

The objective of this problem is to implement state feedback controller using the full state. Start with the simulation files developed in Homework C.10.

- (a) Using the values for ω_{n_ϕ} , ζ_ϕ , ω_{n_θ} , and ζ_θ selected in Homework C.8, find the desired closed loop poles.
- (b) Add the state space matrices A , B , C , D derived in Homework C.6 to your param file.
- (c) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}_{A,B}) = n$.

- (d) Find the feedback gain K so that the eigenvalues of $(A - BK)$ are equal to desired closed loop poles. Find the reference gain k_r so that the DC-gain from ϕ_r to ϕ is equal to one.
- (e) Implement the state feedback scheme and tune the closed loop poles to get good response. You should be able to get much faster response using state space methods.

Solution

From HW C.6, the state space equations for the satellite are given by

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ -0.0300 & 0.0300 & -0.0100 & 0.0100 \\ 0.1500 & -0.1500 & 0.0500 & -0.0500 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 0.2 \\ 0 \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix} x\end{aligned}\quad (11.39)$$

where Equation (11.39) represents the measured outputs. The reference output is the angle ϕ , which implies that

$$y_r = (0 \ 1 \ 0 \ 0) x.$$

Step 1. The controllability matrix is

$$\mathcal{C}_{A,B} = [B, AB, A^2B, A^3B] = \begin{pmatrix} 0 & 0.2000 & -0.0020 & -0.0059 \\ 0 & 0 & 0.0100 & 0.0294 \\ 0.2000 & -0.0020 & -0.0059 & 0.0007 \\ 0 & 0.0100 & 0.0294 & -0.0036 \end{pmatrix}.$$

The determinant is $\det(\mathcal{C}_{A,B}) = -36,000 \neq 0$, therefore the system is controllable.

Step 2. The open loop characteristic polynomial is

$$\Delta_{ol}(s) = \det(sI - A) = s^4 + 0.0600s^3 + 0.18s^2$$

which implies that

$$\begin{aligned}\mathbf{a}_A &= (0.06, 0.18, 0, 0) \\ \mathcal{A}_A &= \begin{pmatrix} 1 & 0.06 & 0.18 & 0 \\ 0 & 1 & 0.06 & 0.18 \\ 0 & 0 & 1 & 0.06 \\ 0 & 0 & 0 & 1 \end{pmatrix}.\end{aligned}$$

Step 3. The desired closed loop polynomial is

$$\begin{aligned}\Delta_{cl}^d(s) &= (s^2 + 2\zeta_\theta\omega_{n_\theta}s + \omega_{n_\theta}^2)(s^2 + 2\zeta_\phi\omega_{n_\phi}s + \omega_{n_\phi}^2) \\ &= s^4 + 4.9275s^3 + 12.1420s^2 + 14.6702s + 8.8637,\end{aligned}$$

when $\omega_\theta = 1.9848$, $\zeta_\theta = 0.707$, $\omega_\phi = 1.5$, $\zeta_\phi = 0.707$, which implies that
 $\alpha = (4.9275, 12.1420, 14.6702, 8.8637)$.

Step 4. The gains are therefore given by

$$\begin{aligned} K &= (\alpha - \mathbf{a}_A) \mathcal{A}_A^{-1} \mathcal{C}_{A,B}^{-1} \\ &= (40.2842, 255.1732, 24.3375, 366.1825) \end{aligned}$$

The feedforward reference gain $k_r = -1/C_r(A - BK)^{-1}B$ is computed using $C_r = (0, 1, 0, 0)$, which gives

$$\begin{aligned} k_r &= \frac{-1}{C_r(A - BK)^{-1}B} \\ &= 295.4573. \end{aligned}$$

Alternatively, we could have used the following Matlab script.

```

1 % satellite parameter file
2
3 % tuning parameters
4 wn_th    = 0.6;
5 zeta_th = 0.707;
6 wn_phi   = 1.1;
7 zeta_phi = 0.707;
8
9 % state space design
10 A = [...
11     0, 0, 1, 0; ...
12     0, 0, 0, 1; ...
13     -P.k/P.Js, P.k/P.Js, -P.b/P.Js, P.b/P.Js; ...
14     P.k/P.Jp, -P.k/P.Jp, P.b/P.Jp, -P.b/P.Jp; ...
15
16 ];
17 B = [0; 0; 1/P.Js; 0];
18 C = [...
19     1, 0, 0, 0; ...
20     0, 1, 0, 0; ...
21 ];
22
23 % gain selection
24 ol_char_poly = charpoly(A);
25 des_char_poly = conv([1,2*zeta_th*wn_th,wn_th^2], ...
26     [1,2*zeta_phi*wn_phi,wn_phi^2]);
27 des_poles = roots(des_char_poly);
28 % is the system controllable?
29 if rank(ctrb(A,B))<4
30     disp('System Not Controllable');
31 else
32     P.K = place(A,B,des_poles);
33     P.kr = -1/([0, 1, 0, 0]*inv(A-B*P.K)*B);
34 end
35
36 sprintf('K: (%f, %f, %f, %f)\nkr: %f\n', P.K(1), P.K(2), P.K(3), P.K(4), P.kr)

```

The Matlab code for the controller is given by

```

1 function tau=satellite_ctrl(in,P)
2     phi_d    = in(1);
3     theta   = in(2);
4     phi      = in(3);
5     t       = in(4);
6
7     % use a digital differentiator to find phidot and thetadot
8     persistent phidot
9     persistent phi_d1
10    persistent thetadot
11    persistent theta_d1
12    % reset persistent variables at start of simulation
13    if t<P.Ts,
14        phidot      = 0;
15        phi_d1     = 0;
16        thetadot   = 0;
17        theta_d1   = 0;
18    end
19    phidot = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts)*phidot...
20        + 2/(2*P.sigma+P.Ts)*(phi-phi_d1);
21    thetadot = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts)*thetadot...
22        + 2/(2*P.sigma+P.Ts)*(theta-theta_d1);
23    phi_d1 = phi;
24    theta_d1 = theta;
25
26    % construct the state
27    x = [theta; phi; thetadot; phidot];
28    % compute the state feedback controller
29    tau = sat( -P.K*x + P.kr*phi_d, P.taumax);
30 end
31
32 %
33 % saturation function
34 function out = sat(in,limit)
35     if      in > limit,      out = limit;
36     elseif in < -limit,     out = -limit;
37     else                  out = in;
38     end
39 end

```

Notes and References

Controllability is typically defined as follows. The state space system

$$\dot{x} = Ax + Bu$$

is said to be controllable from initial state x_0 , if there exists a control function $u(t)$ defined on some interval $t \in [0, T]$ that transitions the state to the origin. It can be shown that for linear time-invariant systems, the system is controllable if and only iff the rank of the controllability matrix $\mathcal{C}_{A,B}$ is equal to the size of the state n . For single input system, this is equivalent to $\mathcal{C}_{A,B}$ being invertible.

CHAPTER 11. FULL STATE FEEDBACK

Controllability is an interesting and deep topic that extends nicely to nonlinear and multi-input systems. For further information see [5, 6, 12].

Ackermann's formula given in Equation (11.31) is only valid single input systems. In the multi-input case, the controllability matrix $\mathcal{C}_{A,B}$ is not square and cannot therefore be inverted. However, a more genera formula can be derived based on the range space if $\mathcal{C}_{A,B}$ has rank n . The more general formula is implemented by the Matlab place command. For additional information on derivation of the feedback gain in the MIMO case see [6].

Chapter 12

Integrator with Full State Feedback

12.1 Theory

In the previous chapter we derived a full state feedback controller that stabilized the system and ensured that the DC-gain from the reference $r(t)$ to the reference output $y_r(t)$ is equal to one. If the model is precise meaning that we know all of the parameters exactly and we have not neglected any dynamics, or if there are no disturbances on the system, then the output will track the reference as time goes to infinity. However, models are never precise, and there are almost always disturbances on the system, which implies that there will be steady state error in the response. In Chapter 9 we showed that adding an integrator to the controller is an effective way of eliminating steady state error caused by model mismatch and input/output disturbances. In this section we will show how to add an integrator to the full state feedback controller derived in Chapter 11.

A block diagram of the system with full state feedback and an integrator is shown in Figure 12.1.

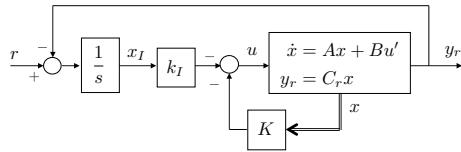


Figure 12.1: Full state feedback with integrator.

Let the state of the integrator be defined as x_I , as shown in Figure 12.1, then

$$\dot{x}_I = r - y_r = r - C_r x.$$

In Appendix e we argue using the root locus, that adding an integrator after

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

the other gains are selected, will change all of the pole locations. Since the pole placement design methods described in Part III were limited to second order system, we did not have an effective method for placing the system poles and the integrator pole simultaneously. On the other hand, the pole placement technique described in Chapter 11 was not limited to second order systems. As such, there should be a way to use the methods of Chapter 11 to simultaneously place both the system poles and the integrator pole. To do so, we augment the states of the original system with the integrator state x_I . Since the dynamics of the original system are given by $\dot{x} = Ax + Bu$, the dynamics of the augmented system are given by

$$\begin{pmatrix} \dot{x} \\ \dot{x}_I \end{pmatrix} = \begin{pmatrix} A & \mathbf{0} \\ -C & \mathbf{0} \end{pmatrix} \begin{pmatrix} x \\ x_I \end{pmatrix} + \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} u + \begin{pmatrix} 0 \\ 1 \end{pmatrix} r, \quad (12.1)$$

where the state of the augmented system is $(x^\top, x_I)^\top$, the interaction matrix, or the “A-matrix,” of the augmented system is

$$A_1 = \begin{pmatrix} A & \mathbf{0} \\ -C & \mathbf{0} \end{pmatrix},$$

and the “B-matrix” of the augmented system is

$$B_1 = \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix}.$$

The first question that might be asked is whether the augmented system (A_1, B_1) is controllable when the original system (A, B) is controllable. Since the original system is controllable, we know that the controllability matrix

$$\mathcal{C}_{A,B} = (B \ AB \ \dots \ A^{n-1}B)$$

has rank n . For the augmented system, the controllability matrix is

$$\mathcal{C}_{A_1,B_1} = \begin{pmatrix} B & AB & A^2B & \dots & A^{n-1}B & A^nB \\ 0 & -C_rB & -C_rAB & \dots & -C_rA^{n-2}B & -C_rA^{n-1}B \end{pmatrix},$$

which will have rank $n+1$ if one of the following conditions is true:

$$\begin{aligned} C_rB &\neq 0 \\ C_rAB &\neq 0 \\ &\vdots \\ C_rA^{n-1}B &\neq 0. \end{aligned}$$

When the augmented system is controllable, we can use the pole placement technique described in Chapter 11 to place the system poles and the pole of the integrator simultaneously using the augmented system. For example, letting A_1 and B_1 represent the augmented system, and using the Matlab place command

```
1 K1 = place(A1,B1,p1,...,pn,pI)
```

where p_I is the integrator pole. The resulting controller is

$$\begin{aligned} u(t) &= -K_1 \begin{pmatrix} x \\ x_I \end{pmatrix} = -(K \ k_I) \begin{pmatrix} x \\ x_I \end{pmatrix} = -Kx - k_I x_I \\ &= \underbrace{-Kx(t)}_{\text{state feedback}} - \underbrace{k_I \int_0^t (r(\tau) - y_r(\tau)) d\tau}_{\text{integral feedback}} . \end{aligned}$$

We can show that adding the integrator ensures that the DC-gain of the closed loop system is equal to one. Indeed from Equation (12.1) the augmented closed loop system is

$$\begin{aligned} \begin{pmatrix} \dot{x} \\ \dot{x}_I \end{pmatrix} &= \begin{pmatrix} A & \mathbf{0} \\ -C & \mathbf{0} \end{pmatrix} \begin{pmatrix} x \\ x_I \end{pmatrix} - \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} (K \ k_I) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} r \\ &= \begin{pmatrix} A - BK & -Bk_I \\ -C & 0 \end{pmatrix} \begin{pmatrix} x \\ x_I \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} r, \end{aligned}$$

where the reference output equation is

$$y_r = (C \ 0) \begin{pmatrix} x \\ x_I \end{pmatrix}.$$

Therefore, the transfer function from $R(s)$ to $Y_r(s)$ is

$$Y_r(s) = (C \ 0) \left(sI - \begin{pmatrix} A - BK & -Bk_I \\ -C & 0 \end{pmatrix} \right)^{-1} \begin{pmatrix} 0 \\ 1 \end{pmatrix} R(s),$$

which implies that the DC-gain is

$$k_{DC} = - (C \ 0) \left(\begin{pmatrix} A - BK & -Bk_I \\ -C & 0 \end{pmatrix} \right)^{-1} \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Defining $M \triangleq (A - BK)^{-1}$, and using the Matrix inversion formula for block matrices given in Appendix f Equation (f.2) we get

$$\begin{aligned} k_{DC} &= - (C \ 0) \begin{pmatrix} M - MBk_I(CMBk_I)^{-1}CM & -MBk_I(CMBk_I)^{-1} \\ -(CMBk_I)^{-1}cM & (CMBk_I)^{-1} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= CMBk_I(CMBk_I)^{-1} \\ &= 1. \end{aligned}$$

12.1.1 Integral Feedback for Linearized Systems

Suppose that the state space model is from a linearized system, i.e, suppose that $x \in \mathbb{R}^n$ is the state and that $x_e \in \mathbb{R}^n$ is the equilibrium state, and that $\hat{x} = x - x_e$

is the state linearized around the equilibrium. Similarly, suppose that $u \in \mathbb{R}^m$ is the input, that $u_e \in \mathbb{R}^m$ is the input at equilibrium, and that $\tilde{u} = u - u_e$ is the input linearized around equilibrium, and that $y_r \in \mathbf{R}^p$ is the reference output, $y_{re} = C_r x_e \in \mathbf{R}^p$ is the reference output at equilibrium, $\tilde{y}_r = y_r - y_{re}$ is the reference output linearized about equilibrium, and the reference input is given by $\tilde{r} = r - y_{re}$. In this case the integral state is selected as $\tilde{x}_I = \int(\tilde{r} - \tilde{y}_r)d\tau$, and the augmented system is given by

$$\begin{pmatrix} \dot{\tilde{x}} \\ \dot{\tilde{x}}_I \end{pmatrix} = \begin{pmatrix} A & \mathbf{0} \\ -C_r & \mathbf{0} \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{x}_I \end{pmatrix} + \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} \tilde{u} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tilde{r}.$$

The pole placement design proceeds as in the previous section with the resulting controller being

$$\tilde{u} = -K\tilde{x} - k_I \int_{-\infty}^t \tilde{e}(\tau)d\tau.$$

Noting that

$$\tilde{e} = \tilde{r} - \tilde{y} = (r - y_{re}) - (y_r - y_{re}) = r - y_r$$

we get that the controller for the linearized system is given by

$$u = u_e - K(x - x_e) - k_I \int_{-\infty}^t (r(\tau) - y_r(\tau))d\tau.$$

12.2 Summary of Design Process - State Feedback with Integrator

The design process for adding an integrator to state feedback can be summarized as follows. Given the plant

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y_r &= C_r x, \end{aligned}$$

where we desire that y_r tracks the reference r using integral action using the controller

$$u(t) = -Kx(t) - k_I \int_{-\infty}^t (r(\tau) - y_r(\tau))d\tau,$$

with desired closed loop gains at $p_1, p_2, \dots, p_n, p_I$.

Step 1. Augment the state evolution equation as

$$\begin{pmatrix} \dot{x} \\ \dot{x}_I \end{pmatrix} = \begin{pmatrix} A & \mathbf{0} \\ -C_r & \mathbf{0} \end{pmatrix} \begin{pmatrix} x \\ x_I \end{pmatrix} + \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} u + \begin{pmatrix} \mathbf{0} \\ I \end{pmatrix} r,$$

and let

$$\begin{aligned} A_1 &= \begin{pmatrix} A & \mathbf{0} \\ -C_r & \mathbf{0} \end{pmatrix} \\ B_1 &= \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} \end{aligned}$$

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

Step 2. Follow the pole placement procedure outlined in Chapter 11 to find K_1 that places the poles of the augmented system at p_1, p_2, \dots, p_n , and p_I .

Step 3. Find the gains K and k_I as

$$K = K_1(1 : n)$$

$$k_I = K_1(n + 1).$$

If Matlab is available, these steps can be implemented using the following script.

```

1 % augment the system
2 A1 = [A, zeros(n,1); -Cr, 0];
3 B1 = [B; 0];
4 % compute the controllability matrix
5 C1 = ctrb(A1,B1);
6 % check for controllability
7 if rank(CC) ~= size(A1,1),
8     disp('The system is controllable')
9 else
10    % place the poles at p1,p2,...,pn, pI
11    K1 = place(A1,B1,p1,p2,p3,p4,pI);
12    % extract the state feedback and integral gains from KK
13    K = K1(1:n);
14    kI = K1(n+1);
15 end

```

12.2.1 Simple example

Given the system

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} x + \begin{pmatrix} 0 \\ 4 \end{pmatrix} u$$

$$y_r = \begin{pmatrix} 5 & 0 \end{pmatrix} x$$

find the feedback gain K to place the closed loop poles at $-1 \pm j$, and the reference gain k_r so that the DC-gain is one, and add an integrator with gain at $p_I = -0.1$.

Step 1. Form the augmented system

$$A_1 = \begin{pmatrix} A & \mathbf{0} \\ -C_r & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 3 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

$$B_1 = \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix}$$

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

Step 2. To place the poles at $p_1 = -1 + j$, $p_2 = -1 - j$, $p_I = -0.1$, we first form the controllability matrix

$$\mathcal{C}_{A_1, B_1} = [B_1, A_1 B_1, A_1^2 B_1] = \begin{pmatrix} 0 & 4 & 12 \\ 4 & 12 & 44 \\ 0 & 0 & 4 \end{pmatrix}.$$

The determinant is $\det(\mathcal{C}_{A_1, B_1}) = 3 \neq 0$, therefore the system is controllable.

The open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A_1) = \det \begin{pmatrix} s & -1 & 0 \\ -2 & s-3 & 0 \\ 1 & 0 & s \end{pmatrix} = s^3 - 3s^2 - 2s,$$

which implies that

$$\mathbf{a}_{A_1} = (-3, -2, 0)$$

$$\mathcal{A}_{A_1} = \begin{pmatrix} 1 & -3 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix}.$$

The desired closed loop polynomial

$$\Delta_{cl}^d(s) = (s + 1 - j)(s + 1 + j)(s + 0.1) = s^3 + 2.1s^2 + 2.2s + 0.2,$$

which implies that

$$\boldsymbol{\alpha} = (2.1, 2.2, 0.2).$$

The augmented gains are therefore given as

$$K_1 = (\boldsymbol{\alpha} - \mathbf{a}_{A_1}) \mathcal{A}_{A_1}^{-1} \mathcal{C}_{A_1, B_1}^{-1}$$

$$= ((2.1, 2.2, 0.2) - (-3, -2, 0)) \begin{pmatrix} 1 & -3 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 4 & 12 \\ 4 & 12 & 44 \\ 0 & 0 & 4 \end{pmatrix}^{-1}$$

$$= (1.05, 1.275, 0.05)$$

Step 3. The feedback gains are therefore given by

$$K = K_1(1 : 2) = (1.05, 1.275)$$

$$k_I = K_1(3) = 0.05$$

Alternatively, we could have used the following Matlab script

```

1 % original state space system
2 A = [0, 1; 2, 4];
3 B = [0; 4];
4 Cr = [5 0];
5 % augment A and B to add integrator
6 A1 = [A, zeros(2,1); -Cr, 0];
7 B1 = [B; 0];
8 % check for controllability
9 CC = ctrb(A1,B1);
10 if rank(CC) ~= 3,
11     disp('The system is not controllable')
12 else
13     % place the poles at p1,p2, pI
14     K1 = place(A1,B1,[-1+j,-1-j,-0.1]);
15     % extract gains
16     K = K1(1:2);
17     k_I = K1(3);
18 end

```

12.3 Design Study A. Single Link Robot Arm



Homework Problem A.12

- (a) Modify the state feedback solution developed in Homework A.11 to add an integrator with anti-windup.
- (b) Add a disturbance to the system and allow the system parameters to change up to 20%.
- (c) Tune the integrator pole (and other gains if necessary) to get good tracking performance.

Solution

Step 1. The original state space equations are

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 1.0000 \\ 0 & -0.667 \end{pmatrix} x + \begin{pmatrix} 0 \\ 66.667 \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 \end{pmatrix} x,\end{aligned}$$

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

therefore the augmented system is

$$A_1 = \begin{pmatrix} A & \mathbf{0} \\ -C & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 & 1.0000 & 0 \\ 0 & -0.667 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

$$B_1 = \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 \\ 66.667 \\ 0 \end{pmatrix}$$

Step 2. After the design in HW A.11, the closed loop poles were located at $p_{1,2} = -3.1191 \pm j3.1200$. We will add the integrator pole at $p_I = -5$. The new controllability matrix

$$\mathcal{C}_{A_1, B_1} = [B_1, A_1 B_1, A_1^2 B_1] = \begin{pmatrix} 0 & 63.6512 & -39.2993 \\ 63.6512 & -39.2993 & 24.2640 \\ 0 & 0 & 63.6512 \end{pmatrix}.$$

The determinant is $\det(\mathcal{C}_{A_1, B_1}) = -2.5788e+05 \neq 0$, therefore the system is controllable.

The open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A_1) = \det \begin{pmatrix} s & -1.0000 & 0 \\ 0 & s + 0.667 & 0 \\ -1 & 0 & s \end{pmatrix} = s^3 + 0.6174s^2,$$

which implies that

$$\mathbf{a}_{A_1} = (0.6174, 0, 0)$$

$$\mathcal{A}_{A_1} = \begin{pmatrix} 1 & 0.6174 & 0 \\ 0 & 1 & 0.6174 \\ 0 & 0 & 1 \end{pmatrix}.$$

The desired closed loop polynomial

$$\Delta_{cl}^d(s) = (s + 3.1191 - j3.12)(s + 3.1191 + j3.12)(s + 5)$$

$$= s^3 + 12.7770s^2 + 69.1350s + 151.2500,$$

which implies that

$$\boldsymbol{\alpha} = (12.7770, 69.1350, 151.2500).$$

The augmented gains are therefore given as

$$K_1 = (\boldsymbol{\alpha} - \mathbf{a}_{A_1})\mathcal{A}_{A_1}^{-1}\mathcal{C}_{A_1, B_1}^{-1}$$

$$= (1.0370, 0.1817, -2.2687)$$

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

Step 3. The feedback gains are therefore given by

$$K = K_1(1 : 2) = (1.0370, \quad 0.1817)$$

$$k_I = K_1(3) = -2.2687$$

Alternatively, we could have used the following Matlab script

```

1 % sample rate
2 P.Ts = 0.01;
3
4 % dirty derivative gain
5 P.sigma = 0.05;
6 P.beta = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts); % dirty derivative gain
7
8 % equilibrium torque
9 P.theta_e = 0*pi/180;
10 P.tau_e = P.m*P.g*P.ell/2*cos(P.theta_e);
11
12 % saturation constraint
13 P.tau_max = 1;
14 tau_max = P.tau_max-P.tau_e;
15
16 %
17 % state space design
18 A = [...
19     0, 1;...
20     0, -3*P.b/P.m/(P.ell^2);...
21 ];
22 B = [0; 3/P.m/(P.ell^2) ];
23 C = [...
24     1, 0;...
25 ];
26 % form augmented system
27 A1 = [A, zeros(2,1); -C, 0];
28 B1 = [B; 0];
29
30 % tuning parameters
31 tr = 0.4;
32 zeta = 0.707;
33 integrator_pole = -5;
34
35 % desired closed loop polynomial
36 wn = 2.2/tr;
37 % gains for pole locations
38 des_char_poly = conv([1,2*zeta*wn,wn^2],poly(integrator_pole));
39 des_poles = roots(des_char_poly);
40
41
42 % is the system controllable?
43 if rank ctrb(A1,B1) ≠ 3,
44     disp('System Not Controllable');
45 else % if so, compute gains
46     K1 = place(A1,B1,des_poles);
47     P.K = K1(1:2);
48     P.ki = K1(3);

```

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

49 end

Matlab code that implements the associated controller listed below.

```

1 function tau=arm_ctrl(in,P)
2     theta_r = in(1);
3     theta   = in(2);
4     t       = in(3);
5
6     % use a digital differentiator to find thetadot
7     persistent thetadot
8     persistent theta_d1
9     % reset persistent variables at start of simulation
10    if t<P.Ts
11        thetadot    = 0;
12        theta_d1    = 0;
13    end
14    thetadot = P.beta*thetadot + (1-P.beta)*((theta-theta_d1)/P.Ts);
15    theta_d1 = theta;
16
17    % implement integrator
18    error = theta_r - theta;
19    persistent integrator
20    persistent error_d1
21    % reset persistent variables at start of simulation
22    if t<P.Ts==1
23        integrator = 0;
24        error_d1 = 0;
25    end
26    integrator = integrator + (P.Ts/2)*(error+error_d1);
27    error_d1 = error;
28
29
30    % construct the state
31    theta_e = 0;
32    x = [theta-theta_e; thetadot];
33    % compute equilibrium torque tau_e
34    tau_e = P.m*P.g*(P.ell/2)*cos(theta);
35    % compute total torque
36    tau_unsat = tau_e - P.K*x - P.ki*integrator;
37    tau = sat( tau_unsat, P.tau_max );
38
39    % integrator anti-windup
40    if P.ki≠0
41        integrator = integrator + P.Ts/P.ki*(tau-tau_unsat);
42    end
43 end
44
45 function out = sat(in,limit)
46     if      in > limit,      out = limit;
47     elseif in < -limit,     out = -limit;
48     else                 out = in;
49     end
50 end

```

12.4 Design Study B. Inverted Pendulum



Homework Problem B.12

- (a) Modify the state feedback solution developed in Homework B.11 to add an integrator with anti-windup to the position feedback.
- (b) Add a constant input disturbance of 0.5 Newtons to the input of the plant and allow the system parameters to change up to 20%.
- (c) Tune the integrator pole (and other gains if necessary) to get good tracking performance.

Solution

Step 1. The original state space equations are

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0 & -2.2167 & -0.0500 & 0 \\ 0 & 24.5238 & 0.1020 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 0.9524 \\ -1.9436 \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} x.\end{aligned}$$

The integrator will only be on $z = x_1$, therefore

$$C_r = (1 \ 0 \ 0 \ 0).$$

The augmented system is therefore

$$\begin{aligned}A_1 &= \begin{pmatrix} A & \mathbf{0} \\ -C_r & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 \\ 0 & -2.2167 & -0.0500 & 0 & 0 \\ 0 & 24.5238 & 0.1020 & 0 & 0 \\ -1.0000 & 0 & 0 & 0 & 0 \end{pmatrix} \\ B_1 &= \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0.9524 \\ -1.9436 \\ 0 \end{pmatrix}\end{aligned}$$

Step 2. After the design in HW B.11, the closed loop poles were located at $p_{1,2} = -1.4140 \pm j1.4144$, $p_{3,4} = -0.8000 \pm j0.6000$. We will add the

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

integrator pole at $p_I = -10$. The new controllability matrix

$$\begin{aligned} \mathcal{C}_{A_1, B_1} &= [B_1, A_1 B_1, A_1^2 B_1, A_1^3 B_1, A_1^4 B_1] \\ &= 1000 \begin{pmatrix} 0 & 0.0010 & -0.0000 & 0.0043 & -0.0004 \\ 0 & -0.0019 & 0.0001 & -0.0477 & 0.0028 \\ 0.0010 & -0.0000 & 0.0043 & -0.0004 & 0.1057 \\ -0.0019 & 0.0001 & -0.0477 & 0.0028 & -1.1691 \\ 0 & 0 & 0.0010 & -0.0000 & 0.0043 \end{pmatrix}. \end{aligned}$$

The determinant is nonzero, therefore the system is controllable.

The open loop characteristic polynomial

$$\begin{aligned} \Delta_{ol}(s) &= \det(sI - A_1) \\ &= \det \begin{pmatrix} s & 0 & -1.0000 & 0 & 0 \\ 0 & s & 0 & -1.0000 & 0 \\ 0 & 2.2167 & s + 0.0500 & 0 & 0 \\ 0 & -24.5238 & -0.1020 & s & 0 \\ 1.0000 & 0 & 0 & 0 & s \end{pmatrix} \\ &= s^5 + 0.0500s^4 - 24.5238s^3 - s^2, \end{aligned}$$

which implies that

$$\begin{aligned} \mathbf{a}_{A_1} &= (0.0500, -24.5238, -1, 0, 0) \\ \mathcal{A}_{A_1} &= \begin{pmatrix} 1 & 0.05 & -24.5238 & -1 & 0 \\ 0 & 1 & 0.05 & -24.5238 & -1 \\ 0 & 0 & 1 & 0.05 & -24.5238 \\ 0 & 0 & 0 & 1 & 0.05 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \end{aligned}$$

The desired closed loop polynomial

$$\begin{aligned} \Delta_{cl}^d(s) &= (s + 1.4140 - j1.4144)(s + 1.4140 + j1.4144) \dots \\ &\quad (s + 0.8 - j0.6)(s + 0.8 + j0.6)(s + 10) \\ 1.0000 &= s^5 + 18.2955s^4 + 117.3685s^3 + 397.6722s^2 + 576.9796s + 416.4551, \end{aligned}$$

which implies that

$$\boldsymbol{\alpha} = (18.2955, 117.3685, 397.6722, 576.9796, 416.4551).$$

The augmented gains are therefore given as

$$\begin{aligned} K_1 &= (\boldsymbol{\alpha} - \mathbf{a}_{A_1}) \mathcal{A}_{A_1, B_1}^{-1} \mathcal{C}_{A_1, B_1}^{-1} \\ &= (-29.4377, -85.6531, -21.4235, -19.8345, 21.2477) \end{aligned}$$

Step 3. The feedback gains are therefore given by

$$\begin{aligned} K &= K_1(1 : 4) = (-29.4377, -85.6531, -21.4235, -19.8345) \\ k_I &= K_1(5) = 21.2477 \end{aligned}$$

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

Alternatively, we could have used the following Matlab script

```

1 % inverted Pendulum parameter file
2
3 % dirty derivative parameters
4 P.sigma = 0.05; % cutoff freq for dirty derivative
5 P.beta = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts); % dirty derivative gain
6
7 % saturation limits
8 P.F_max = 5; % Max Force, N
9 theta_max = 30.0*pi/180.0; % Max theta, rads
10
11 %%%%%%%%%%%%%%%%
12 % state feedback control with integrator
13 %%%%%%%%%%%%%%%%
14 % tunning parameters
15 tr_z = 1.5; % rise time for position
16 tr_theta = 0.5; % rise time for angle
17 zeta_z = 0.707; % damping ratio position
18 zeta_th = 0.707; % damping ratio angle
19 integrator_pole = -10; % integrator pole
20
21 % state space equations
22 A = [...
23     0, 0, 1, 0;...
24     0, 0, 0, 1;...
25     0, -P.m1*P.g/P.m2, -P.b/P.m2, 0;...
26     0, (P.m1+P.m2)*P.g/P.m2/P.ell, P.b/P.m2/P.ell, 0;...
27 ];
28 B = [0; 0; 1/P.m2; -1/P.m2/P.ell];
29 C = [...
30     1, 0, 0, 0;...
31     0, 1, 0, 0;...
32 ];
33 % form augmented system
34 Cout = [1,0,0,0];
35 A1 = [A, zeros(4,1); -Cout, 0];
36 B1 = [B; 0];
37
38 % compute gains
39 wn_th = 2.2/tr_theta; % natural frequency for angle
40 wn_z = 2.2/tr_z; % natural frequency for position
41 des_char_poly = conv(conv([1,2*zeta_z*wn_z,wn_z^2],...
42 [1,2*zeta_th*wn_th,wn_th^2]),...
43 poly(integrator_pole));
44 des_poles = roots(des_char_poly);
45
46 % is the system controllable?
47 if rank ctrb(A1,B1) ≠ 5
48     disp('System Not Controllable');
49 else % if so, compute gains
50     K1 = place(A1,B1,des_poles);
51     P.K = K1(1:4);
52     P.ki = K1(5);
53 end
54

```

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

```

55 sprintf('K: [%f, %f, %f, %f]\nki: %f\n',...
56     P.K(1), P.K(2), P.K(3), P.K(4), P.ki)

```

Matlab code that implements the associated controller listed below.

```

1  function F=pendulum_ctrl(in,P)
2      z_r    = in(1);
3      z     = in(2);
4      theta = in(3);
5      t     = in(4);
6
7      % use a digital differentiator to find zdot and thetadot
8      persistent zdot
9      persistent z_d1
10     persistent thetadot
11     persistent theta_d1
12     % reset persistent variables at start of simulation
13     if t<P.Ts,
14         zdot      = 0;
15         z_d1      = 0;
16         thetadot   = 0;
17         theta_d1   = 0;
18     end
19     zdot = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts)*zdot...
20         + 2/(2*P.sigma+P.Ts)*(z-z_d1);
21     thetadot = (2*P.sigma-P.Ts)/(2*P.sigma+P.Ts)*thetadot...
22         + 2/(2*P.sigma+P.Ts)*(theta-theta_d1);
23     z_d1 = z;
24     theta_d1 = theta;
25
26     % integrator
27     error = z_r - z;
28     persistent integrator
29     persistent error_d1
30     % reset persistent variables at start of simulation
31     if t<P.Ts==1,
32         integrator = 0;
33         error_d1   = 0;
34     end
35     integrator = integrator + (P.Ts/2)*(error+error_d1);
36     error_d1 = error;
37
38     % construct the state
39     x = [z; theta; zdot; thetadot];
40     % compute the state feedback controller
41     F_unsat = -P.K*x - P.ki*integrator;
42     F = sat( F_unsat, P.F_max);
43
44     % integrator anti-windup
45     if P.ki≠0,
46         integrator = integrator + P.Ts/P.ki*(F-F_unsat);
47     end
48
49 end
50
51 %

```

```

52 % saturation function
53 function out = sat(in,limit)
54     if      in > limit,      out = limit;
55     elseif in < -limit,    out = -limit;
56     else                out = in;
57     end
58 end

```

12.5 Design Study C. Satellite Attitude Control



Homework Problem C.12

- (a) Modify the state feedback solution developed in Homework C.11 to add an integrator with anti-windup to the feedback loop for ϕ .
- (b) Add a constant input disturbance of 1 Newtons-meters to the input of the plant and allow the plant parameters to vary up to 20%.
- (c) Tune the integrator pole (and other gains if necessary) to get good tracking performance.

Solution

Step 1. The original state space equations are

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ -0.0300 & 0.0300 & -0.0111 & 0.0111 \\ 0.1357 & -0.1357 & 0.0500 & -0.0500 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 0.2105 \\ 0 \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} x.\end{aligned}$$

The integrator will only be on $\phi = x_2$, therefore

$$C_r = (0 \ 1 \ 0 \ 0).$$

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

The augmented system is therefore

$$A_1 = \begin{pmatrix} A & \mathbf{0} \\ -C_r & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 \\ -0.0300 & 0.0300 & -0.0111 & 0.0111 & 0 \\ 0.1357 & -0.1357 & 0.0500 & -0.0500 & 0 \\ 0 & -1.0000 & 0 & 0 & 0 \end{pmatrix}$$

$$B_1 = \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0.2105 \\ 0 \\ 0 \end{pmatrix}$$

Step 2. After the design in HW C.11, the closed loop poles were located at $p_{1,2} = -1.0605 \pm j1.0608$, $p_{3,4} = -0.7016 \pm j0.7018$. We will add the integrator pole at $p_I = -1$. The new controllability matrix

$$\mathcal{C}_{A_1, B_1} = [B_1, A_1 B_1, A_1^2 B_1, A_1^3 B_1, A_1^4 B_1]$$

$$= \begin{pmatrix} 0 & 0.2105 & -0.0023 & -0.0062 & 0.0008 \\ 0 & 0 & 0.0105 & 0.0279 & -0.0034 \\ 0.2105 & -0.0023 & -0.0062 & 0.0008 & 0.0010 \\ 0 & 0.0105 & 0.0279 & -0.0034 & -0.0044 \\ 0 & 0 & 0 & 0.0105 & 0.0279 \end{pmatrix}.$$

The determinant is nonzero, therefore the system is controllable.

The open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A_1)$$

$$= s^5 + 0.0611s^4 + 0.1657s^3,$$

which implies that

$$\mathbf{a}_{A_1} = (0.0611, 0.1657, 0, 0, 0)$$

$$\mathcal{A}_{A_1} = \begin{pmatrix} 1 & 0.0611 & 0.1657 & 0 & 0 \\ 0 & 1 & 0.0611 & 0.1657 & 0 \\ 0 & 0 & 1 & 0.0611 & 0.1657 \\ 0 & 0 & 0 & 1 & 0.0611 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The desired closed loop polynomial

$$\Delta_{cl}^d(s) = (s + 1.0605 - j1.0608)(s + 1.0605 + j1.0608) \dots$$

$$(s + 0.7016 - j0.7018)(s + 0.7016 + j0.7018)(s + 1)$$

$$= s^5 + 3.4038s^4 + 5.2934s^3 + 4.4761s^2 + 2.0221s + 0.4356,$$

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

which implies that

$$\boldsymbol{\alpha} = (3.4038, \quad 5.2934 \quad 4.4761, \quad 2.0221, \quad 0.4356).$$

The augmented gains are therefore given as

$$\begin{aligned} K_1 &= (\boldsymbol{\alpha} - \mathbf{a}_{A_1}) \mathcal{A}_{A_1}^{-1} \mathcal{C}_{A_1, B_1}^{-1} \\ &= (19.1496, \quad 43.4140, \quad 16.7190, \quad 111.6301, \quad -14.5200). \end{aligned}$$

Step 3. The feedback gains are therefore given by

$$\begin{aligned} K &= K_1(1 : 2) = (19.1496, \quad 43.4140, \quad 16.7190, \quad 111.6301) \\ k_I &= K_1(3) = -14.5200. \end{aligned}$$

Alternatively, we could have used the following Matlab script

```

1 % satellite parameter file
2
3
4 %%%%%%
5 % state feedback control with integrator
6 %%%%%%
7 % tuning parameters
8 wn_th = 0.6;
9 zeta_th = 0.707;
10 wn_phi = 1.1;
11 zeta_phi = 0.707;
12 integrator_pole = -1;
13
14 % state space design
15 A = [...
16     0, 0, 1, 0; ...
17     0, 0, 0, 1; ...
18     -P.k/P.Js, P.k/P.Js, -P.b/P.Js, P.b/P.Js; ...
19     P.k/P.Jp, -P.k/P.Jp, P.b/P.Jp, -P.b/P.Jp; ...
20 ];
21 ];
22 B = [0; 0; 1/P.Js; 0];
23 C = [...
24     1, 0, 0, 0; ...
25     0, 1, 0, 0; ...
26 ];
27
28 % form augmented system
29 Cout = [0, 1, 0, 0];
30 A1 = [A, zeros(4,1); -Cout, 0];
31 B1 = [B; 0];
32
33 % compute gains
34 ol_char_poly = charpoly(A);
35 des_char_poly = conv(conv([1, 2*zeta_th*wn_th, wn_th^2], ...
36 [1, 2*zeta_phi*wn_phi, wn_phi^2]), ...

```

CHAPTER 12. INTEGRATOR WITH FULL STATE FEEDBACK

```
37         poly(integrator_pole));
38 des_poles = roots(des_char_poly);
39
40 % is the system controllable?
41 if rank ctrb(A1,B1) ≠ 5,
42     disp('System Not Controllable');
43 else % if so, compute gains
44     K1 = place(A1,B1,des_poles);
45     P.K = K1(1:4);
46     P.ki = K1(5);
47 end
48
49 sprintf('K: [%f, %f, %f, %f]\nki: %f\n',...
50     P.K(1), P.K(2), P.K(3), P.K(4), P.ki)
```

See <http://controlbook.byu.edu> for the complete solution.

Notes and References

Integral control for linear state space systems is a well established technique. Additional examples can be found in [19, 5, 4]. A nice introduction to integral control for nonlinear systems is given in [20].

Chapter 13

Observers

13.1 Theory

In the previous two chapters we assumed that the full state x was available to the controller for full state feedback. In this chapter we remove that assumption and show how the state can be reconstructed given the measurements. We will assume that the state space model is given by

$$\dot{x} = Ax + Bu \quad (13.1)$$

$$\begin{aligned} y_r &= C_r x \\ y_m &= C_m x, \end{aligned} \quad (13.2)$$

where $y_r \in \mathbb{R}^r$ is the reference output as discussed in the previous two chapters, and $y_m \in \mathbb{R}^p$ is the measured output, or the signals measured by the sensors. While the objective of the controller is the force $y_r(t)$ to converge to the reference $r(t)$, the objective of the observer is to estimate the state $x(t)$ given a history of the input $u(t)$ and the measured output $y_m(t)$. In this book, the estimate of the state will be denoted as $\hat{x}(t)$.

A block diagram of the closed loop system with observer based control is shown in Figure 13.1. Comparing Figure 13.1 to the case of full state feedback in Figure 11.4 note that the controller uses the estimated state \hat{x} instead of the full state x . In other words, the controller in Equation (11.36) becomes

$$u = -K\hat{x} + k_r r.$$

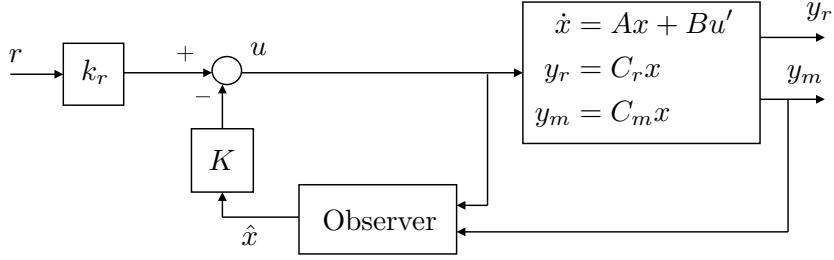


Figure 13.1: Closed loop system with observer based feedback.

In this chapter we will design a so-called Luenberger observer that has the structure given by

$$\hat{x} = \underbrace{A\hat{x} + Bu}_{\text{predictor}} + \underbrace{L(y_m - C_m\hat{x})}_{\text{corrector}}. \quad (13.3)$$

The first term in Equation (13.3), i.e., $\hat{x} = A\hat{x} + Bu$ is a copy of the equations of motion for the plant given in Equation (13.1). The second term given by $L(y_m - C_m\hat{x})$ is a correction term, where L is the observer gain, and $(y_m - C_m\hat{x})$ is the difference between the measured output $y_m = C_m x$ and the predicted value of the measured output $C_m\hat{x}$. The term $y_m - C_m x$ is called the *innovation*.

Defining the observer error as $e = x - \hat{x}$ and differentiating with respect to time gives

$$\begin{aligned}\dot{e} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + Bu - (A\hat{x} + Bu + Ly_m - LC_m\hat{x}) \\ &= A(x - \hat{x}) + LC_m x - LC_m\hat{x} \\ &= (A - LC_m)(x - \hat{x}) \\ &= (A - LC_m)e.\end{aligned}$$

Therefore, the poles that govern the time-evolution of the observation error are given by the eigenvalues of $A - LC_m$. The observer design problem is therefore to select L to place the eigenvalues of $A - LC_m$ at locations specified by the control engineer. The derivation of a formula for the observer gain L proceeds in a manner similar to the derivation of the feedback gain K in Chapter 11.

13.1.1 Observer Design using Observer Canonic Form

Given the general SISO monic transfer function

$$Y(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} U(s), \quad (13.4)$$

where $m > n$, and taking the inverse Laplace transform gives

$$\frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \dot{y} + a_0 y(t) = b_m \frac{d^m u}{dt^m} + b_{m-1} \frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_1 \dot{u} + b_0 u(t).$$

Solving for the n^{th} derivative of y and collecting terms with similar order derivatives gives

$$\begin{aligned}\frac{d^n y}{dt^n} &= -a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} - a_{m+1} \frac{d^{m+1} y}{dt^{m+1}} \\ &\quad + \left(b_m \frac{d^m u}{dt^m} - a_m \frac{d^m y}{dt^m} \right) + \cdots + (b_1 \dot{u} - a_1 \dot{y}) + (b_0 u - a_0 y).\end{aligned}$$

Integrating n times gives

$$\begin{aligned}y(t) &= \int \left[-a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} - \cdots - a_{m+1} \frac{d^{m+1} y}{dt^{m+1}} \right. \\ &\quad \left. + \int \left[\left(b_m \frac{d^m u}{dt^m} - a_m \frac{d^m y}{dt^m} \right) + \cdots + \int \left[(b_1 \dot{u} - a_1 \dot{y}) + \int (b_0 u - a_0 y) \right] \dots \right] \dots \right].\end{aligned}\tag{13.5}$$

The analog computer implementation of Equation (13.5) is shown in Figure 13.2.

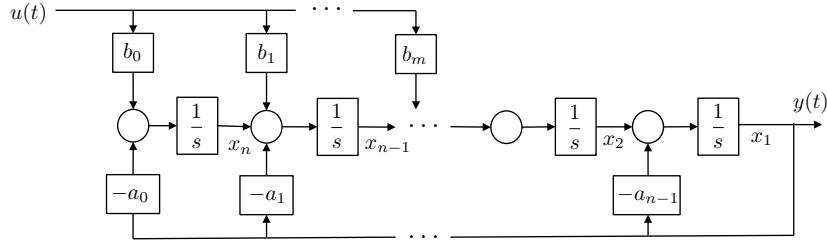


Figure 13.2: Block diagram showing the analog computer implementation of Equation (13.5).

Labeling the states as the output of the integrators, as shown in Figure 13.2 and writing the equations in linear state space form gives

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{pmatrix} = \begin{pmatrix} -a_{n-1} & 1 & \dots & 0 & 0 \\ -a_{n-2} & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ -a_1 & 0 & \dots & 0 & 1 \\ -a_0 & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_m \\ b_1 \\ b_0 \end{pmatrix} u \tag{13.6}$$

$$= (1 \ 0 \ \dots \ 0 \ 0) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}. \tag{13.7}$$

We define the observer canonic realization as

$$\begin{aligned}\dot{x}_o &= A_o x_o + B_o u \\ y &= C_o x_o,\end{aligned}$$

where

$$A_o \triangleq \begin{pmatrix} -a_{n-1} & 1 & \dots & 0 & 0 \\ -a_{n-2} & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ -a_1 & 0 & \dots & 0 & 1 \\ -a_0 & 0 & \dots & 0 & 0 \end{pmatrix} \quad (13.8)$$

$$B_o \triangleq \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_m \\ \vdots \\ b_1 \\ b_0 \end{pmatrix} \quad (13.9)$$

$$C_o \triangleq (1 \ 0 \ \dots \ 0 \ 0). \quad (13.10)$$

Comparing the state space equations in observer canonic form in Equations (13.8)–(13.10) with the state space equations for control canonic form in Equations (11.14)–(11.16), we see that

$$\begin{aligned}A_o &= A_c^\top \\ B_o &= C_c^\top \\ C_o &= B_c^\top.\end{aligned}$$

Comparing the state equation in Equation (13.6) with the transfer function in Equation (13.4) we see that the eigenvalues of A_o are the roots of the open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A_o) = s^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0.$$

Since C_o in Equation (13.7) has the structure of one in the first element followed by zeros, we see that the structure of $A_o - L_o C_o$ is given by

$$A_o - L_o C_o = \begin{pmatrix} -(a_{n-1} + \ell_1) & 1 & \dots & 0 & 0 \\ -(a_{n-2} + \ell_2) & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ -(a_1 + \ell_{n-1}) & 0 & \dots & 0 & 1 \\ -(a_0 + \ell_n) & 0 & \dots & 0 & 0 \end{pmatrix}.$$

The characteristic polynomial for the closed loop observation error is therefore given by

$$\begin{aligned}\Delta_{cl,obs}(s) &= \det(sI - (A_o - L_o C_o)) \\ &= s^n + (a_{n-1} + \ell_1)s^{n-1} + \cdots + (a_1 + \ell_{n-1})s + (a_0 + \ell_n).\end{aligned}$$

If the desired characteristic polynomial for the closed loop observation error is

$$\Delta_{cl,obs}^d(s) = s^n + \beta_{n-1}s^{n-1} + \cdots + \beta_1s + \beta_0, \quad (13.11)$$

then we have that

$$\begin{aligned}a_{n-1} + \ell_1 &= \beta_{n-1} \\ a_{n-2} + \ell_2 &= \beta_{n-2} \\ &\vdots \\ a_1 + \ell_{n-1} &= \beta_1 \\ a_0 + \ell_n &= \beta_0.\end{aligned}$$

Defining the vectors

$$\begin{aligned}\mathbf{a}_A &\stackrel{\triangle}{=} (a_{n-1}, a_{n-2}, \dots, a_1, a_0) \\ \boldsymbol{\beta} &\stackrel{\triangle}{=} (\beta_{n-1}, \beta_{n-2}, \dots, \beta_1, \beta_0)^\top \\ L_o &\stackrel{\triangle}{=} (\ell_1, \ell_2, \dots, \ell_{n-1}, \ell_n)^\top,\end{aligned}$$

we have that the observer gain satisfies

$$L_o = \boldsymbol{\beta} - \mathbf{a}_A^\top, \quad (13.12)$$

where the subscript “o” is used to emphasize that the gains are when the state equations are in observer canonic form. As an example, suppose that the transfer function model of the system is given by

$$Y(s) = \frac{9s + 20}{s^3 + 6s^2 - 11s + 8} U(s),$$

then the state space equations in observer canonic form are given by

$$\begin{aligned}\dot{x}_o &= \begin{pmatrix} -6 & 1 & 0 \\ 11 & 0 & 1 \\ -8 & 0 & 0 \end{pmatrix} x_o + \begin{pmatrix} 0 \\ 9 \\ 20 \end{pmatrix} u \\ y &= (1 \ 0 \ 0) x_o.\end{aligned}$$

The open loop characteristic polynomial is

$$\Delta_{ol}(s) = s^3 + 6s^2 - 11s + 8,$$

with roots at -7.5885 , and $0.7942 \pm j0.6507$. Suppose that the desired poles for the observation error are at -10 and $-20 \pm j20$, then the desired closed loop polynomial for the observation error is

$$\Delta_{cl,obs}^d(s) = (s + 10)(s + 20 - j20)(s + 20 + j20) = s^3 + 50s^2 + 1200s + 8000,$$

then $\mathbf{a}_{A_o} = (6, -11, 8)$, $\boldsymbol{\beta} = (50, 1200, 8000)^\top$, and the observer gain that places the eigenvalues of $A_o - L_o C_o$ at the desired locations is given by

$$L_o = \boldsymbol{\beta} - \mathbf{a}_{A_o}^\top = \begin{pmatrix} 50 \\ 1200 \\ 8000 \end{pmatrix} - \begin{pmatrix} 6 \\ -11 \\ 8 \end{pmatrix} = \begin{pmatrix} 44 \\ 1211 \\ 7992 \end{pmatrix}.$$

13.1.2 Observer Design: General State Space Equations

The formula given in Equation (13.12) assumes that the state space equations are in observer canonic form. Similar to the general full state feedback case discussed in Section 11.1.3, a formula for the observer gains can be derived for general state space equations. Following the method used in Section 13.12, the strategy is to find a state transformation matrix that converts the general state space equations into observer canonic form. Rather than repeating the steps in Section 13.12 we will derive the formula for the observer gain by noting the similarities between the design problem for full state feedback, and the design problem for the observer gain.

For full state feedback, the feedback gains K that place the eigenvalues of $A - BK$ at the roots of the desired characteristic equation given in Equation (11.19) is given by Equation (11.31) or

$$K = (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1}, \quad (13.13)$$

where

$$\mathcal{A}_A = \begin{pmatrix} 1 & a_{n-1} & a_{n-2} & \dots & a_2 & a_1 \\ 0 & 1 & a_{n-1} & \vdots & a_3 & a_2 \\ 0 & 0 & 1 & \vdots & a_4 & a_3 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & 1 & a_{n-1} \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix},$$

$$\mathcal{C}_{A,B} = (B \ AB \ A^2B \ \dots \ A^{n-1}B),$$

and where the subscripts are used to indicate that \mathcal{A}_A is a function of the matrix A , and $\mathcal{C}_{A,B}$ is a function of the matrices A and B .

The observer design problem is to find the observer gain L that places the eigenvalues of $A - LC$ at the roots of Equation (13.11). Recalling from Appendix f that for a general square matrix M , the eigenvalues of M equal the eigenvalues of M^\top , the observer problem can be posed as finding L that places

the eigenvalues of $A^\top - C^\top L^\top$ at the roots of Equation (13.11). This problem is identical to the full state feedback gain problem with A replaced by A^\top , B replaced by C^\top , and K replaced by L^\top . Making the appropriate substitutions in Ackermann's formula (13.13) gives

$$L^\top = (\beta^\top - \mathbf{a}_{A^\top}) \mathcal{A}_{A^\top}^{-1} \mathcal{C}_{A^\top, C^\top}^{-1} \quad (13.14)$$

Since $\det(sI - A) = \det(sI - A^\top)$, the characteristic equation for A and A^\top are identical, which implies that $\mathbf{a}_A = \mathbf{a}_{A^\top}$ and $\mathcal{A}_A = \mathcal{A}_{A^\top}$. Taking the transpose of Equation (13.14) gives

$$L = \mathcal{C}_{A^\top, C^\top}^{-\top} \mathcal{A}_A^{-\top} (\beta - \mathbf{a}_A^\top), \quad (13.15)$$

where we have used the notation

$$M^{-\top} \triangleq (M^{-1})^\top = (M^\top)^{-1}.$$

Defining the *observability matrix*

$$\begin{aligned} \mathcal{O}_{A,C} &\triangleq \mathcal{C}_{A^\top, C^\top}^\top \\ &= (C^\top \quad A^\top C^\top \quad (A^\top)^2 C^\top \quad \dots \quad (A^\top)^{n-1} C^\top)^\top \\ &= \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{pmatrix}, \end{aligned}$$

the observer gain (13.15) can be written as

$$L = \mathcal{O}_{A,C}^{-1} \mathcal{A}_A^{-\top} (\beta - \mathbf{a}_A^\top), \quad (13.16)$$

where L is well defined if the observability matrix $\mathcal{O}_{A,C}$ is invertible. Accordingly, the system (A, C) is said to be *observable* if $\text{rank}(\mathcal{O}_{A,C}) = n$. Equation (13.16) is Ackermann's formula for the observer gain L . Ackermann's formula is only valid for single output systems. For multi-output systems there are more advanced methods that are implemented in the Matlab `place` command. The observer gain L that places the eigenvalues of $A - LC$ at locations q_1, q_2, \dots, q_n can be found using the matlab command

```

1  if rank(ovs(A,C))<=n,
2      disp('System Not Observable');
3  else % if so, compute observer gains
4      L = place(A',C',[q1,q2,...,qn])';
5  end

```

where Line 1 checks to see if the system is observable by checking whether $\text{rank}(\mathcal{O}_{A,C}) = n$. The gain L is found in Line 4, where it is important to note the three transposes which correspond to the transposes in Equation (13.14).

If an observer is used in the control law, as discussed in Chapter 12, then the complete observer based controller is shown in Figure 13.3. Note that integral feedback uses the reference output y_r , whereas the observer uses the measured output y_m . There are many systems for which the measured output is identical to the reference output.

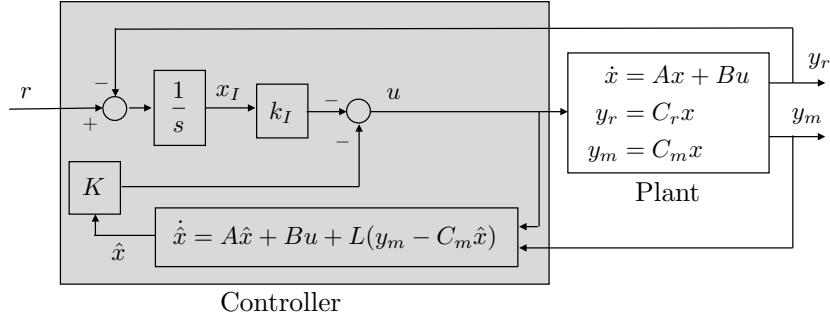


Figure 13.3: Closed loop system with observer based feedback, including integral feedback.

13.1.3 Separation Principle

The observer based feedback design consists of two steps:

- Find the state feedback gains K to place the eigenvalues of $A - BK$ at specified locations p_1, \dots, p_n ,
- Find the observer gains L to place the eigenvalues of $A - LC$ at specified locations q_1, \dots, q_n .

Since the closed loop system includes both the observer and the state feedback, it is important to analyze whether the closed loop system is stable, and to determine the closed loop poles. If the integrator is excluded in Figure 13.3 then the closed loop system has $2n$ states: n states associated with the open loop plant x , and n states associated with the observer \hat{x} . Since the reference input doesn't affect the internal stability of the systems, setting $r = 0$ the equations of motion for the closed loop system can be written as

$$\begin{aligned}\dot{x} &= Ax + Bu \\ \dot{\hat{x}} &= A\hat{x} + Bu + L(y - C_m\hat{x}) \\ u &= -K\hat{x} \\ y &= C_m x,\end{aligned}$$

which implies that

$$\begin{pmatrix} \dot{x} \\ \dot{\hat{x}} \end{pmatrix} = \begin{pmatrix} A & -BK \\ LC_m & A - BK - LC_m \end{pmatrix} \begin{pmatrix} x \\ \hat{x} \end{pmatrix}.$$

Recalling that a change of variables does not change the eigenvalues of the system, and defining the observation error as $e = x - \hat{x}$, and noting that

$$\begin{pmatrix} x \\ e \end{pmatrix} = \begin{pmatrix} I & \mathbf{0} \\ I & -I \end{pmatrix} \begin{pmatrix} x \\ \hat{x} \end{pmatrix},$$

we have that

$$\begin{aligned} \begin{pmatrix} \dot{x} \\ \dot{e} \end{pmatrix} &= \begin{pmatrix} I & \mathbf{0} \\ I & -I \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{\hat{x}} \end{pmatrix} \\ &= \begin{pmatrix} I & \mathbf{0} \\ I & -I \end{pmatrix} \begin{pmatrix} A & -BK \\ LC_m & A - BK - LC_m \end{pmatrix} \begin{pmatrix} x \\ \hat{x} \end{pmatrix} \\ &= \begin{pmatrix} I & \mathbf{0} \\ I & -I \end{pmatrix} \begin{pmatrix} A & -BK \\ LC_m & A - BK - LC_m \end{pmatrix} \begin{pmatrix} I & \mathbf{0} \\ I & -I \end{pmatrix}^{-1} \begin{pmatrix} x \\ e \end{pmatrix}, \end{aligned}$$

and noting that

$$\begin{pmatrix} I & \mathbf{0} \\ I & -I \end{pmatrix}^{-1} = \begin{pmatrix} I & \mathbf{0} \\ I & -I \end{pmatrix}$$

gives

$$\begin{pmatrix} \dot{x} \\ \dot{e} \end{pmatrix} = \begin{pmatrix} A - BK & BK \\ \mathbf{0} & A - LC_m \end{pmatrix} \begin{pmatrix} x \\ e \end{pmatrix}.$$

Using the determinant properties for block diagonal matrices discussed in Appendix f, it is straight forward to show that

$$\text{eig} \begin{pmatrix} A - BK & BK \\ \mathbf{0} & A - LC_m \end{pmatrix} = \text{eig}(A - BK) \cup \text{eig}(A - LC_m).$$

Therefore the poles of the closed loop system are precisely the poles selected during the design of the feedback gains K in addition to the poles selected during the design of the observer gains L . This happy circumstance is called the *separation principle*, which in essence implies that the controller and the observer can be designed separately and then combined without affecting the stability of the system. Unfortunately the separation principle does not hold in general for nonlinear systems, and even more unfortunately, while the separation principle guarantees that stability is guaranteed, it turns out that the addition of an observer, can negatively impact the robustness of a state feedback design.

13.1.4 Observer Design for Linearized Systems

Suppose that the state space model is from a linearized system, i.e, suppose that $x \in \mathbb{R}^n$ is the state and that $x_e \in \mathbb{R}^n$ is the equilibrium state, and that $\tilde{x} = x - x_e$ is the state linearized around the equilibrium. Similarly, suppose that $u \in \mathbb{R}^m$ is the input, that $u_e \in \mathbb{R}^m$ is the input at equilibrium, and that $\tilde{u} = u - u_e$ is the input linearized around equilibrium, and that $y_m \in \mathbb{R}^p$ is the measured output, $y_{me} = C_m x_e \in \mathbb{R}^p$ is the measured output at equilibrium,

and $\tilde{y}_m = y_m - y_{me}$ is the measured output linearized about equilibrium, then the linearized state space model is given by

$$\begin{aligned}\dot{\tilde{x}} &= A\tilde{x} + B\tilde{u} \\ \tilde{y}_m &= C_m\tilde{x}.\end{aligned}$$

The observer design problem then finds an observer gain so that the eigenvalues of $A - LC_m$ are at specified locations, and the resulting observer is

$$\dot{\hat{x}} = A\hat{x} + B\hat{u} + L(\tilde{y}_m - C_m\hat{x}),$$

where $\hat{x} = \hat{x} - x_e$. Writing the observer in terms of \hat{x} gives

$$\dot{\hat{x}} - \dot{x}_e = A(\hat{x} - x_e) + B(u - u_e) + L((y_m - y_{me}) - C_m(\hat{x} - x_e)).$$

Noting that $\dot{x}_e = 0$ and that $y_{me} = C_m x_e$ gives

$$\dot{\hat{x}} = A(\hat{x} - x_e) + B(u - u_e) + L(y_m - C_m\hat{x}).$$

13.2 Summary of Design Process - Observer Design

The procedure for designing an observer is summarized as follows.

Given the plant

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y_m &= C_m x\end{aligned}$$

find the observer gain L so that the state observer

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y_m - C_m\hat{x})$$

has stable estimation error dynamics with poles located at q_1, q_2, \dots, q_n .

Step 1. Check to see if the system is observable by computing the observability matrix

$$\mathcal{O}_{A,C_m} = \begin{pmatrix} C_m \\ C_mA \\ C_mA^2 \\ \vdots \\ C_mA^{n-1} \end{pmatrix}$$

and checking to see if $\text{rank}(\mathcal{O}_{A,C_m}) = n$.

Step 2. Find the open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A) = s^n + a_{n-1}s^{n-1} + \cdots + a_1s + a_0,$$

and construct the row vector

$$\mathbf{a}_A = (a_{n-1}, \ a_{n-2}, \ \dots, \ a_1, \ a_0)$$

and the matrix

$$\mathcal{A}_A = \begin{pmatrix} 1 & a_{n-1} & a_{n-2} & \cdots & a_2 & a_1 \\ 0 & 1 & a_{n-1} & \cdots & a_3 & a_2 \\ \dots & & \ddots & & & \vdots \\ 0 & 0 & \cdots & 0 & 1 & a_{n-1} \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$$

Step 3. Find the desired characteristic polynomial for the observation error

$$\Delta_{obs}^d(s) = (s - q_1)(s - q_2) \cdots (s - q_n) = s^2 + \beta_{n-1}s^{n-1} + \cdots + \beta_1s + \beta_0,$$

and construct the row vector

$$\boldsymbol{\beta} = (\beta_{n-1}, \ \beta_{n-2}, \ \dots, \ \beta_1, \ \beta_0).$$

Step 4. Compute the desired observer gains as

$$L = \mathcal{O}_{A,C_m}^{-1}(\mathcal{A}_A^\top)^{-1}(\boldsymbol{\beta} - \mathbf{a}_A^\top)$$

If Matlab is available, these steps can be implemented using the following script.

```

1 % is the system observable?
2 if rank(obsv(A,C))<n,
3     disp('System Not Observable');
4 else % if so, compute observer gains
5     L = place(A',C',[q1,q2,...,qn])';
6 end

```

13.2.1 Simple example

Given the system

$$\begin{aligned} \dot{x} &= \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}x + \begin{pmatrix} 0 \\ 4 \end{pmatrix}u \\ y &= \begin{pmatrix} 5 & 0 \end{pmatrix}x \end{aligned}$$

find the observer gain L so that the state observer

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x})$$

has stable estimation error dynamics with poles located at $-10 \pm j10$.

Step 1. The observability matrix is given by

$$\mathcal{O} = \begin{pmatrix} C \\ CA \end{pmatrix} = \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}.$$

The determinant is $\det(\mathcal{O}) = 25 \neq 0$, therefore the system is observable.

Step 2. The open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A) = \det \begin{pmatrix} s & -1 \\ -2 & s-3 \end{pmatrix} = s^2 - 3s - 2,$$

which implies that

$$\mathbf{a} = (-3, -2)$$

$$\mathcal{A} = \begin{pmatrix} 1 & -3 \\ 0 & 1 \end{pmatrix}$$

Step 3. The desired characteristic polynomical for the observation error is

$$\Delta_{obs}^d(s) = (s + 10 - j10)(s + 10 + j10) = s^2 + 20s + 200,$$

which implies that

$$\boldsymbol{\beta} = (20, 200).$$

Step 4. The observation gains are therefore given as

$$\begin{aligned} L &= \mathcal{O}^{-1}(\mathcal{A}^\top)^{-1}(\boldsymbol{\beta} - \mathbf{a})^\top \\ &= \begin{pmatrix} \frac{1}{5} & 0 \\ 0 & \frac{1}{5} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix} ((20, 200) - (-3, -2))^\top \\ &= \begin{pmatrix} 1/5 & 0 \\ 3/5 & 1/5 \end{pmatrix} \begin{pmatrix} 23 \\ 202 \end{pmatrix} \\ &= \begin{pmatrix} 23/5 \\ 271/5 \end{pmatrix} \end{aligned}$$

Alternatively, we could have used the following Matlab script

```

1 % original state space system
2 A = [0, 1; 2, 3];
3 B = [0; 4];
4 C = [5 0];
5
6 q1 = -10+j*10;
7 q2 = -10-j*10;
8
9 % is the system observable?
10 if rank(obsv(A,C))<2,
11     disp('System Not Observable');
12 else % if so, compute observer gains
13     L = place(A',C',[q1,q2])';
14 end

```

The corresponding observer is given by

$$\dot{\hat{x}} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \hat{x} + \begin{pmatrix} 0 \\ 4 \end{pmatrix} u + \begin{pmatrix} 23/5 \\ 271/5 \end{pmatrix} (y - (5 \ 0) \hat{x}), \quad (13.17)$$

and can be implemented using the following Matlab code.

```

1      persistent xhat          % estimated state (for observer)
2      persistent u           % delayed input   (for observer)
3      if t<Ts,
4          xhat = [0; 0];
5          u    = 0;
6      end
7      N = 10;
8      for i=1:N,
9          xhat = xhat + Ts/N*( A*xhat + B*u + L*(y-C*xhat) );
10     end

```

Digital implementation of the observer in Equation (13.17) requires two persistent variables, one for \hat{x} and one for u . Without additional information, the observer will always be initialized at zero, as shown in Line 4. As shown in Line 5, u is also initialized to zero. Since the state feedback control will be based on \hat{x} , and since \hat{x} depends on u as in Line 9, one of the quantities must be computed first. The **for**-loop in Lines 8-10, approximates the differential equation

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x})$$

by approximating

$$\dot{\hat{x}}(t) \approx \frac{\hat{x}(t) - \hat{x}(t-T)}{T}$$

to get

$$\hat{x}(t) = \hat{x}(t-T) + T(A\hat{x}(t-T) + Bu(t-T) + L(y(t) - C\hat{x}(t-T))).$$

To improve the approximation, T should be made as small as possible. The for-loop in Lines 8-10 makes T , $N = 10$ times smaller than the sample period T_s .

13.3 Design Study A. Single Link Robot Arm

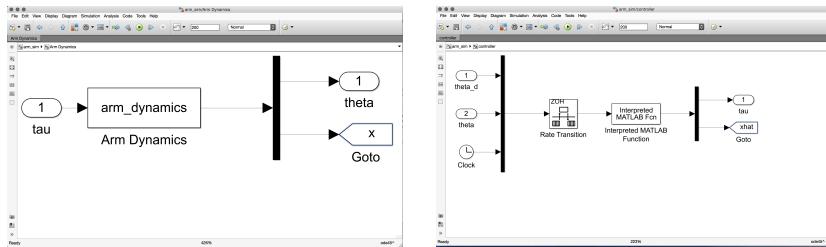


Homework Problem A.13

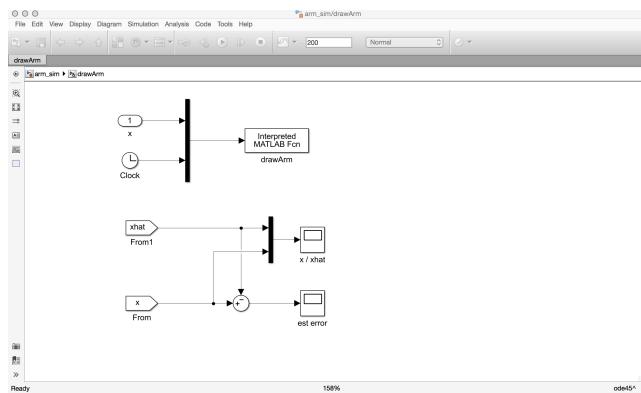
The objective of this problem is to design an observer that estimates the state of the system and uses the estimated state in the controller designed in Homework A.12.

CHAPTER 13. OBSERVERS

- (a) The first step is to modify the Simulink diagram so that we can observe the performance of the observer. Modify the Simulink diagram from Homework A.112 so that the output of the s-function defining the dynamics is both y and x as shown in Figure 13.3. In addition, modify the Simulink diagram so that the controller outputs both u and \hat{x} , as also shown in Figure 13.3.



The animation block can also be modified to include plots of x and \hat{x} and also the estimation error $x - \hat{x}$, as shown in Figure 13.3.



The s-function defining the system will need to be modified so that the output of the block is both the normal output, as well as the actual state:

```

1  sizes.NumOutputs      = 1+2; % in mdlInitializeSizes
2
3  function sys=mdlOutputs(t,x,u,P)
4      theta      = x(1);
5      thetadot  = x(2);
6      tau        = u(1);
7      sys = [theta; x];

```

Similarly, the output of the control block will need to be modified so that the output of the function is both the control output u as well as the state estimate \hat{x} :

```

1 function out=arm_ctrl(in,P)
2 % control code goes here
3 out = [u; xhat];
4 end

```

- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, without an input disturbance. Modify `arm_dynamics.m` so that the parameters known to the controller are the actual plant parameters (uncertainty parameter $\alpha = 0$).
- (c) Verify that the state space system is observable by checking that $\text{rank}(\mathcal{O}_{A,C}) = n$.
- (d) In the control block, add an observer to estimate the state \hat{x} , and use the estimate of the state in your feedback controller. Tune the poles of the controller and observer to obtain good performance.
- (e) As motivation for the next chapter, add an input disturbance to the system of 0.01 and observe that there is steady state error in the response even though there is an integrator. This is caused by a steady state error in the observation error. In the next chapter we will show how to remove the steady state error in the observation error.

Solution

Matlab code used to design the observer based controller is shown below:

```

1 % sample rate
2 P.Ts = 0.01;
3
4 % equilibrium torque
5 P.theta_e = 0*pi/180;
6 P.tau_e = P.m*P.g*P.ell/2*cos(P.theta_e);
7
8 %
9 % state space design
10 P.A = [...
11     0, 1;...
12     0, -3*P.b/P.m/(P.ell^2);...
13 ];
14 P.B = [0; 3/P.m/(P.ell^2) ];
15 P.C = [...
16     1, 0;...
17 ];
18
19 % form augmented system
20 A1 = [P.A, zeros(2,1); -P.C, 0];
21 B1 = [P.B; 0];
22
23 % tuning parameters for control
24 wn = 5.5;

```

CHAPTER 13. OBSERVERS

```

25 zeta = 0.707;
26 integrator_pole = -1;
27
28 % compute gains
29 des_char_poly = conv([1,2*zeta*wn,wn^2],poly(integrator_pole));
30 des_poles = roots(des_char_poly);
31 if rank(ctrb(A1,B1))≠3,
32     disp('System Not Controllable');
33 else % if so, compute gains
34     K1 = place(A1,B1,des_poles);
35     P.K = K1(1:2);
36     P.ki = K1(3);
37 end
38
39 % observer design
40 % tuning parameters for observer
41 wn_obs = 10;
42 zeta_obs = 0.707;
43
44 % desired observer poles
45 des_observ_char_poly = [1,2*zeta*wn,wn^2];
46 des_observ_poles = roots(des_observ_char_poly);
47
48 % is the system observable?
49 if rank(observ(P.A,P.C))≠2,
50     disp('System Not Observable');
51 else % if so, compute gains
52     P.L = place(P.A',P.C',des_observ_poles)';
53 end

```

Lines 1–53 are identical to the state feedback design from Homework A.12 except that we are using ω_n as a tuning parameter instead of t_r . This is to highlight the bandwidth separation between the controller and the observer. The poles of the observation error are selected in Lines 55–69 in a form that allows $\omega_{n_{obs}}$ to be tuned for performance. The observability check is in Line 64–66, and the observer gains are computed using the `place` command in Line 68. Note the transposes on this line.

Matlab code for the observer based control is shown below:

```

1 function out=arm_ctrl(in,P)
2     theta_r = in(1);
3     theta_m = in(2);
4     t      = in(3);
5
6     % implement observer
7     persistent xhat      % estimated state (for observer)
8     persistent tau       % delayed input (for observer)
9     if t<P.Ts,
10         xhat = [0; 0];
11         tau = 0;
12     end
13     N = 10;
14     % compute equilibrium torque at thetahat
15     tau_e = P.m*P.g*(P.ell/2)*cos(xhat(1));

```

```

16    for i=1:N,
17        xhat = xhat + ...
18            P.Ts/N*(P.A*xhat+P.B*(tau-tau_e)...
19                +P.L*(theta_m-P.C*xhat));
20    end
21    thetahat = xhat(1);
22
23    % implement integrator
24    error = theta_r - thetahat;
25    persistent integrator
26    persistent error_d1
27    % reset persistent variables at start of simulation
28    if t<P.Ts==1,
29        integrator = 0;
30        error_d1 = 0;
31    end
32    integrator = integrator + (P.Ts/2)*(error+error_d1);
33    error_d1 = error;
34
35    % compute control torque
36    tau_e = P.m*P.g*(P.ell/2)*cos(thetahat);
37    tau_unsat = tau_e - P.K*xhat - P.ki*integrator;
38    tau = sat( tau_unsat, P.tau_max );
39
40    % integrator anti-windup
41    if P.ki≠0,
42        integrator = integrator + P.Ts/P.ki*(tau-tau_unsat);
43    end
44
45    out = [tau; xhat];
46 end
47
48 function out = sat(in,limit)
49    if in > limit,      out = limit;
50    elseif in < -limit,  out = -limit;
51    else                 out = in;
52    end
53 end

```

The observer is implemented in Lines 6–21. Note that the control signal on line 37 uses \hat{x} instead of x , and that $\hat{\theta}$ is used instead of θ on lines 24 and 36. See <http://controlbook.byu.edu> for the complete solution.

13.4 Design Study B. Inverted Pendulum



Homework Problem B.13

The objective of this problem is to design an observer that estimates the state of the system and to use the estimated state in the controller designed in Homework B.12.

- (a) Modify the simulink diagram from HW B.12 as described in the description of HW A.13 to add a plot of the true state x , and estimated state \hat{x} , and the observation error $x - \hat{x}$.
- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, without an input disturbance. Modify `pendulum_dynamics.m` so that the parameters known to the controller are the actual plant parameters (uncertainty parameter $\alpha = 0$).
- (d) In the control block, add an observer to estimate the state \hat{x} , and use the estimate of the state in your feedback controller. Tune the poles of the controller and observer to obtain good performance.
- (e) As motivation for the next chapter, add an input disturbance to the system of 0.05 and observe that there is steady state error in the response even though there is an integrator. This is caused by a steady state error in the observation error. In the next chapter we will show how to remove the steady state error in the observation error.

Solution

Matlab code used to design the observer based controller is shown below:

```

1 % inverted Pendulum parameter file
2
3 % saturation limits
4 P.F_max = 5; % Max Force, N
5
6 %%%%%%%%%%%%%%%%
7 % observer based control with integrator
8 %%%%%%%%%%%%%%%%
9 % tuning parameters for control
10 tr_z = 1.5; % rise time for position
11 tr_theta = 0.5; % rise time for angle
12 zeta_z = 0.707; % damping ratio position
13 zeta_th = 0.707; % damping ratio angle
14 integrator_pole = -5; % integrator pole
15 % tuning parameters for observer
16 wn_th_obs = 5*wn_th;
17 wn_z_obs = 5*wn_z;
18
19 % state space equations
20 P.A = [...
21     0, 0, 1, 0;...
22     0, 0, 0, 1;...
23     0, -P.m1*P.g/P.m2, -P.b/P.m2, 0;...
24     0, (P.m1+P.m2)*P.g/P.m2/P.ell, P.b/P.m2/P.ell, 0;...
25 ];
26 P.B = [0; 0; 1/P.m2; -1/P.m2/P.ell ];
27 P.C = [...
28     1, 0, 0, 0;...
29     0, 1, 0, 0;...

```

```

30      ];
31  % form augmented system
32 Cout = [1,0,0,0];
33 A1 = [P.A, zeros(4,1); -Cout, 0];
34 B1 = [P.B; 0];
35
36 % compute control gains
37 wn_th = 2.2/tr_theta; % natural frequency for angle
38 wn_z = 2.2/tr_z; % natural frequency for position
39 des_char_poly = conv(conv([1,2*zeta_z*wn_z,wn_z^2],...
40                      [1,2*zeta_th*wn_th,wn_th^2]),...
41                      poly(integrator_pole));
42 des_poles = roots(des_char_poly);
43
44 % is the system controllable?
45 if rank ctrb(A1,B1)≠5
46     disp('System Not Controllable');
47 else % if so, compute gains
48     K1 = place(A1,B1,des_poles);
49     P.K = K1(1:4);
50     P.ki = K1(5);
51 end
52
53 % computer observer gains
54 des_observ_char_poly = conv([1,2*zeta_z*wn_z_obs,wn_z_obs^2],...
55                           [1,2*zeta_th*wn_th_obs,wn_th_obs^2]);
56 des_observ_poles = roots(des_observ_char_poly);
57
58 % is the system observable?
59 if rank obsv(P.A,P.C)≠4
60     disp('System Not Observable');
61 else % if so, compute gains
62     P.L = place(P.A', P.C', des_observ_poles)';
63 end
64
65 sprintf('K: [%f, %f, %f, %f]\nki: %f\nL: [%f, %f, %f, %f]^T',...
66     P.K(1), P.K(2), P.K(3), P.K(4), P.ki, P.L(1), P.L(2), P.L(3), P.L(4))

```

Matlab code for the observer based control is shown below:

```

1 function out=pendulum_ctrl(in,P)
2     z_r      = in(1);
3     z_m      = in(2);
4     theta_m = in(3);
5     t       = in(4);
6
7     % implement observer
8     persistent xhat      % estimated state (for observer)
9     persistent F
10    if t<P.Ts,
11        xhat = [-0.1;0.0;0;0];
12        F   = 0;
13    end
14    N = 10;
15    for i=1:N,
16        xhat = xhat + ...

```

```

17         P.Ts/N*(P.A*xhat+P.B*F...
18             +P.L*([z_m;theta_m]-P.C*xhat));
19     end
20     zhat = xhat(1);
21
22     % integrator
23     error = z_r - zhat;
24     persistent integrator
25     persistent error_d1
26     % reset persistent variables at start of simulation
27     if t<P.Ts==1,
28         integrator = 0;
29         error_d1 = 0;
30     end
31     integrator = integrator + (P.Ts/2)*(error+error_d1);
32     error_d1 = error;
33
34     % compute the state feedback controller
35     F_unsat = -P.K*xhat - P.ki*integrator;
36     F = sat(F_unsat, P.F_max);
37
38     % integrator anti-windup
39     if P.ki≠0,
40         integrator = integrator + P.Ts/P.ki*(F-F_unsat);
41     end
42
43     out = [F; xhat];
44
45 end
46
47 %-----%
48 % saturation function
49 function out = sat(in,limit)
50     if in > limit,      out = limit;
51     elseif in < -limit,   out = -limit;
52     else                  out = in;
53     end
54 end

```

See <http://controlbook.byu.edu> for the complete solution.

13.5 Design Study C. Satellite Attitude Control



Homework Problem C.13

The objective of this problem is to design an observer that estimates the state of the system and to use the estimated state in the controller designed in Homework C.12.

- (a) Modify the simulink diagram from HW C.12 as described in the description of HW A.13 to add a plot of the true state x , and estimated state \hat{x} , and

the observation error $x - \hat{x}$.

- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, without an input disturbance. Modify `satellite_dynamics.m` so that the parameters known to the controller are the actual plant parameters (uncertainty parameter $\alpha = 0$).
- (d) In the control block, add an observer to estimate the state \hat{x} , and use the estimate of the state in your feedback controller. Tune the poles of the controller and observer to obtain good performance.
- (e) As motivation for the next chapter, add an input disturbance to the system of 1 and observe that there is steady state error in the observation error. In the next chapter we will show how to remove the steady state error in the observation error.

Solution

Matlab code used to design the observer based controller is shown below:

```

1 % satellite parameter file
2
3
4 %%%%%%%%%%%%%%%%
5 % state feedback control with integrator
6 %%%%%%%%%%%%%%%%
7 % tuning parameters
8 wn_th = 0.6;
9 zeta_th = 0.707;
10 wn_phi = 1.1;
11 zeta_phi = 0.707;
12 integrator_pole = -1;
13 % pick observer poles
14 wn_th_obs = 10*wn_th;
15 wn_phi_obs = 10*wn_phi;
16
17 % state space design
18 P.A = [
19     0, 0, 1, 0; ...
20     0, 0, 0, 1; ...
21     -P.k/P.Js, P.k/P.Js, -P.b/P.Js, P.b/P.Js; ...
22     P.k/P.Jp, -P.k/P.Jp, P.b/P.Jp, -P.b/P.Jp; ...
23 ];
24 ];
25 P.B = [0; 0; 1/P.Js; 0];
26 P.C = [
27     1, 0, 0, 0; ...
28     0, 1, 0, 0; ...
29 ];
30
31 % form augmented system
32 Cout = [0,1,0,0];
33 A1 = [P.A, zeros(4,1); -Cout, 0];

```

```

34 B1 = [P.B; 0];
35
36 % compute gains
37 ol_char_poly = charpoly(P.A);
38 des_char_poly = conv(conv([1,2*zeta_th*wn_th,wn_th^2],...
39 [1,2*zeta_phi*wn_phi,wn_phi^2]),...
40 poly(integrator_pole));
41 des_poles = roots(des_char_poly);
42
43 % is the system controllable?
44 if rank(ctrb(A1,B1))≠5,
45     disp('System Not Controllable');
46 else % if so, compute gains
47     K1 = place(A1,B1,des_poles);
48     P.K = K1(1:4);
49     P.ki = K1(5);
50 end
51
52 % observer design
53 des_observ_char_poly = conv(...%
54 [1,2*zeta_phi*wn_phi_obs,wn_phi_obs^2],...
55 [1,2*zeta_th*wn_th_obs,wn_th_obs^2]);
56 des_observ_poles = roots(des_observ_char_poly);
57
58 % is the system observable?
59 if rank(obsv(P.A,P.C))≠4,
60     disp('System Not Observable');
61 else % if so, compute gains
62     P.L = place(P.A', P.C', des_observ_poles)';
63 end
64
65
66 sprintf('K: [%f, %f, %f, %f]\nki: %f\nL: [%f, %f, %f, %f]^T',...
67 P.K(1), P.K(2), P.K(3), P.K(4), P.ki, P.L(1), P.L(2), P.L(3), P.L(4))

```

Matlab code for the observer based control is shown below:

```

1 function out=satellite_ctrl(in, P)
2     phi_r      = in(1);
3     theta_m    = in(2);
4     phi_m      = in(3);
5     t          = in(4);
6
7     % implement observer
8     persistent xhat      % estimated state (for observer)
9     persistent tau
10    if t<P.Ts
11        xhat = [0;0;0;0];
12        tau = 0;
13    end
14    N = 10;
15    for i=1:N
16        xhat = xhat + ...
17            P.Ts/N*(P.A*xhat+P.B*tau...
18            +P.L*([theta_m;phi_m]-P.C*xhat));
19    end

```

```

20    phihat = xhat(2);
21
22    % integrator
23    error = phi_r - phihat;
24    persistent integrator
25    persistent error_d1
26    % reset persistent variables at start of simulation
27    if t<P.Ts==1
28        integrator = 0;
29        error_d1 = 0;
30    end
31    integrator = integrator + (P.Ts/2)*(error+error_d1);
32    error_d1 = error;
33
34    % compute the state feedback controller
35    tau_unsat = -P.K*xhat - P.ki*integrator;
36    tau = sat( tau_unsat, P.taumax );
37
38    % integrator anti-windup
39    if P.ki≠0
40        integrator = integrator + P.Ts/P.ki*(tau-tau_unsat);
41    end
42
43    out = [tau; xhat];
44
45 end
46
47 %
48 % saturation function
49 function out = sat(in,limit)
50     if in > limit,         out = limit;
51     elseif in < -limit,    out = -limit;
52     else                  out = in;
53 end
54 end

```

See <http://controlbook.byu.edu> for the complete solution.

Notes and References

The original source for Luenberger observers is in [21]. Observer design using pole placement is covered in most introductory textbooks on control, see for example [4, 22, 23], as well as introductions to state space design [5, 6]. The most common definition of observability is that a dynamic system is said to be *observable* if the initial state x_0 can be reconstructed by observing the output over any finite interval. Linear time-invariant systems are observable if and only if the observability matrix is full rank [5]. Observer theory and design is an interesting topic in its own right. The design of nonlinear observers is currently an active area of research. The most commonly used observer for nonlinear systems is the extended Kalman filter [24, 25].

Chapter 14

Disturbance Observers

14.1 Theory

In previous sections we have discussed that modeling errors and other real-world effects result in an input disturbance to the system as shown in Figure 14.1. The

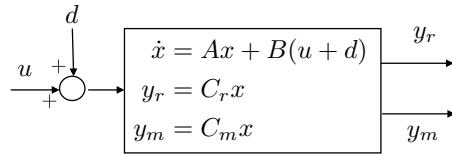


Figure 14.1: Input disturbance to plant modeled by state space equations.

objective of this section is explain the effect of the disturbance on an observer. It turns out that the disturbance creates a steady state bias in the observer states. We will also describe how the observer can be augmented, the dual of adding an integrator, to estimate the disturbance and to remove the steady state observation error.

To understand the effect of the disturbance on the observer, note that when the disturbance is present, the state space equations for the system are

$$\begin{aligned}\dot{x} &= Ax + B(u + d) \\ y_m &= C_m x.\end{aligned}$$

The standard equation for the observer is

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y_m - C_m \hat{x}). \quad (14.1)$$

Defining the observation error as $e = x - \hat{x}$ and taking the derivative with

respect to time gives

$$\dot{e} = \dot{x} - \hat{x} \quad (14.2)$$

$$\begin{aligned} &= Ax + Bu + Bd - A\hat{x} - Bu - LC_m x + LC_m \hat{x} \\ &= (A - LC_m)e + Bd. \end{aligned} \quad (14.3)$$

Taking the Laplace transform of both sides, and including the initial condition, and solving for $E(s)$ gives

$$E(s) = (sI - (A - LC_m))^{-1}e(0) + (sI - (A - LC_m))^{-1}B\mathcal{L}\{d(t)\}.$$

Therefore, the error is due to two terms. The first term involves the initial conditions $e(0)$ and decays to zero if the eigenvalues of $A - LC_m$ are in the open left half of the complex plane. The second term is driven by the disturbance and for a constant bounded disturbance, will result in a constant bounded observation error.

If the disturbance $d(t)$ were perfectly known, then we could modify the observer in Equation (14.1) as

$$\dot{\hat{x}} = A\hat{x} + B(u + d) + L(y_m - C_m \hat{x}), \quad (14.4)$$

so that the evolution of the observation error become

$$\begin{aligned} \dot{e} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + B(u + d) - A\hat{x} - B(u + d) - LC_m x + LC_m \hat{x} \\ &= (A - LC_m)e, \end{aligned}$$

which, in contrast to Equation (14.3) is not driven by the disturbance d . However, since d is not known it needs to be estimated. The basic idea in this chapter is to estimate d using an augmented observer.

If we assume that d is constant, then the differential equation that governs the evolution of d is given by $\dot{d} = 0$. Therefore, if we augment the states x of the system with the disturbance, then the equations of motion can be written as

$$\begin{aligned} \begin{pmatrix} \dot{x} \\ \dot{d} \end{pmatrix} &= \begin{pmatrix} A & B \\ \mathbf{0} & 0 \end{pmatrix} \begin{pmatrix} x \\ d \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u \\ y &= \begin{pmatrix} C_m & 0 \end{pmatrix} \begin{pmatrix} x \\ d \end{pmatrix}. \end{aligned}$$

Define the augmented state space matrices as

$$\begin{aligned} A_2 &= \begin{pmatrix} A & B \\ \mathbf{0} & 0 \end{pmatrix} \\ C_2 &= \begin{pmatrix} C_m & 0 \end{pmatrix}. \end{aligned}$$

CHAPTER 14. DISTURBANCE OBSERVERS

If (A_2, C_2) are observable, then using the techniques discussed in Chapter 13 we can design an observer for the augmented system, where the augmented observer equations are

$$\begin{pmatrix} \dot{\hat{x}} \\ \dot{\hat{d}} \end{pmatrix} = \begin{pmatrix} A & B \\ \mathbf{0} & 0 \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{d} \end{pmatrix} + \begin{pmatrix} L \\ L_d \end{pmatrix} \left(y_m - (C_m \quad 0) \begin{pmatrix} \hat{x} \\ \hat{d} \end{pmatrix} \right)$$

or in component form

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + B(u + \hat{d}) + L(y_m - C_m\hat{x}) \\ \dot{\hat{d}} &= L_d(y_m - C_m\hat{x}), \end{aligned}$$

where the gain $L_2 = (L^\top, L_d)^\top$ is obtained using the Ackerman formula

$$L_2 = \mathcal{O}_{A_2, C_2}^{-1}(\mathcal{A}_{A_2}^\top)^{-1}(\boldsymbol{\beta}_d - \mathbf{a}_{A_2})^\top,$$

where \mathbf{a}_{A_2} are obtained from the characteristic polynomial of A_2 and $\boldsymbol{\beta}_d$ are obtained from the desired characteristic polynomial for $A_2 - L_2 C_2$. Alternatively, L_2 can be obtained from the Matlab place command as

```
1 L2 = place(A2', C2', desired_observer_poles)'
```

14.1.1 Simple Example

Consider the example in Section 11.2.1, 12.2.1, and 13.2.1. A block diagram for the system is shown in Figure 14.2, where a constant input disturbance has been added to the system.

CHAPTER 14. DISTURBANCE OBSERVERS

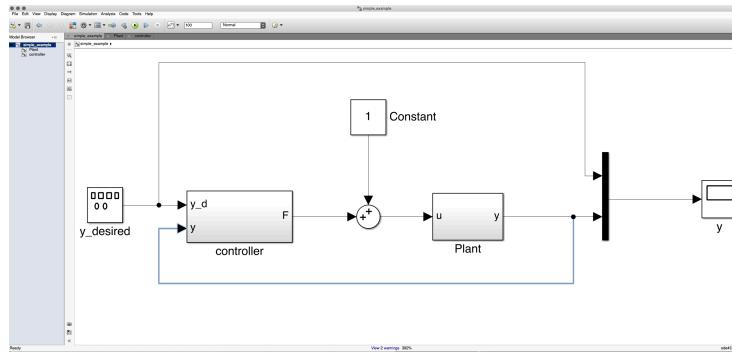


Figure 14.2: Block diagram for simple example, where a constant input disturbance has been added to the system.

Suppose that a standard feedback controller without an integrator and without a disturbance observer is designed for this system. The associated Matlab code is

```

1 P.A = [0, 1; 2, 3];
2 P.B = [0; 4];
3 P.C = [5, 0];
4 P.D = 0;
5
6 % pick poles of controller
7 wn_ctrl = 1;
8 zeta_ctrl = 0.707;
9 charpoly = [1,2*zeta_ctrl*wn_ctrl,wn_ctrl^2];
10 des_ctrl_poles = roots(charpoly);
11
12 % pick poles of observer
13 wn_observ = 10;
14 zeta_observ = 0.707;
15 charpoly = [1,2*zeta_observ*wn_observ,wn_observ^2];
16 des_observ_poles = roots(charpoly);
17
18 % is the system controllable?
19 if rank(ctrb(P.A,P.B))≠2,
20     disp('System Not Controllable');
21 else
22     P.K = place(P.A,P.B,des_ctrl_poles);
23     P.kr = -1/(P.C*inv(P.A-P.B*P.K)*P.B);
24 end

```

CHAPTER 14. DISTURBANCE OBSERVERS

```

25
26 % is the system observable?
27 if rank(obsv(P.A,P.C))≠2,
28     disp('System Not Observable');
29 else % if so, compute gains
30     P.L = place(P.A',P.C',des_obsvo_poles)';
31 end

```

The associated control code is

```

1 % define and initialize persistent variables
2 persistent xhat          % estimated state (for observer)
3 persistent u              % delayed input (for observer)
4
5 N = 10; % number of integration steps for each sample
6
7 % initialize persistent variables
8 if t==0,
9     xhat = zeros(2,1);
10    u = 0;
11 end
12 % solve observer differential equations
13 for i=1:N,
14     xhat = xhat + P.Ts/N*(P.A*xhat + P.B*u + P.L*(y-P.C*xhat));
15 end
16 % observer based feedback controller
17 u = P.kr*r - P.K*xhat;

```

The step response for this controller is shown in Figure 14.3, where it is obvious that the input disturbance is causing a large steady state error in the system response, as well as in the estimation error.

CHAPTER 14. DISTURBANCE OBSERVERS

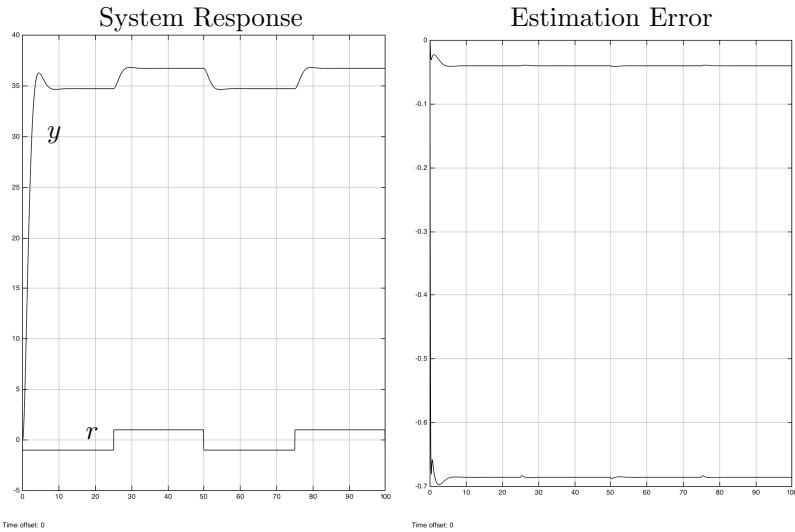


Figure 14.3: Step response for standard observer based control without integrator and without disturbance observer.

If an integrator is added to the system using the following design file

```

1  integrator_pole = -1.0; % pole for integrator
2  % augment system to add integrator
3  A1 = [P.A, zeros(2,1); -P.C, 0];
4  B1 = [P.B; 0];
5  % is the system controllable?
6  if rank ctrb(A1,B1) ≠ 3,
7      disp('System Not Controllable');
8  else
9      K1 = place(A1,B1,[des_ctrl_poles,integrator_pole]);
10     P.K = K1(1:2);
11     P.ki = K1(3);
12 end

```

and the control code listed below:

```

1  persistent integrator
2  persistent error_d1
3  persistent xhat % estimated state (for observer)
4  persistent u % delayed input (for observer)
5
6  % initialize persistent variables
7  if t==0,
8      xhat = zeros(2,1);
9      u = 0;
10     integrator = 0;
11     error_d1 = 0;
12 end

```

CHAPTER 14. DISTURBANCE OBSERVERS

```

13 % solve observer differential equations
14 for i=1:N,
15     xhat = xhat + P.Ts/N*(P.A*xhat + P.B*u + P.L*(y-P.C*xhat));
16 end
17 % implement integrator
18 error = r-y;
19 integrator = integrator + (P.Ts/2)*(error+error_d1);
20 error_d1 = error;
21 % observer based feedback controller
22 u = - P.K*xhat - P.ki*integrator;

```

The response of the system with the integrator is shown in Figure 14.4, where it is clear that the steady state error in the response has been removed, but where there is now a steady state response in the estimation error.

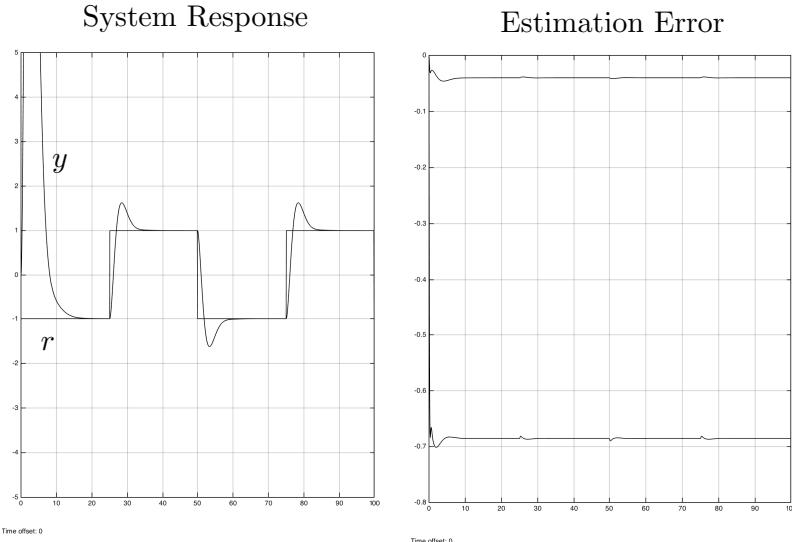


Figure 14.4: Step response for observer based control with integrator but without disturbance observer.

Now consider the case where the integrator is not present, but we add a disturbance observer. The design code is listed below.

```

1 % is the system controllable?
2 if rank(ctrb(P.A,P.B))<2,
3     disp('System Not Controllable');
4 else
5     P.K = place(P.A,P.B,des_ctrl_poles);
6     P.kr = -1/(P.C*inv(P.A-P.B*P.K)*P.B);
7 end
8
9 % augment system to disturbance observer
10 A2 = [P.A, P.B; zeros(1,2), 0];

```

CHAPTER 14. DISTURBANCE OBSERVERS

```

11 C2 = [P.C, 0];
12 % is the system observable?
13 if rank(obsv(A2,C2))<3,
14     disp('System Not Observable');
15 else % if so, compute gains
16     L2 = place(A2',C2',[des_obsrv_poles;dis_obsrv_pole]);
17     P.L = L2(1:2);
18     P.Ld = L2(3);
19 end

```

The associated control code is listed below.

```

1 persistent xhat          % estimated state (for observer)
2 persistent dhat          % disturbance estimate
3 persistent u              % delayed input (for observer)
4 % initialize persistent variables
5 if t==0,
6     xhat = zeros(2,1);
7     dhat = 0;
8     u = 0;
9 end
10 % solve observer differential equations
11 for i=1:N,
12     xhat = xhat + P.Ts/N*(P.A*xhat + P.B*(u+dhat) + P.L*(y-P.C*xhat));
13     dhat = dhat + P.Ts/N*P.Ld*(y-P.C*xhat);
14 end
15 % observer based feedback controller with disturbance estimate
16 u = P.kr*r - P.K*xhat - dhat;

```

The disturbance estimator is on Line 13. Note the presence of \hat{d} on Line 12. The disturbance estimate is subtracted from the control input on Line 16. The resulting system response is shown in Figure 14.5

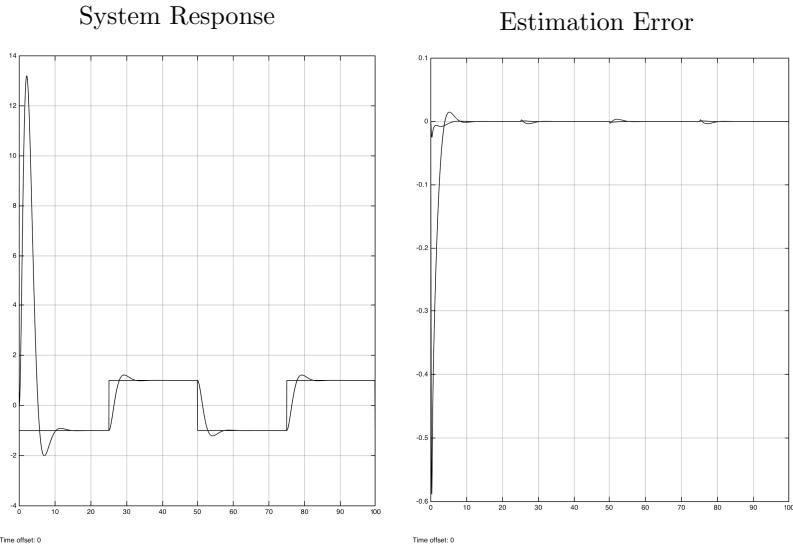


Figure 14.5: Step response for observer based control without integrator but disturbance observer.

Note that in this case, the steady state error in both the system response and the estimation error has been removed. At this point, the integrator can be added back into the system to remove the effect of model mismatch and other disturbances on the system.

14.2 Design Study A. Single Link Robot Arm



Homework Problem A.14

- (a) Modify your solution from HW A.13 so that the uncertainty parameter in `arm_dynamics.m` is $\alpha = 0.2$, representing 20% inaccuracy in the knowledge of the system parameters, and so that the input disturbance is 0.5 Newton-meters. Also, add noise to the output channel θ_m with standard deviation of 0.001. Before adding the disturbance observer, run the simulation and note that the controller is not robust to the large input disturbance.
- (b) Add a disturbance observer to the controller, and verify that the steady state error in the estimator has been removed. Tune the system to get good response.

Solution

Matlab code used to design the observer based controller is shown below:

```

1 % sample rate
2 P.Ts = 0.01;
3
4 % equilibrium torque
5 P.theta_e = 0*pi/180;
6 P.tau_e = P.m*P.g*P.ell/2*cos(P.theta_e);
7
8 % saturation constraint
9 P.tau_max = 1;
10 tau_max = P.tau_max-P.tau_e;
11
12 %
13 % state space design
14 P.A = [...
15     0, 1;...
16     0, -3*P.b/P.m/(P.ell^2);...
17 ];
18 P.B = [0; 3/P.m/(P.ell^2) ];
19 P.C = [...
20     1, 0;...
21 ];
22 % form augmented system for integrator
23 A1 = [P.A, zeros(2,1); -P.C, 0];
24 B1 = [P.B; 0];
25
26 % tuning parameters for control
27 tr = 0.2;
28 zeta = 0.707;
29 integrator_pole = -1.0;
30
31 % desired closed loop polynomial
32 wn = 2.2/tr;
33 des_char_poly = conv([1,2*zeta*wn,wn^2],poly(integrator_pole));
34 des_poles = roots(des_char_poly);
35
36 % is the system controllable?
37 if rank ctrb(A1,B1) ≠ 3,
38     disp('System Not Controllable');
39 else % if so, compute gains
40     K1 = place(A1,B1,des_poles);
41     P.K = K1(1:2);
42     P.ki = K1(3);
43 end
44
45 % observer design
46 % form augmented system for disturbance observer
47 A2 = [P.A, P.B; zeros(1,2), 0];
48 C2 = [P.C, 0];
49
50 % observer design
51 % tuning parameters for observer
52 wn_obs = 10;
53 zeta_obs = 1.707;
54 dist_pole = -5.5;
55
56 % desired observer poles
57 des_obsrv_char_poly = conv([1,2*zeta*wn,wn^2],poly(dist_pole));

```

CHAPTER 14. DISTURBANCE OBSERVERS

```

58 des_obsrv_poles = roots(des_obsrv_char_poly);
59
60 % is the system observable?
61 if rank(obsrv(A2,C2))≠3
62     disp('System Not Observable');
63 else % if so, compute gains
64     L2 = place(A2',C2',des_obsrv_poles)';
65     P.L = L2(1:2);
66     P.Ld = L2(3);
67 end

```

Matlab code for the observer based control is shown below:

```

1 function out=arm_ctrl(in,P)
2     theta_r = in(1);
3     theta_m = in(2);
4     t      = in(3);
5
6     % implement observer
7     persistent xhat      % estimated state (for observer)
8     persistent dhat      % estimate disturbance
9     persistent tau       % delayed input (for observer)
10    if t<P.Ts
11        xhat = [0; 0];
12        dhat = 0;
13        tau = 0;
14    end
15    % compute equilibrium torque tau_e
16    theta_e = 0;
17    theta = xhat(1);
18    tau_e = P.m*P.g*(P.ell/2)*cos(theta);
19    x_e = [theta_e; 0];
20    N = 10;
21    for i=1:N
22        xhat = xhat + ...
23            P.Ts/N*(P.A*(xhat-x_e)+P.B*(tau-tau_e+dhat)...
24            +P.L*(theta_m-P.C*xhat));
25        dhat = dhat + P.Ts/N*P.Ld*(theta_m-P.C*xhat);
26    end
27
28    % add integrator
29    error = theta_r - theta;
30    persistent integrator
31    persistent error_d1
32    % reset persistent variables at start of simulation
33    if t<P.Ts==1
34        integrator = 0;
35        error_d1 = 0;
36    end
37    integrator = integrator + (P.Ts/2)*(error+error_d1);
38    error_d1 = error;
39
40    % compute the state feedback controller
41    tau_unsat = tau_e-P.K*(xhat-x_e)-P.ki*integrator-dhat;
42    tau = sat( tau_unsat, P.tau_max );
43

```

```

44 % integrator anti-windup
45 if P.ki≠0
46     integrator = integrator + P.Ts/P.ki*(tau-tau_unsat);
47 end
48
49 out = [tau; xhat];
50
51 end
52
53 function out = sat(in,limit)
54     if in > limit,         out = limit;
55     elseif in < -limit,    out = -limit;
56     else                  out = in;
57 end
58 end

```

See <http://controlbook.byu.edu> for the complete solution.

14.3 Design Study B. Inverted Pendulum



Homework Problem B.14

- (a) Modify your solution from HW B.13 so that the uncertainty parameter in `pendulum_dynamics.m` is $\alpha = 0.2$, representing 20% inaccuracy in the knowledge of the system parameters, and so that the input disturbance is 0.5. Also, add noise to the output channels z_m and θ_m with standard deviation of 0.001. Before adding the disturbance observer, run the simulation and note that the controller is not robust to the large input disturbance.
- (b) Add a disturbance observer to the controller, and verify that the steady state error in the estimator has been removed. Tune the system to get good response.

Solution

Matlab code used to design the observer based controller is shown below:

```

1 % inverted Pendulum parameter file
2
3 % saturation limits
4 P.F_max = 5; % Max Force, N
5
6 %%%%%%%%%%%%%%%%
7 % observer based control with integrator
8 %%%%%%%%%%%%%%%%
9 % tuning parameters for control
10 tr_z = 1.5; % rise time for position
11 tr_theta = 0.5; % rise time for angle
12 zeta_z = 0.707; % damping ratio position
13 zeta_th = 0.707; % damping ratio angle

```

CHAPTER 14. DISTURBANCE OBSERVERS

```

14 integrator_pole = -10; % integrator pole
15 % tuning parameters for observer
16 tr_z_obs = tr_z/10;
17 tr_theta_obs = tr_theta/10;
18 dist_observ_pole = -1; % pole for disturbance observer
19
20
21 % state space equations
22 P.A = [...
23     0, 0, 1, 0;...
24     0, 0, 0, 1;...
25     0, -P.m1*P.g/P.m2, -P.b/P.m2, 0;...
26     0, (P.m1+P.m2)*P.g/P.m2/P.ell, P.b/P.m2/P.ell, 0;...
27 ];
28 P.B = [0; 0; 1/P.m2; -1/P.m2/P.ell ];
29 P.C = [...
30     1, 0, 0, 0;...
31     0, 1, 0, 0;...
32 ];
33 % form augmented system
34 Cout = [1,0,0,0];
35 A1 = [P.A, zeros(4,1); -Cout, 0];
36 B1 = [P.B; 0];
37
38 % compute control gains
39 wn_th = 2.2/tr_theta; % natural frequency for angle
40 wn_z = 2.2/tr_z; % natural frequency for position
41 des_char_poly = conv(conv([1,2*zeta_z*wn_z,wn_z^2],...
42 [1,2*zeta_th*wn_th,wn_th^2]),...
43 poly(integrator_pole));
44 des_poles = roots(des_char_poly);
45
46 % is the system controllable?
47 if rank(ctrb(A1,B1))≠5
48     disp('System Not Controllable');
49 else % if so, compute gains
50     K1 = place(A1,B1,des_poles);
51     P.K = K1(1:4);
52     P.ki = K1(5);
53 end
54
55 % computer observer gains
56
57 % observer design
58 % form augmented system for disturbance observer
59 A2 = [P.A, P.B; zeros(1,4), zeros(1,1)];
60 C2 = [P.C, zeros(2,1)];
61 % pick observer poles
62 wn_z_obs = 2.2/tr_z_obs; % natural frequency for position
63 wn_th_obs = 2.2/tr_theta_obs; % natural frequency for angle
64
65 des_observ_char_poly = conv([1,2*zeta_z*wn_z_obs,wn_z_obs^2],...
66 [1,2*zeta_th*wn_th_obs,wn_th_obs^2]);
67 des_observ_poles = roots(des_observ_char_poly);
68
69 % is the system observable?
70 if rank(observ(A2,C2))≠5,

```

CHAPTER 14. DISTURBANCE OBSERVERS

```

71     disp('System Not Observable');
72 else % if so, compute gains
73     L2 = place(A2', C2', [des_observ_poles;dist_observ_pole])';
74     P.L = L2(1:4,:);
75     P.Ld = L2(5,:);
76 end
77
78 sprintf('K:\t[%f, %f, %f, %f]\nki:\t%f\nL^T:\t[%f, %f, %f, %f;\n\t %f, %f, %f, %f]\nLd:\t[%f, %f
79     P.K(1), P.K(2), P.K(3), P.K(4), P.ki,...
80     P.L(1,1), P.L(2,1), P.L(3,1), P.L(4,1), P.L(1,2), P.L(2,2), P.L(3,2), P.L(4,2),...
81     P.Ld(1), P.Ld(2))

```

Matlab code for the observer based control is shown below:

```

1 function out=pendulum_ctrl(in,P)
2     z_r      = in(1);
3     z_m      = in(2);
4     theta_m = in(3);
5     t        = in(4);
6
7     % implement observer
8     persistent xhat      % estimated state (for observer)
9     persistent dhat      % estimate disturbance
10    persistent F         % delayed input (for observer)
11    if t<P.Ts,
12        xhat = [0;0;0;0];
13        dhat = 0;
14        F    = 0;
15    end
16    N = 10;
17    for i=1:N,
18        xhat = xhat + ...
19                    P.Ts/N*(P.A*xhat+P.B*(F+dhat)...
20                                +P.L*([z_m;theta_m]-P.C*xhat));
21        dhat = dhat + P.Ts/N*P.Ld*([z_m;theta_m]-P.C*xhat);
22    end
23    zhat = xhat(1);
24
25
26    % integrator
27    error = z_r - zhat;
28    persistent integrator
29    persistent error_d1
30    % reset persistent variables at start of simulation
31    if t<P.Ts==1,
32        integrator = 0;
33        error_d1   = 0;
34    end
35    integrator = integrator + (P.Ts/2)*(error+error_d1);
36    error_d1 = error;
37
38    % compute the state feedback controller
39    F_unsat = -P.K*xhat - P.ki*integrator - dhat;
40    F = sat(F_unsat, P.F_max);
41
42    % integrator anti-windup

```

```

43     if P.ki≠0,
44         integrator = integrator + P.Ts/P.ki*(F-F_unsat);
45     end
46
47     out = [F; xhat];
48
49 end
50
51 %
52 % saturation function
53 function out = sat(in,limit)
54     if in > limit,      out = limit;
55     elseif in < -limit,  out = -limit;
56     else                out = in;
57     end
58 end

```

See <http://controlbook.byu.edu> for the complete solution.

14.4 Design Study C. Satellite Attitude Control



Homework Problem C.14

- (a) Modify the parameter file from HW C.18 to use inexact parameters, as in HW C.15 and HW C.17. Explicitly add an additional input disturbance of 1 Newton-meters. Before adding the disturbance observer, run the simulation and observe the bias in the state. The integrator anti-windup that limits integral action when $\dot{\phi}$ is small will likely not work because of the state bias. You may need to comment out the anti-windup scheme for the integrator to work, in which case you will get large steady state error.
- (b) Add a disturbance observer to the controller, and verify that the steady state error in the estimator has been removed. Experiment with the system to understand the response with and without the integrator, and with and without anti-windup.
- (a) Modify your solution from HW C.13 so that the uncertainty parameter in `satellite_dynamics.m` is $\alpha = 0.2$, representing 20% inaccuracy in the knowledge of the system parameters, and so that the input disturbance is 1.0. Also, add noise to the output channels z_m and θ_m with standard deviation of 0.001.
- (b) Add a disturbance observer to the controller, and verify that the steady state error in the estimator has been removed. Tune the system to get good response.

Solution

Matlab code used to design the observer based controller is shown below:

```

1 % satellite parameter file
2
3
4 %%%%%%%%%%%%%%
5 % state feedback control with integrator
6 %%%%%%%%%%%%%%
7 % tuning parameters
8 wn_th = 0.6;
9 zeta_th = 0.707;
10 wn_phi = 1.1;
11 zeta_phi = 0.707;
12 integrator_pole = -1;
13 % pick observer poles
14 wn_th_obs = 10*wn_th;
15 wn_phi_obs = 10*wn_phi;
16
17 % state space design
18 P.A = [...
19     0, 0, 1, 0;...
20     0, 0, 0, 1;...
21     -P.k/P.Js, P.k/P.Js, -P.b/P.Js, P.b/P.Js;...
22     P.k/P.Jp, -P.k/P.Jp, P.b/P.Jp, -P.b/P.Jp;...
23 ];
24 P.B = [0; 0; 1/P.Js; 0];
25 P.C = [...
26     1, 0, 0, 0;...
27     0, 1, 0, 0;...
28 ];
29
30
31 % form augmented system
32 Cout = [0,1,0,0];
33 A1 = [P.A, zeros(4,1); -Cout, 0];
34 B1 = [P.B; 0];
35
36 % compute gains
37 ol_char_poly = charpoly(P.A);
38 des_char_poly = conv(conv([1,2*zeta_th*wn_th,wn_th^2],...
39 [1,2*zeta_phi*wn_phi,wn_phi^2]),...
40 poly(integrator_pole));
41 des_poles = roots(des_char_poly);
42
43 % is the system controllable?
44 if rank ctrb(A1,B1) ≠ 5,
45     disp('System Not Controllable');
46 else % if so, compute gains
47     K1 = place(A1,B1,des_poles);
48     P.K = K1(1:4);
49     P.ki = K1(5);
50 end
51
52 % observer design
53 % form augmented system for disturbance observer
54 A2 = [P.A, P.B; zeros(1,4), zeros(1,1)];
55 C2 = [P.C, zeros(2,1)];
56 % pick observer poles
57 wn_th_obs = 10*wn_th;

```

CHAPTER 14. DISTURBANCE OBSERVERS

```

58 wn_phi_obs    = 10*wn_phi;
59 dist_observ_pole = -1;
60 des_observ_char_poly = conv(conv(...
61             [1,2*zeta_phi*wn_phi_obs,wn_phi_obs^2],...
62             [1,2*zeta_th*wn_th_obs,wn_th_obs^2]),...
63             poly(dist_observ_pole));
64 des_observ_poles = roots(des_observ_char_poly);
65
66 % is the system observable?
67 if rank(observ(A2,C2))≠5,
68     disp('System Not Observable');
69 else % if so, compute gains
70     L2 = place(A2', C2', des_observ_poles)';
71     P.L = L2(1:4,:);
72     P.Ld = L2(5,:);
73 end
74
75
76 sprintf('K: [%f, %f, %f, %f]\nki: %f\nL: [%f, %f, %f, %f]^T\nLd: [%f, %f]^T\n',...
77     P.K(1), P.K(2), P.K(3), P.K(4), P.ki, P.L(1), P.L(2), P.L(3), P.L(4), P.Ld(1), P.Ld(2))

```

Matlab code for the observer based control is shown below:

```

1 function out=satellite_ctrl(in,P)
2     phi_r      = in(1);
3     theta_m   = in(2);
4     phi_m     = in(3);
5     t         = in(4);
6
7     % implement observer
8     persistent xhat      % estimated state (for observer)
9     persistent dhat      % estimate disturbance
10    persistent tau       % delayed input (for observer)
11    if t<P.Ts
12        xhat = [0;0;0;0];
13        dhat = 0;
14        tau   = 0;
15    end
16    N = 10;
17    for i=1:N
18        xhat = xhat + ...
19                    P.Ts/N*(P.A*xhat+P.B*(tau+dhat)...
20                                +P.L*([theta_m;phi_m]-P.C*xhat));
21        dhat = dhat + P.Ts/N*P.Ld*([theta_m;phi_m]-P.C*xhat);
22    end
23    phihat = xhat(2);
24
25    % integrator
26    error = phi_r - phihat;
27    persistent integrator
28    persistent error_d1
29    % reset persistent variables at start of simulation
30    if t<P.Ts==1
31        integrator = 0;
32        error_d1 = 0;
33    end

```

```

34     integrator = integrator + (P.Ts/2)*(error+error_d1);
35     error_d1 = error;
36
37     % compute the state feedback controller
38     tau_unsat = -P.K*xhat - P.ki*integrator - dhat;
39     tau = sat( tau_unsat, P.tauamax);
40
41         % integrator anti-windup
42     if P.ki≠0
43         integrator = integrator + P.Ts/P.ki*(tau-tau_unsat);
44     end
45
46     out = [tau; xhat];
47
48 end
49
50 %
51 % saturation function
52 function out = sat(in,limit)
53     if in > limit,      out = limit;
54     elseif in < -limit,   out = -limit;
55     else                  out = in;
56     end
57 end

```

See <http://controlbook.byu.edu> for the complete solution.

Notes and References

Disturbance observers are one example of how observers can be used to estimate a variety of different parameters in the system. In general, observes can be used to estimate model parameters like mass and inertia, but in this case, nonlinear observers are required. The most commonly used nonlinear observer is the extended Kalman filter. Derivations of the extended Kalman filter can be found in [25, 24, 26]. In robotics, observers are used to estimate the system state, but also the biases in gyroscopes and accelerometers [27], which are a from of input disturbances.

Part V

Loopshaping Control Design

This part of the book introduces the loopshaping design method based on frequency response characteristics of the system. In Part II we derived transfer function and state space models based on the linearized system. In Part III we showed that PID controllers can be used to regulate the output of second order systems. For higher order systems, PID can be used if the original system can be written as a cascade, or series, of second order systems. In Part IV we showed that controllers can be derived for higher order systems using state space models. The state space design method is a time-domain method that focuses on shaping the time-response of the system. In this part of the book we extend PID control using frequency domain ideas where we shape the frequency response of the system. When the transfer function is available, it provides a compact representation of the frequency response of the open loop system. Therefore, in this part we will focus on transfer function models of the system. The methods in this part are not constrained to second order systems and do not depend on a cascade structure. In fact, they do not even require a transfer function of the system, as long as the frequency response, or Bode plot is known.

Chapter 15 is a review of frequency response models of linear time-invariant systems, and develops intuition concerning the Bode plot of the input-output response. In Chapter 16 we show how performance specifications for reference tracking, disturbance rejection, and noise attenuation can be related to the frequency response of the open loop control-plant pair. Chapter 17 addresses stability of the closed-loop system from a frequency response point of view. Chapter 18 represents the culminating chapter in this part of the book and describes how to design the controller to achieve closed loop stability and to satisfy design specifications on reference tracking, disturbance rejection, and noise attenuation.

Chapter 15

Frequency Response of LTI Systems

15.1 Theory

15.1.1 Manipulating Complex Numbers

Any complex number z can be represented in rectangular coordinates as

$$z = \Re\{z\} + j\Im\{z\},$$

where $\Re\{z\}$ is the real part of z , and $\Im\{z\}$ is the imaginary part of z . Similarly, z can be represented in polar form as

$$z = |z| e^{j\angle z}, \quad (15.1)$$

where $|z|$ denotes the magnitude of z and $\angle z$ denotes the angle or phase of z . The relationship between the real and imaginary parts, and the magnitude and phase of z are depicted in Figure 15.1.

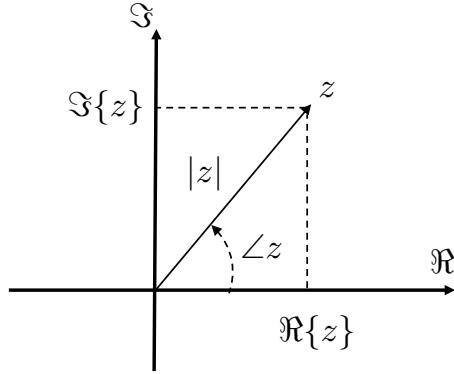


Figure 15.1: Relationship between the real and imaginary parts of z , and the magnitude and phase of z .

From the geometry shown in Figure 15.1 we can see that

$$|z| = \sqrt{\Re\{z\}^2 + \Im\{z\}^2}$$

$$\angle z = \tan^{-1} \frac{\Im\{z\}}{\Re\{z\}}.$$

Similarly, from Euler's relationship

$$e^{j\theta} = \cos \theta + j \sin \theta,$$

we have from Equation (15.1) that

$$z = |z| e^{j\angle z} = |z| \cos \angle z + j |z| \sin \angle z$$

which implies that

$$\Re\{z\} = |z| \cos \angle z$$

$$\Im\{z\} = |z| \sin \angle z.$$

The conjugate of z , denoted \bar{z} is given by

$$\bar{z} = \Re\{z\} - j\Im\{z\} = |z| e^{-j\angle z}.$$

Note that $z\bar{z} = |z|^2$.

Suppose that $z_1 = |z_1| e^{j\angle z_1}$ and $z_2 = |z_2| e^{j\angle z_2}$ are two complex numbers, then

$$z_1 z_2 = |z_1| e^{j\angle z_1} |z_2| e^{j\angle z_2} = |z_1| |z_2| e^{j\angle z_1} e^{j\angle z_2} = |z_1| |z_2| e^{j(\angle z_1 + \angle z_2)},$$

therefore when multiplying complex numbers, their magnitudes multiply, but their phases add. Similarly we have

$$\frac{z_1}{z_2} = \frac{|z_1| e^{j\angle z_1}}{|z_2| e^{j\angle z_2}} = \frac{|z_1| e^{j\angle z_1}}{|z_2| e^{j\angle z_2}} = \frac{|z_1|}{|z_2|} e^{j(\angle z_1 - \angle z_2)},$$

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

therefore when dividing complex numbers, their magnitudes divide, but their phases subtract.

A transfer function $H(s)$ is a complex number for any specific value of s . Therefore $H(s)$ can be expressed in both rectangular and polar forms as

$$H(s) = \Re\{H(s)\} + j\Im\{H(s)\} = |H(s)| e^{j\angle H(s)}.$$

For example, if $H(s) = \frac{1}{s+1}$, then when $s = 2+j3$

$$H(2+j3) = \frac{1}{2+j3+1} = \frac{1}{3+j3} = \frac{1}{3+j3} \frac{3-j3}{3-j3} = \frac{1}{6} - j \frac{1}{6} = \sqrt{\frac{1}{18}} e^{j\frac{\pi}{4}}.$$

If $H(s)$ has multiple poles and zeros, then magnitude and phase of $H(s)$ can be represented in terms of the magnitude and phase of the poles and zeros. For example, suppose that

$$H(s) = \frac{K(s+z_1)(s+z_2)\dots(s+z_m)}{(s+p_1)(s+p_2)\dots(s+p_n)}.$$

Since $(s+a)$ is a complex number for any s , it can be written in polar form as $s+a = |s+a| e^{j\angle(s+a)}$, therefore

$$\begin{aligned} H(s) &= \frac{K(s+z_1)(s+z_2)\dots(s+z_m)}{(s+p_1)(s+p_2)\dots(s+p_n)} \\ &= \frac{|K| e^{j\angle K} |s+z_1| e^{j\angle(s+z_1)} |s+z_2| e^{j\angle(s+z_2)} \dots |s+z_m| e^{j\angle(s+z_m)}}{|s+p_1| e^{j\angle(s+p_1)} |s+p_2| e^{j\angle(s+p_2)} \dots |s+p_n| e^{j\angle(s+p_n)}} \\ &= \frac{|K| |s+z_1| |s+z_2| \dots |s+z_m|}{|s+p_1| |s+p_2| \dots |s+p_n|} \frac{e^{j\angle K} e^{j\angle(s+z_1)} e^{j\angle(s+z_2)} \dots e^{j\angle(s+z_m)}}{e^{j\angle(s+p_1)} e^{j\angle(s+p_2)} \dots e^{j\angle(s+p_n)}} \\ &= \frac{|K| |s+z_1| |s+z_2| \dots |s+z_m|}{|s+p_1| |s+p_2| \dots |s+p_n|} e^{j(\angle K + \sum_{i=1}^m \angle(s+z_i) - \sum_{i=1}^n \angle(s+p_i))}. \end{aligned}$$

Therefore

$$|H(s)| = \frac{|K| |s+z_1| |s+z_2| \dots |s+z_m|}{|s+p_1| |s+p_2| \dots |s+p_n|} \quad (15.2)$$

$$\angle H(s) = \angle K + \sum_{i=1}^m \angle(s+z_i) - \sum_{i=1}^n \angle(s+p_i). \quad (15.3)$$

15.1.2 Frequency Response of LTI Systems

In systems theory, the magnitude and phase representation of the transfer function $H(s)$ is used because they have important physical meaning. In particular, suppose that $H(s)$ represents a physical system with input $u(t)$ and output $y(t)$ as shown in Figure 15.2.

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

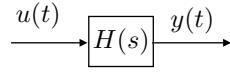


Figure 15.2: LTI system represented by the transfer function $H(s)$ with input $u(t)$ and output $y(t)$.

If the input to the system is given by a sinusoid of magnitude A and frequency ω_0 , i.e.,

$$u(t) = A \sin(\omega_0 t),$$

then the output is given by

$$y(t) = A |H(j\omega_0)| \sin(\omega_0 t + \angle H(j\omega_0)). \quad (15.4)$$

Therefore, the transfer function $H(s)$ is an elegant and compact representation of the frequency response of the system, in that the magnitude of $H(j\omega_0)$ describes the gain of the system for input signals with frequency ω_0 , and the phase of $H(j\omega_0)$ describes the phase shift imposed by the system on input signals with frequency ω_0 . For example, Figure 15.3 shows the input-output response of an LTI system with transfer function $H(s) = \frac{1}{s+1}$. The input signal, which is shown in grey, is $u(t) = \sin(\omega_0 t)$ where ω_0 changes in each subplot. The output signal is shown in blue and is given by Equation (15.4). Note that in subplot (a) the time scale is different than the other subplots. When $\omega_0 = 0.1$, $H(j\omega_0) \approx 0.995e^{j5^\circ}$. Accordingly, in subplot (a), the output is only slightly attenuated, with very little phase shift. In contrast, when $\omega_0 = 5$, $H(j\omega_0) \approx 0.2e^{j80^\circ}$, and in subplot (f) we see that the output is approximately 20% of the input with a phase shift approaching 90 degrees.

Therefore, the response of an LTI system to inputs at different frequencies, can be visualized by plotting the magnitude and phase of $H(j\omega)$ as a function of ω . The frequency response of $H(s) = 1/(s+1)$ is plotted in two different ways in Figure 15.4. In subfigure (a) a linear scale is used for both $|H(j\omega)|$ and $\angle H(j\omega)$, as well as the frequency scale ω . In subfigure (b), the scale for magnitude is in dB: $20 \log |H(j\omega)|$ whereas the scale for $\angle H(j\omega)$ is linear, however, in both cases a log scale for ω is used on the x -axis. The plot in subfigure (b) is called a Bode plot. It should be clear from Figure 15.4 that the Bode plot provides much more useful information than a linear plot, since it more effectively reveals the behavior of the system at lower frequencies. For this reason, Bode plots are the most common method to display the frequency response of the system.

As a note, since the magnitude of $H(j\omega)$ is displayed on the Bode plot as $20 \log |H(j\omega)|$ in units of dB, note that $20 \log |H(j\omega)| = 0$ dB implies that $|H(j\omega)| = 1$. Similarly $20 \log |H(j\omega)| = 20$ dB implies that $|H(j\omega)| = 10$ and $20 \log |H(j\omega)| = 40$ dB implies that $|H(j\omega)| = 100$. Going in the other direction, $20 \log |H(j\omega)| = -20$ dB implies that $|H(j\omega)| = 0.1$, and $20 \log |H(j\omega)| = -40$ dB implies that $|H(j\omega)| = 0.01$. In general,

$$20 \log |H(j\omega)| = 20n \text{ dB} \implies |H(j\omega)| = 10^n.$$

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

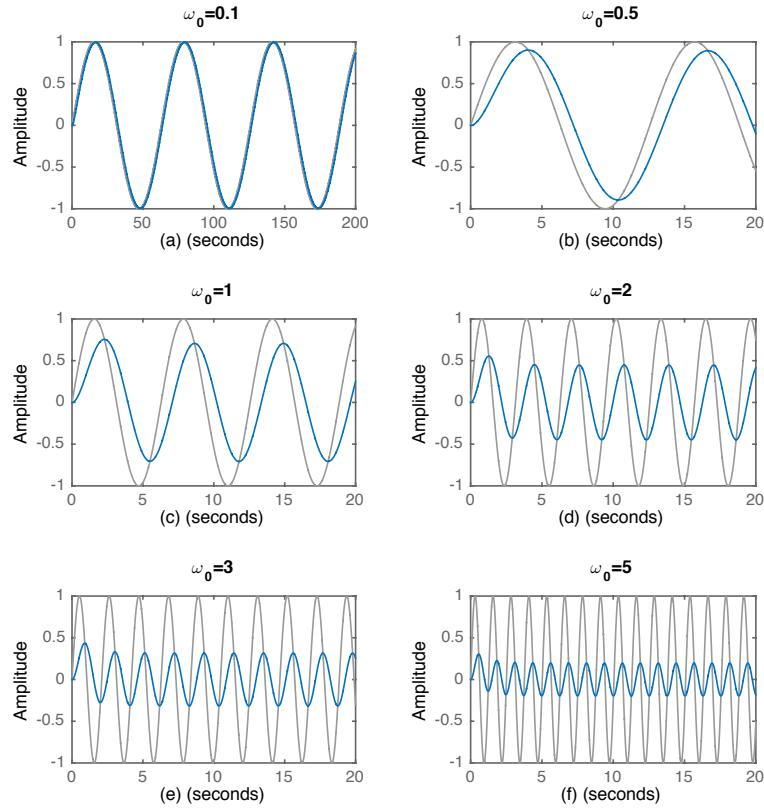


Figure 15.3: Frequency response to LTI system with transfer function $H(s) = 1/(s + 1)$.

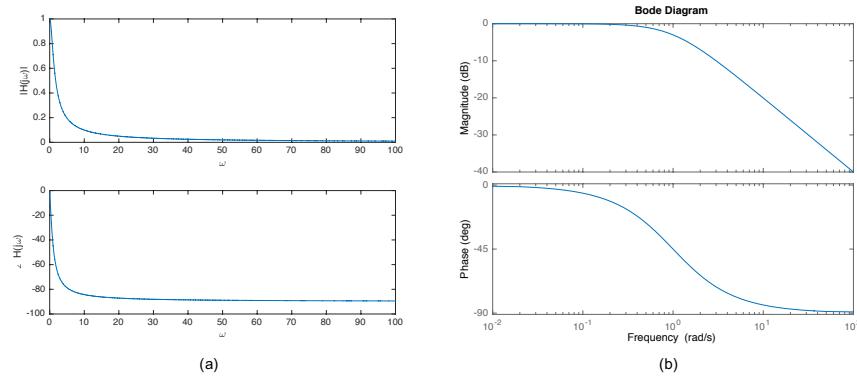


Figure 15.4: Frequency response of $H(s) = 1/(s + 1)$ on a linear scale (a), and a log plot (b).

15.1.3 Straight-line Approximations for Bode Plots

While the Bode plot for a system can be easily generated using the Matlab `bode` command, it is still useful to understand how to approximately draw a Bode plot by hand. The reason for this is that in the loopshaping design methodology, we add elements to the control transfer function $C(s)$ to "shape" the Bode plot of the loop gain $P(s)C(s)$ where $P(s)$ is the transfer function of the plant. Learning to approximately draw the Bode plot by hand will provide helpful insights in this process. Therefore, in this section we provide a brief tutorial on how to approximate the Bode plot for a transfer function $H(s)$.

We start by putting the transfer function into *Bode canonical form* by factoring out the poles and zeros as follow:

$$\begin{aligned} H(s) &= \frac{K(s + z_1)(s + z_2) \dots (s + z_m)}{(s + p_1)(s + p_2) \dots (s + p_n)} \\ &= \frac{Kz_1 z_2 \dots z_m}{p_1 p_2 \dots p_m} \frac{(1 + s/z_1)(1 + s/z_2) \dots (1 + s/z_m)}{(1 + s/p_1)(1 + s/p_2) \dots (1 + s/p_n)}. \end{aligned}$$

Therefore, following Equations (15.2) and (15.3) we have

$$|H(j\omega)| = \left| \frac{Kz_1 z_2 \dots z_m}{p_1 p_2 \dots p_m} \right| \frac{|1 + j\omega/z_1| |1 + j\omega/z_2| \dots |1 + j\omega/z_m|}{|1 + j\omega/p_1| |1 + j\omega/p_2| \dots |1 + j\omega/p_n|} \quad (15.5)$$

$$\angle H(j\omega) = \angle \left(\frac{Kz_1 z_2 \dots z_m}{p_1 p_2 \dots p_m} \right) + \sum_{i=1}^m \angle(1 + j\omega/z_i) - \sum_{i=1}^n \angle(1 + j\omega/p_i). \quad (15.6)$$

Since $20 \log |AB| = 20 \log |A| + 20 \log |B|$, Equation (15.5) gives

$$\begin{aligned} 20 \log |H(j\omega)| &= 20 \log \left| \frac{Kz_1 z_2 \dots z_m}{p_1 p_2 \dots p_m} \right| \\ &\quad + \sum_{i=1}^m 20 \log |1 + j\omega/z_i| - \sum_{i=1}^n 20 \log |1 + j\omega/p_i|. \end{aligned} \quad (15.7)$$

Therefore, drawing the Bode plot for a general transfer function with real poles and zeros, can be decomposed into drawing the Bode plot for each pole and zero, and then graphically adding them to get the general Bode plot. Note first that

$$20 \log \left| \frac{Kz_1 z_2 \dots z_m}{p_1 p_2 \dots p_m} \right|$$

is not a function of ω and is therefore a constant line on the Bode plot. Also note that

$$\angle \left(\frac{Kz_1 z_2 \dots z_m}{p_1 p_2 \dots p_m} \right)$$

is also a constant and is either 0 degrees if $\left(\frac{Kz_1 z_2 \dots z_m}{p_1 p_2 \dots p_m} \right) > 0$, or 180 degrees if $\left(\frac{Kz_1 z_2 \dots z_m}{p_1 p_2 \dots p_m} \right) < 0$.

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

We start by drawing the Bode plot when $H(s) = s + z$ has a single zero at $s = -z$ and $z > 0$. The first step is to put the transfer function in Bode canonical form as

$$H(j\omega) = z(1 + j\frac{\omega}{z}).$$

Therefore, from Equation (15.7) we have

$$20 \log |H(j\omega)| = 20 \log |z| + 20 \log \left| 1 + j\frac{\omega}{z} \right| = 20 \log |z| + 20 \log \sqrt{1 + (\omega/z)^2}.$$

When $\omega/z \ll 1$ we have that $\sqrt{1 + (\omega/z)^2} \approx 1$, implying that

$$20 \log |H(j\omega)| \approx 20 \log |z|.$$

Note that since this term is not a function of ω , it is a constant value on the Bode plot. On the other hand, when $\omega/z \gg 1$ we have that $\sqrt{1 + (\omega/z)^2} \approx |\omega/z|$, giving

$$20 \log |H(j\omega)| \approx 20 \log |z| + 20 \log |\omega/z|.$$

Note that the second term is a straight line on a Bode plot that intersects the 0 dB axis at $\omega = z$ and increases 20 dB for every factor of 10 increase in ω . Therefore, on the Bode plot, the slope of the second term is +20 dB/decade. A straight line approximation is shown in Figure 15.5, along with the Bode plot produced by Matlab.

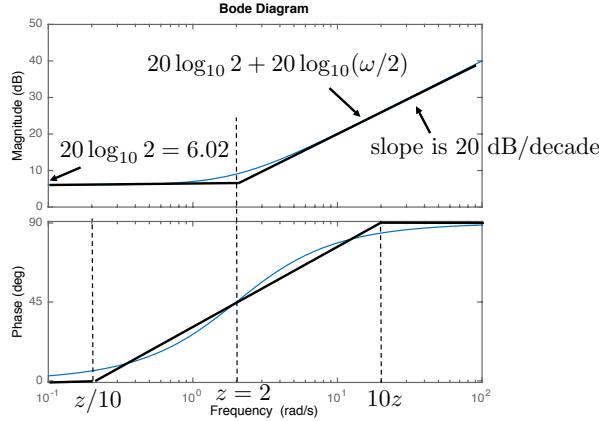


Figure 15.5: Straight line approximation for a single zero.

Note that the largest mismatch between the straight line approximation and the actual Bode plot is at $\omega = z$. In that case we have

$$20 \log |H(jz)| \approx 20 \log |z| + 20 \log \sqrt{2} = 20 \log |z| + 3 \text{ dB}.$$

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

Therefore, at the corner frequency z the Bode plot is 3 dB above the low frequency response.

To approximate the phase plot, note that

$$\angle H(j\omega) = \angle z(1 + j\omega/z) = \tan^{-1} \frac{\omega}{z}. \quad (15.8)$$

When $\omega = z$ we have that $\tan^{-1} 1 = 45$ degrees. When $\omega \ll z$, we have that $\tan^{-1}(\omega/z) \approx 0$ degrees, and when $\omega \gg z$, we have that $\tan^{-1}(\omega/z) \approx 90$ degrees. The phase plot as calculated by Matlab is shown in Figure 15.5. An adequate straight-line approximation is obtained by setting the phase to zeros when $\omega < z/10$, i.e., one decade less than z , and setting the phase to 90 degrees when $\omega > 10z$, i.e., one decade greater than z , and drawing a straight line between those points so that the phase at z is 45 degrees. The straight line approximation for the phase is also shown in black Figure 15.5.

The straight line approximation for phase as described above assumes that $z > 0$, i.e., that the zero is in the left half of the complex plane. For right half plane zeros when $z < 0$, the phase as calculated in Equation (15.6) is

$$\angle H(j\omega) = \angle z + \angle(1 + j\omega/z) = 180^\circ - \tan^{-1} \frac{\omega}{|z|}.$$

Therefore, the phase starts at 180 degrees and decreases by 90 degrees as ω gets large rather than increasing. Figure 15.6 shows the Bode plot and its straight line approximation when the zero is in the right half plane. When a system has a zero in the right half plane, it is called non-minimum phase because of the 180 degree phase increase at low frequency.

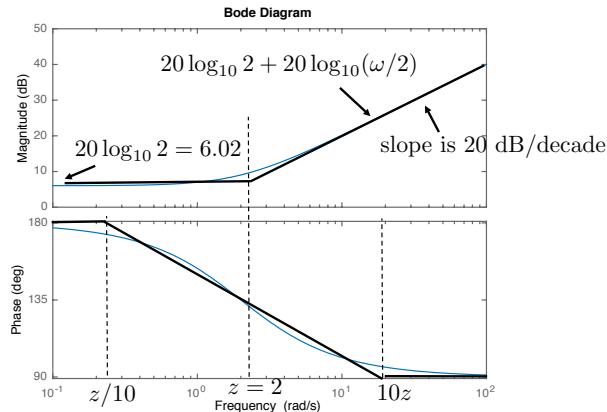


Figure 15.6: Straight line approximation for a zero in the right half of the complex plane.

Now consider drawing the Bode plot for a single pole, i.e., $H(s) = \frac{p}{s+p}$, which has a single pole at $s = -p$ which is in the right half plane when $p > 0$.

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

In Bode canonical form $H(s)$ becomes

$$H(j\omega) = \frac{1}{1 + j\frac{\omega}{p}}.$$

Therefore, from Equation (15.7) we have

$$20 \log |H(j\omega)| = 20 \log |1| - 20 \log \left| 1 + j\frac{\omega}{p} \right| = -20 \log \sqrt{1 + (\omega/p)^2}.$$

When $\omega/p \ll 1$, we have that $\sqrt{1 + (\omega/p)^2} \approx 1$, implying that

$$20 \log |H(j\omega)| \approx 0.$$

Note that since this term is not a function of ω , it is a constant value at 0 dB on the Bode plot. On the other hand, when $\omega/p \gg 1$, we have that $\sqrt{1 + (\omega/p)^2} \approx |\omega/p|$, giving

$$20 \log |H(j\omega)| \approx -20 \log |\omega/p|.$$

Note that the second term is a straight line on a Bode plot that intersects the 0 dB axis when $\omega = p$ and decreases at -20 dB for every factor of 10 increase in ω . Therefore, on the Bode plot, the slope of the second term is -20 dB/decade. A straight line approximation is shown in Figure 15.7, along with the Bode plot produced by Matlab.

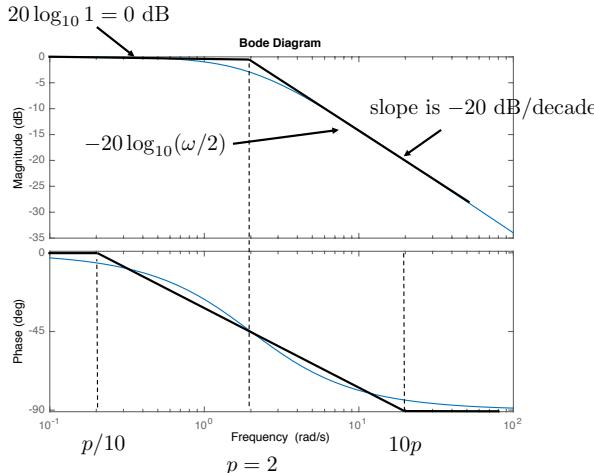


Figure 15.7: Straight line approximation for a single pole.

Note that the largest mismatch between the straight line approximation and the actual Bode is at $\omega = p$. In that case we have

$$20 \log |H(jp)| \approx -20 \log \sqrt{2} = -3 \text{ dB}.$$

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

Therefore, at the corner frequency p the Bode plot is 3 dB below the low frequency response.

To approximate the phase plot, note that

$$\angle H(j\omega) = -\angle(1 + j\frac{\omega}{p}) = -\tan^{-1} \frac{\omega}{p}. \quad (15.9)$$

When $\omega = p$ we have that $-\tan^{-1} 1 = -45$ degrees. When $\omega \ll p$, we have that $-\tan^{-1}(\omega/p) \approx 0$ degrees, and when $\omega \gg p$, $-\tan^{-1}(\omega/p) \approx -90$ degrees. The phase plot as calculated by Matlab is shown in Figure 15.7. An adequate straight-line approximation is obtained by setting the phase to zeros when $\omega < p/10$, i.e., one decade less than p , and setting the phase to -90 degrees when $\omega > 10p$, i.e., one decade greater than p , and drawing a straight line between those points so that the phase at p is -45 degrees. The straight line approximation for the phase is also shown in Figure 15.7.

The straight line approximation for phase as described above assumes that $p > 0$, i.e., that the pole is in the left half of the complex plane. For right half plane poles when $p < 0$, the phase as calculated in Equation (15.6) is

$$\angle H(j\omega) = -\angle(1 + j\omega/p) = \tan^{-1} \frac{\omega}{|p|}.$$

Therefore, the phase starts at 0 degrees and increases by 90 degrees as ω gets large. Figure 15.8 shows the Bode plot and its straight line approximation when the pole is in the right half plane. Note that stability and instability are not directly evident from shapes of the magnitude and phase plots.

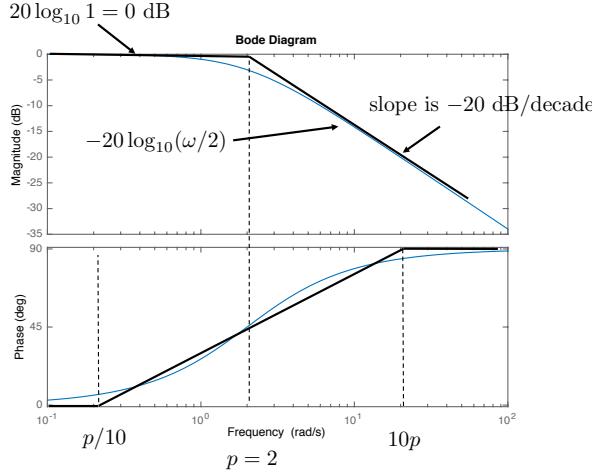


Figure 15.8: Straight line approximation for a pole in the right half of the complex plane.

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

Suppose now that the poles are complex. Let

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}.$$

In Bode canonical form we have

$$\begin{aligned} H(j\omega) &= \frac{1}{\omega_n^2} \frac{\omega_n^2}{\frac{(j\omega)^2}{\omega_n^2} + 2\zeta\omega_n \left(\frac{j\omega}{\omega_n^2}\right) + 1} \\ &= \frac{1}{\left(1 - \left(\frac{\omega}{\omega_n}\right)^2\right) + j2\zeta\left(\frac{\omega}{\omega_n}\right)} \\ &= \frac{1}{\sqrt{\left(1 - \frac{\omega^2}{\omega_n^2}\right)^2 + 4\zeta^2 \frac{\omega^2}{\omega_n^2}}} e^{-j \tan^{-1} \left(\frac{2\zeta \frac{\omega}{\omega_n}}{1 - \frac{\omega^2}{\omega_n^2}} \right)}. \end{aligned}$$

Therefore

$$\begin{aligned} |H(j\omega)| &= \frac{1}{\sqrt{\left(1 - \frac{\omega^2}{\omega_n^2}\right)^2 + 4\zeta^2 \frac{\omega^2}{\omega_n^2}}} \\ \angle H(j\omega) &= -\tan^{-1} \left(\frac{2\zeta \frac{\omega}{\omega_n}}{1 - \frac{\omega^2}{\omega_n^2}} \right). \end{aligned}$$

When $\omega \ll \omega_n$ we have that $20 \log_{10} |H(j\omega)| \approx 20 \log_{10} |1| = 0$ dB. Also, $\angle H(j\omega) \approx -\tan^{-1} \frac{0}{1} = 0$ degrees. Alternatively, when $\omega \gg \omega_n$ we have

$$\begin{aligned} 20 \log_{10} |H(j\omega)| &\approx -20 \log \sqrt{\left(\frac{\omega^2}{\omega_n^2}\right)^2 + 4\zeta^2 \frac{\omega^2}{\omega_n^2}} \\ &\approx -20 \log \sqrt{\left(\frac{\omega^2}{\omega_n^2}\right)^2} \\ &\approx -20 \log \left| \frac{\omega}{\omega_n} \right|^2 \\ &= -40 \log \left| \frac{\omega}{\omega_n} \right|, \end{aligned}$$

which is a straight line that crosses the 0 dB axis at ω_n with a slope of -40 dB/decade.

When $\omega = \omega_n$ we have $20 \log_{10} |H(j\omega)| = -20 \log_{10} \sqrt{4\zeta^2} = -20 \log_{10} |2\zeta|$.

When $\zeta = 0.707 = \frac{\sqrt{2}}{2}$, then $20 \log_{10} |H(j\omega)| = -20 \log_{10} 1/\sqrt{2} = -3$ dB. For the angle plot, when $\omega \ll \omega_n$ we have $\angle H(j\omega) \approx \tan^{-1} \frac{0}{1} = 0$ degrees. When $\omega \gg \omega_n$ we have $\angle H(j\omega) \approx \tan^{-1} (2\zeta \frac{\omega_n}{\omega}) \approx -180$ degrees, and when $\omega = \omega_n$ we have $\angle H(j\omega) \approx -\tan^{-1} \frac{2\zeta}{0} = -90$ degrees.

The Bode plot for different value of ζ is shown in Figure 15.9

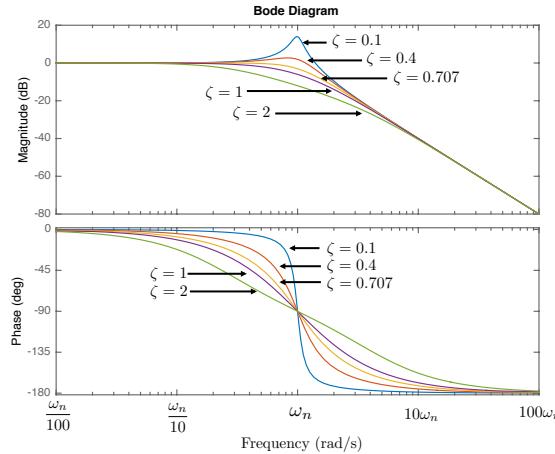


Figure 15.9: The Bode plot for a second order system for various values of ζ .

15.1.4 Example

Find the straight line approximation for the Bode plot of the transfer function

$$H(s) = \frac{20}{s(s + 10)}. \quad (15.10)$$

In Bode canonical form we have

$$H(j\omega) = \frac{20}{10} \frac{1}{j\omega(1 + j\frac{\omega}{10})}.$$

Therefore

$$20 \log_{10} |H(j\omega)| = 20 \log_{10} 2 - 20 \log_{10} |\omega| - 20 \log_{10} \sqrt{1 + \left(\frac{\omega}{10}\right)^2}. \quad (15.11)$$

Note that the term due to the pole at zeros, namely $-20 \log_{10} |\omega|$ is a straight line that intercepts the 0 dB line at $\omega = 1$, and has a slope of -20 dB per decade. Therefore, the Bode plot for magnitude will be the graphical addition of three terms:

1. A constant term at $20 \log_{10} 2 = 2$ dB,
2. A straight line with slope of -20 dB/decade, passing through the 0 dB line at $\omega = 1$, and
3. A first order low pass filter with cut-off frequency at $\omega = 10$.

Similarly, the phase is given by

$$\begin{aligned}\angle H(j\omega) &= \angle 2 - \angle j\omega - \angle(1 + j\frac{\omega}{10}) \\ &= 0 - \tan^{-1} \frac{\omega}{0} - \tan^{-1} \frac{\omega}{10} \\ &= -90^\circ - \tan^{-1} \frac{\omega}{10}.\end{aligned}$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 15.10.

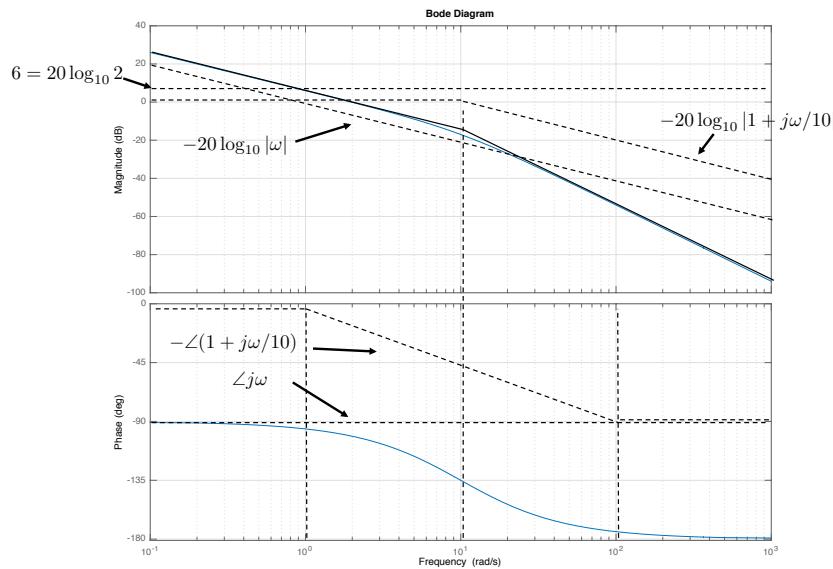


Figure 15.10: Bode plot for the transfer function given in Equation (15.10).

15.1.5 Example

Find the straight line approximation for the Bode plot of the transfer function

$$H(s) = \frac{8000(s+1)(s+10)}{(s+2)(s+20)(s+200)}. \quad (15.12)$$

In Bode canonical form we have

$$\begin{aligned}H(j\omega) &= \frac{8000 \cdot 10}{2 \cdot 20 \cdot 200} \frac{(1 + j\omega)(1 + j\frac{\omega}{10})}{(1 + j\frac{\omega}{2})(1 + j\frac{\omega}{20})(1 + j\frac{\omega}{200})} \\ &= 10 \frac{(1 + j\omega)(1 + j\frac{\omega}{10})}{(1 + j\frac{\omega}{2})(1 + j\frac{\omega}{20})(1 + j\frac{\omega}{200})}.\end{aligned}$$

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

Therefore

$$\begin{aligned} 20 \log_{10} |H(j\omega)| &= 20 \log_{10} 10 + 20 \log_{10} |1 + j\omega| + 20 \log_{10} \left|1 + j\frac{\omega}{10}\right| \\ &\quad - 20 \log_{10} \left|1 + j\frac{\omega}{2}\right| - 20 \log_{10} \left|1 + j\frac{\omega}{20}\right| - 20 \log_{10} \left|1 + j\frac{\omega}{200}\right|. \end{aligned} \quad (15.13)$$

Therefore, the Bode plot for magnitude will be the graphical addition of a constant gain, two zeros, and three poles. Similarly, the phase is given by

$$\begin{aligned} \angle H(j\omega) &= \angle 10 + \angle(1 + j\omega) + \angle\left(1 + j\frac{\omega}{10}\right) \\ &\quad - \angle\left(1 + j\frac{\omega}{2}\right) - \angle\left(1 + j\frac{\omega}{20}\right) - \angle\left(1 + j\frac{\omega}{200}\right). \end{aligned}$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 15.11.

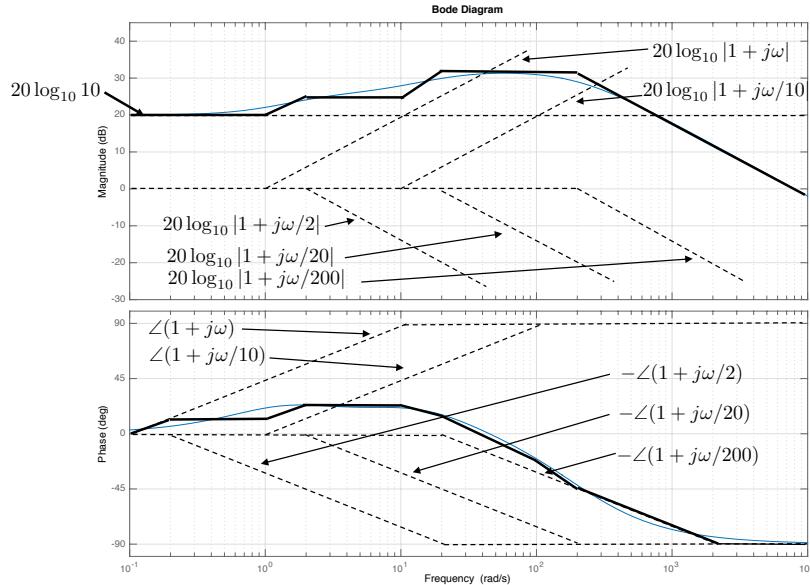


Figure 15.11: Bode plot for the transfer function given in Equation (15.12).

15.2 Design Study A. Single Link Robot Arm



Homework Problem A.15

Draw by hand the Bode plot of the single link robot arm from torque $\tilde{\tau}$ to angle $\tilde{\theta}$ given that the equilibrium angle is $\theta_e = 0$. Use the Matlab `bode` command and compare your results.

Solution

From HW A.5, the transfer function for the single link robot arm is

$$P(s) = \frac{3/m\ell^2}{s(s + 3b/m\ell^2)} = \frac{44.44}{s(s + 0.4444)}. \quad (15.14)$$

In Bode canonical form we have

$$P(j\omega) = \frac{100}{(j\omega)(1 + j\frac{\omega}{0.4444})}$$

Therefore

$$20 \log_{10} |P(j\omega)| = 20 \log_{10} 100 - 20 \log_{10} |j\omega| - 20 \log_{10} \left| 1 + j\frac{\omega}{0.4444} \right|. \quad (15.15)$$

Therefore, the Bode plot for magnitude will be the graphical addition of a constant gain, an integrator, and a pole. Similarly, the phase is given by

$$\angle P(j\omega) = \angle 100 - \angle(j\omega) - \angle(1 + j\frac{\omega}{0.4444}).$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 15.12.

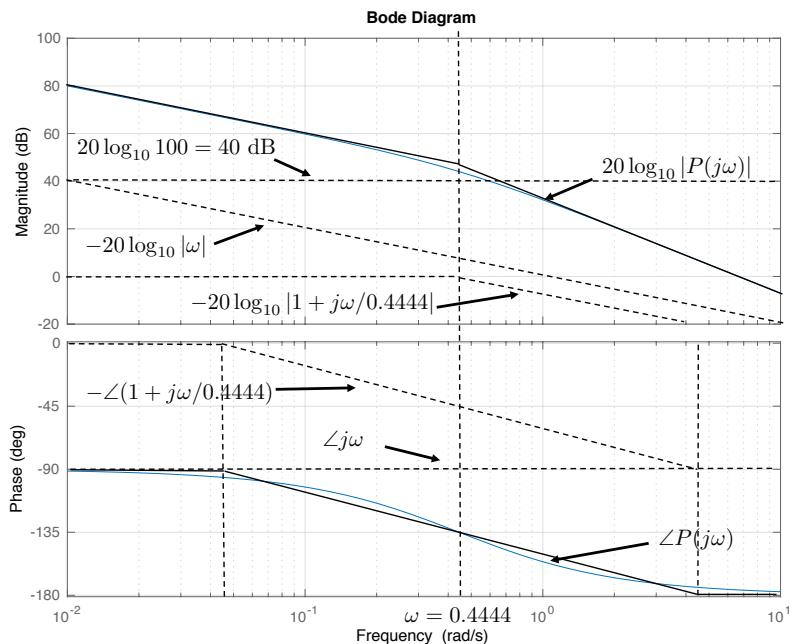


Figure 15.12: Bode plot for the transfer function given in Equation (15.14).

The Matlab command to generate the Bode plot is

```

1 >> P = tf([100],[1, 0.4444, 0]);
2 >> figure(1), clf, bode(P), grid on

```

15.3 Design Study B. Inverted Pendulum



Homework Problem B.15

- (a) Draw by hand the Bode plot of the inner loop transfer function from force F to angle θ for the inverted pendulum. Use the Matlab `bode` command and compare your results.
- (b) Draw by hand the Bode plot of the outer loop transfer function from angle θ to position z for the inverted pendulum. Use the Matlab `bode` command and compare your results.

Solution

From HW B.5, the transfer function for the inner loop of the inverted pendulum is

$$P_{in}(s) = \frac{-2/m_2\ell}{s^2 - \frac{2(m_1+m_2)g}{m_2\ell}} = \frac{-4}{s^2 - 49} = \frac{-4}{(s+7)(s-7)}. \quad (15.16)$$

In Bode canonical form we have

$$P_{in}(j\omega) = \frac{0.0816}{(1 + j\frac{\omega}{7})(1 - j\frac{\omega}{7})}$$

Therefore

$$\begin{aligned} 20 \log_{10} |P_{in}(j\omega)| &= 20 \log_{10} 0.0816 \\ &\quad - 20 \log_{10} \left| 1 + j\frac{\omega}{7} \right| - 20 \log_{10} \left| 1 - j\frac{\omega}{7} \right|. \end{aligned} \quad (15.17)$$

Therefore, the Bode plot for magnitude will be the graphical addition of a constant gain, a right half plane pole, and a left half plane pole. Similarly, the phase is given by

$$\angle P_{in}(j\omega) = \angle 0.0816 - \angle \left(1 + j\frac{\omega}{7} \right) - \angle \left(1 - j\frac{\omega}{7} \right).$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 15.13.

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

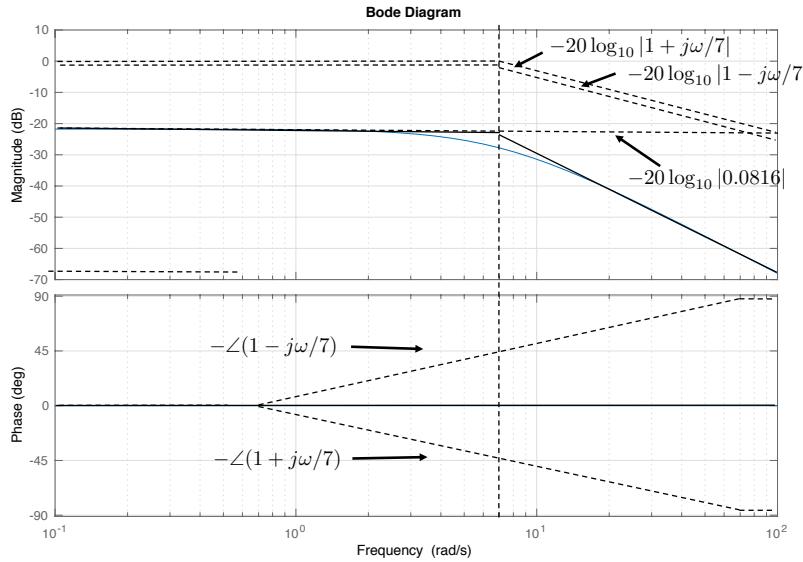


Figure 15.13: Bode plot for the transfer function given in Equation (15.17).

The Matlab command to generate the Bode plot is

```
1 >> Pin = tf([-4], [1, 0, -49]);
2 >> figure(1), clf, bode(Pin), grid on
```

From HW B.5, the transfer function for the outer loop of the inverted pendulum is

$$P_{out}(s) = \frac{g}{s^2} = \frac{9.8}{s^2}. \quad (15.18)$$

In Bode canonical form we have

$$P_{out}(j\omega) = \frac{9.8}{(j\omega)^2}.$$

Therefore

$$20 \log_{10} |P_{out}(j\omega)| = 20 \log_{10} 9.8 - 20 \log_{10} |\omega|^2. \quad (15.19)$$

Similarly, the phase is given by

$$\angle P_{out}(j\omega) = \angle 9.8 - \angle(j\omega) - \angle(\omega).$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 15.14.

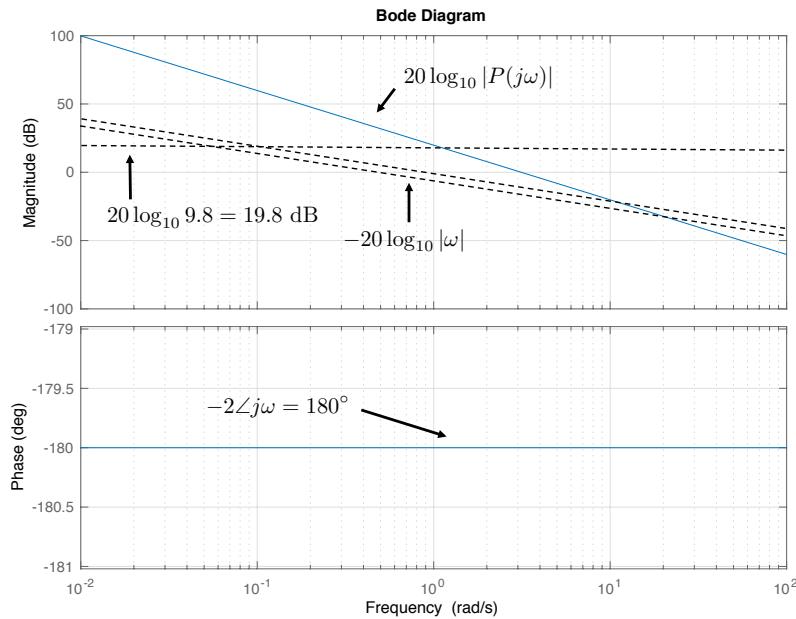


Figure 15.14: Bode plot for the transfer function given in Equation (15.19).

The Matlab command to generate the Bode plot is

```
1 >> Pout = tf([9.8], [1, 0, 0]);
2 >> figure(1), clf, bode(Pout), grid on
```

15.4 Design Study C. Satellite Attitude Control



Homework Problem C.15

- (a) Draw by hand the Bode plot of the inner loop transfer function from torque τ to angle θ for the satellite attitude problem. Use the Matlab `bode` command and compare your results.
- (b) Draw by hand the Bode plot of the outer loop transfer function from body angle θ to panel angle ϕ for the satellite attitude problem. Use the Matlab `bode` command and compare your results.

Solution

From HW C.5, the transfer function for the inner loop of the satellite attitude problem is

$$P_{in}(s) = \frac{1/J_s}{s^2 + \frac{b}{J_s}s + \frac{k}{J_s}} = \frac{0.2}{s^2 + 0.01s + 0.03}. \quad (15.20)$$

In Bode canonical form we have

$$P_{in}(j\omega) = \frac{6.67}{1 + 0.33j\omega + \left(j\frac{\omega}{0.173}\right)^2}$$

Therefore

$$20 \log_{10} |P_{in}(j\omega)| = 20 \log_{10} 6.67 - 20 \log_{10} \left| 1 + 0.33j\omega + \left(j\frac{\omega}{0.173}\right)^2 \right| \quad (15.21)$$

Therefore, the Bode plot for magnitude will be the graphical addition of a constant gain, and a complex pole. Similarly, the phase is given by

$$\angle P_{in}(j\omega) = \angle 6.67 - \angle \left(1 + 0.33j\omega + \left(j\frac{\omega}{0.173}\right)^2 \right).$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 15.15.

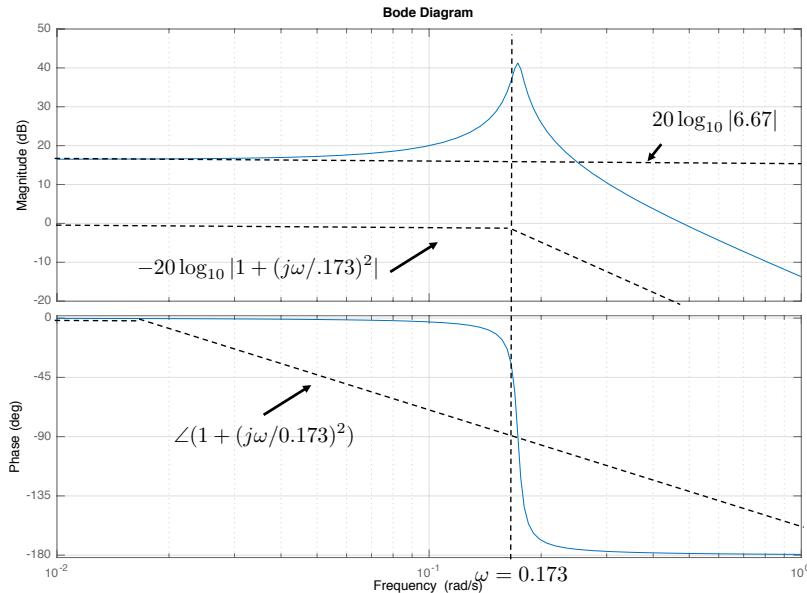


Figure 15.15: Bode plot for the transfer function given in Equation (15.21).

The Matlab command to generate the Bode plot is

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

```

1 >> Pin = tf([0.2], [1, 0.01, 0.03]);
2 >> figure(1), clf, bode(Pin), grid on

```

From HW C.5, the transfer function for the outer loop of the satellite is

$$P_{out}(s) = \frac{\frac{b}{J_p}s + \frac{k}{J_p}}{s^2 + \frac{b}{J_p}s + \frac{k}{J_p}} = \frac{0.05s + 0.15}{s^2 + 0.05s + 0.15}. \quad (15.22)$$

In Bode canonical form we have

$$P_{out}(j\omega) = \frac{1 + j\frac{\omega}{3}}{1 + j\frac{\omega}{3} + \left(j\frac{\omega}{0.3873}\right)^2}$$

Therefore

$$20 \log_{10} |P_{out}(j\omega)| = 20 \log_{10} |1 + j\omega/3| - 20 \log_{10} \left| 1 + j\omega/3 + \left(j\frac{\omega}{0.3873}\right)^2 \right| \quad (15.23)$$

Therefore, the Bode plot for magnitude will be the graphical addition of a zero, and a complex pole. Similarly, the phase is given by

$$\angle P_{out}(j\omega) = \angle(1 + j\omega/3) - \angle\left(1 + j\omega/3 + \left(j\frac{\omega}{0.3873}\right)^2\right).$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 15.16.

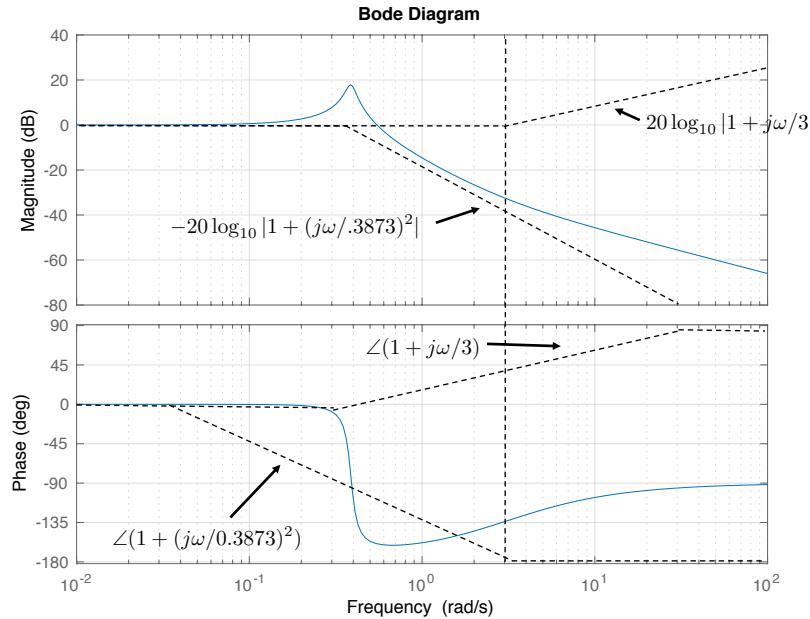


Figure 15.16: Bode plot for the transfer function given in Equation (15.23).

CHAPTER 15. FREQUENCY RESPONSE OF LTI SYSTEMS

The Matlab command to generate the Bode plot is

```
1 >> Pout = tf([0.05, 0.15], [1, 0.05, 0.15]);  
2 >> figure(1), clf, bode(Pout), grid on
```

Notes and References

A good introduction to frequency response for linear time-invariant system is [16]. A good overview of the Bode plot and straight line approximations to the Bode plot is contained in [4], and many other introductory textbooks on feedback control.

Chapter 16

Frequency Domain Specifications

16.1 Theory

In this chapter we show how the magnitude of the frequency response of the open loop system can be used to determine the performance of the closed-loop feedback system. A general feedback loop is shown in Figure 16.1. In this feedback system, $r(t)$ is the reference input to be tracked by $y(t)$, $d_{in}(t)$ is an unknown input disturbance, d_{out} is an unknown output disturbance, and $n(t)$ is an unknown noise signal. Note that because of the presence of n , the error signal $e = r - y$ is no longer the input to $C(s)$: the input to $C(s)$ is now $r - y - n$. To find the transfer function from the inputs r , d_{in} , d_{out} , and n to the error $e(t)$ we start by finding the transfer functions to y . Starting at $Y(s)$ and following the loop backward until returning to Y we get

$$\begin{aligned} Y(s) &= -D_{out}(s) + P(-D_{in}(s) + C(R(s) - N(s) + Y(s))) \\ \implies (1+PC)Y(s) &= PCR(s) - PCN(s) - D_{out}(s) - PD_{in}(s) \\ Y(s) &= \frac{PC}{1+PC}R(s) - \frac{PC}{1+PC}N(s) - \frac{1}{1+PC}D_{out}(s) - \frac{P}{1+PC}D_{in}(s). \end{aligned} \tag{16.1}$$

Defining the error to be $E(s) \triangleq R(s) - Y(s)$ we get

$$E(s) = R(s) - Y(s) \tag{16.2}$$

$$= \frac{1}{1+PC}R(s) + \frac{PC}{1+PC}N(s) + \frac{1}{1+PC}D_{out}(s) + \frac{P}{1+PC}D_{in}(s), \tag{16.3}$$

Therefore,

1. The transfer function from the reference r to the error e is $\frac{1}{1+P(s)C(s)}$,

CHAPTER 16. FREQUENCY DOMAIN SPECIFICATIONS

2. The transfer function from the noise n to the error e is $\frac{P(s)C(s)}{1+P(s)C(s)}$,
3. The transfer function from the output disturbance d_{out} to the error e is $\frac{1}{1+P(s)C(s)}$, and
4. The transfer function from the input disturbance d_{in} to the error e is $\frac{P(s)}{1+P(s)C(s)}$.

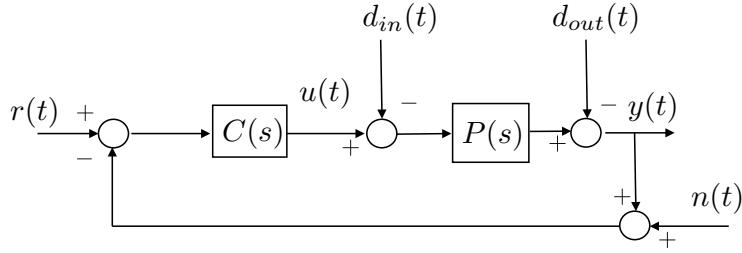


Figure 16.1: General feedback loop.

For typical applications, we have that

1. The reference signal $r(t)$ is a low frequency signal with frequency content below ω_r ,
2. The noise signal $n(t)$ is a high frequency signal with frequency content above ω_d ,
3. The output disturbance signal $d_{out}(t)$ is a low frequency signal with frequency content below $\omega_{d_{out}}$,
4. The input disturbance signal $d_{in}(t)$ is a low frequency signal with frequency content below $\omega_{d_{in}}$.

The objective for the control design is to find $C(s)$ so that the following specifications are satisfied:

1. **Tracking:** Track reference signals $r(t)$ with frequency content below ω_r so that the error due to the reference signal satisfies

$$|e(t)| \leq \gamma_r |r(t)|,$$

2. **Noise Attenuation:** Attenuate noise signals $n(t)$ with frequency content above ω_{no} so that the error due to the noise signal satisfies

$$|e(t)| \leq \gamma_n |n(t)|,$$

3. **Reject Output Disturbances:** Reject output disturbance signals $d_{out}(t)$ with frequency content below $\omega_{d_{out}}$ so that the error due to the output disturbance signal satisfies

$$|e(t)| \leq \gamma_{d_{out}} |d_{out}(t)|,$$

4. **Reject Input Disturbances:** Reject input disturbance signals $d_{in}(t)$ with frequency content below $\omega_{d_{in}}$ so that the error due to the input disturbance signal satisfies

$$|e(t)| \leq \gamma_{d_{in}} |d_{in}(t)|.$$

The objective in the remainder of this chapter is to show how each of these specifications can be translated into a requirement on the magnitude of the (open) loop gain $20 \log_{10} |P(j\omega)C(j\omega)|$.

16.1.1 Tracking

The transfer function from the reference signal $r(t)$ to the error $e(t)$ is

$$E(s) = \frac{1}{1 + P(s)C(s)} R(s).$$

Therefore

$$|E(j\omega)| \leq \left| \frac{1}{1 + P(j\omega)C(j\omega)} \right| |R(j\omega)|.$$

The objective is to track reference signals $r(t)$ with frequency content below ω_r so that the error due to the reference signal satisfies

$$|e(t)| \leq \gamma_r |r(t)|.$$

Therefore, the controller $C(s)$ needs to be designed so that

$$\left| \frac{1}{1 + P(j\omega)C(j\omega)} \right| \leq \gamma_r,$$

or equivalently so that

$$|1 + P(j\omega)C(j\omega)| \geq \frac{1}{\gamma_r}.$$

If $|P(j\omega)C(j\omega)| \gg 1$, then a sufficient condition is that

$$|P(j\omega)C(j\omega)| \geq \frac{1}{\gamma_r},$$

or equivalently

$$20 \log_{10} |P(j\omega)C(j\omega)| \geq 20 \log_{10} 1/\gamma_r, \quad (16.4)$$

which needs to be satisfied for all $\omega \leq \omega_r$. On a Bode plot, the constraint represented by Equation (16.4) is shown graphically in Figure 16.2.

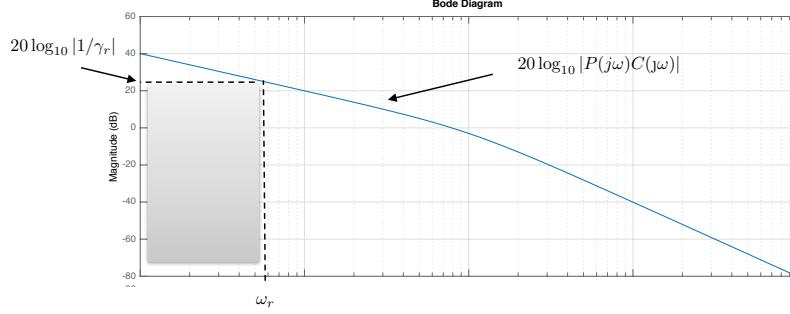


Figure 16.2: Bode plot representation of the loopshaping constraint corresponding to tracking $r(t)$ so that the corresponding error satisfies $|e(t)| \leq \gamma_r |r(t)|$.

Alternatively, from the Bode plot of PC we can compute the tracking characteristics of the closed loop system. For example, suppose that on the Bode plot

$$20 \log_{10} |P(j\omega)C(j\omega)| > B_r \text{ dB}$$

for all $\omega < \omega_r$, then

$$|P(j\omega)C(j\omega)| > 10^{B_r/20},$$

for all $\omega < \omega_r$. If $|P(j\omega)C(j\omega)| \gg 1$ for all $\omega < \omega_r$, then

$$\left| \frac{1}{1 + P(j\omega)C(j\omega)} \right| \lesssim 10^{-B_r/20},$$

which implies that

$$|E(j\omega)| \leq 10^{-B_r/20} |R(j\omega)|$$

for all $\omega < \omega_r$, or in other words that

$$|e(t)| \leq 10^{-B_r/20} |r(t)|, \quad (16.5)$$

for all $r(t)$ with frequency content below ω_r , which implies that

$$\gamma_r = 10^{-B_r/20}.$$

16.1.2 Noise Attenuation

The transfer function from the noise signal $n(t)$ to the error $e(t)$ is

$$E(s) = \frac{P(s)C(s)}{1 + P(s)C(s)} N(s).$$

Therefore

$$|E(j\omega)| \leq \left| \frac{P(j\omega)C(j\omega)}{1 + P(j\omega)C(j\omega)} \right| |N(j\omega)|.$$

CHAPTER 16. FREQUENCY DOMAIN SPECIFICATIONS

The objective is to attenuate noise signals $n(t)$ with frequency content above ω_{no} so that the error due to the noise signal satisfies

$$|e(t)| \leq \gamma_n |n(t)|.$$

Therefore, the controller $C(s)$ needs to be designed so that

$$\left| \frac{P(j\omega)C(j\omega)}{1 + P(j\omega)C(j\omega)} \right| \leq \gamma_n.$$

If $|P(j\omega)C(j\omega)| \ll 1$, then a sufficient condition is that

$$|P(j\omega)C(j\omega)| \leq \gamma_n,$$

or equivalently

$$20 \log_{10} |P(j\omega)C(j\omega)| \leq 20 \log_{10} \gamma_n, \quad (16.6)$$

which needs to be satisfied for all $\omega \geq \omega_{no}$. On a Bode plot, the constraint represented by Equation (16.6) is shown graphically in Figure 16.3.

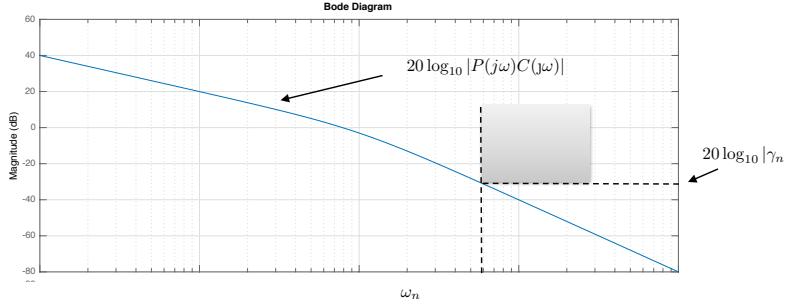


Figure 16.3: Bode plot representation of the loopshaping constraint corresponding to attenuating the noise $n(t)$ so that the corresponding error satisfies $|e(t)| \leq \gamma_n |n(t)|$.

Alternatively, from the Bode plot of PC we can compute the noise attenuation characteristics of the closed loop system. For example, suppose that on the Bode plot

$$20 \log_{10} |P(j\omega)C(j\omega)| < -B_n \text{ dB}$$

for all $\omega > \omega_{no}$, then

$$|P(j\omega)C(j\omega)| < 10^{-B_n/20},$$

for all $\omega > \omega_{no}$. If $P(j\omega)C(j\omega) \ll 1$ for all $\omega > \omega_{no}$, then

$$\left| \frac{P(j\omega)C(j\omega)}{1 + P(j\omega)C(j\omega)} \right| \lesssim 10^{-B_n/20},$$

which implies that

$$|E(j\omega)| \leq 10^{-B_n/20} |N(j\omega)|$$

for all $\omega > \omega_{no}$, or in other words that

$$|e(t)| \leq 10^{-B_n/20} |n(t)|,$$

for all $n(t)$ with frequency content above ω_{no} , which implies that

$$\gamma_n = 10^{-B_n/20}.$$

16.1.3 Reject Output Disturbances

The transfer function from the output disturbance signals $d_{out}(t)$ to the error $e(t)$ is

$$E(s) = \frac{1}{1 + P(s)C(s)} D_{out}(s).$$

Therefore

$$|E(j\omega)| \leq \left| \frac{1}{1 + P(j\omega)C(j\omega)} \right| |D_{out}(j\omega)|.$$

The objective is to attenuate output disturbance signals $d_{out}(t)$ with frequency content below $\omega_{d_{out}}$ so that the error due to the output disturbance signal satisfies

$$|e(t)| \leq \gamma_{d_{out}} |d_{out}(t)|.$$

Therefore, the controller $C(s)$ needs to be designed so that

$$\left| \frac{1}{1 + P(j\omega)C(j\omega)} \right| \leq \gamma_{d_{out}},$$

or equivalently so that

$$|1 + P(j\omega)C(j\omega)| \geq \frac{1}{\gamma_{d_{out}}}.$$

If $|P(j\omega)C(j\omega)| \gg 1$, then a sufficient condition is that

$$|P(j\omega)C(j\omega)| \geq \frac{1}{\gamma_{d_{out}}},$$

or equivalently

$$20 \log_{10} |P(j\omega)C(j\omega)| \geq 20 \log_{10} 1/\gamma_{d_{out}}, \quad (16.7)$$

which needs to be satisfied for all $\omega \leq \omega_{d_{out}}$. On a Bode plot, the constraint represented by Equation (16.7) is shown graphically in Figure 16.4.

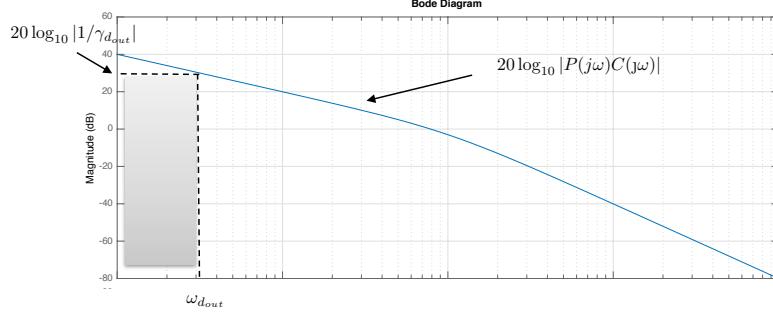


Figure 16.4: Bode plot representation of the loopshaping constraint corresponding to rejecting the output disturbance $d_{out}(t)$ so that the corresponding error satisfies $|e(t)| \leq \gamma_{d_{out}} |d_{out}(t)|$.

Alternatively, from the Bode plot of PC we can compute the output disturbance rejection characteristics of the closed loop system. For example, suppose that on the Bode plot

$$20 \log_{10} |P(j\omega)C(j\omega)| > B_{d_{out}} \text{ dB}$$

for all $\omega < \omega_{d_{out}}$, then

$$|P(j\omega)C(j\omega)| > 10^{B_{d_{out}}/20},$$

for all $\omega < \omega_{d_{out}}$. If $|P(j\omega)C(j\omega)| \gg 1$ for all $\omega < \omega_{d_{out}}$, then

$$\left| \frac{1}{1 + P(j\omega)C(j\omega)} \right| \lesssim 10^{-B_{d_{out}}/20},$$

which implies that

$$|E(j\omega)| \leq 10^{-B_{d_{out}}/20} |D_{out}(j\omega)|$$

for all $\omega < \omega_{d_{out}}$, or in other words that

$$|e(t)| \leq 10^{-B_{d_{out}}/20} |d_{out}(t)|,$$

for all $d_{out}(t)$ with frequency content below $\omega_{d_{out}}$, which implies that

$$\gamma_{d_{out}} = 10^{-B_{d_{out}}/20}.$$

16.1.4 Reject Input Disturbances

The transfer function from the input disturbance signals $d_{in}(t)$ to the error $e(t)$ is

$$E(s) = \frac{P(s)}{1 + P(s)C(s)} D_{in}(s).$$

CHAPTER 16. FREQUENCY DOMAIN SPECIFICATIONS

Therefore

$$|E(j\omega)| \leq \left| \frac{P(j\omega)}{1 + P(j\omega)C(j\omega)} \right| |D_{in}(j\omega)|.$$

The objective is to attenuate input disturbance signals $d_{in}(t)$ with frequency content below $\omega_{d_{in}}$ so that the error due to the input disturbance signal satisfies

$$|e(t)| \leq \gamma_{d_{in}} |d_{in}(t)|.$$

Therefore, the controller $C(s)$ needs to be designed so that

$$\left| \frac{P(j\omega)}{1 + P(j\omega)C(j\omega)} \right| \leq \gamma_{d_{in}},$$

or equivalently so that

$$\frac{|1 + P(j\omega)C(j\omega)|}{|P(j\omega)|} \geq \frac{1}{\gamma_{d_{in}}}.$$

If $|P(j\omega)C(j\omega)| \gg 1$, then a sufficient condition is that

$$\frac{|P(j\omega)C(j\omega)|}{|P(j\omega)|} \geq \frac{1}{\gamma_{d_{in}}},$$

or equivalently

$$20 \log_{10} |P(j\omega)C(j\omega)| - 20 \log_{10} |P(j\omega)| \geq 20 \log_{10} 1/\gamma_{d_{in}}, \quad (16.8)$$

which needs to be satisfied for all $\omega \leq \omega_{d_{in}}$. On a Bode plot, the constraint represented by Equation (16.8) is shown graphically in Figure 16.5, where it can be seen that the loop gain $|PC|$ must be above the magnitude of the plant $|P|$ by the specified amount.

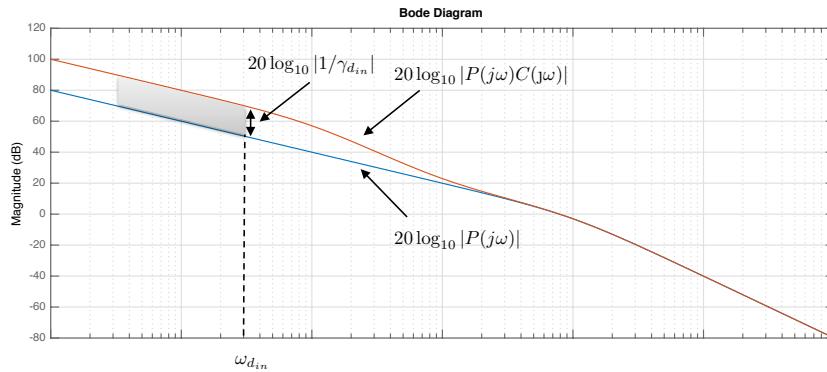


Figure 16.5: Bode plot representation of the loopshaping constraint corresponding to rejecting the input disturbance $d_{in}(t)$ so that the corresponding error satisfies $|e(t)| \leq \gamma_{d_{in}} |d_{in}(t)|$.

CHAPTER 16. FREQUENCY DOMAIN SPECIFICATIONS

Alternatively, from the Bode plot of PC we can compute the input disturbance rejection characteristics of the closed loop system. For example, suppose that on the Bode plot

$$20 \log_{10} |P(j\omega)C(j\omega)| - 20 \log_{10} |P(j\omega)| > B_{d_{in}} \text{ dB}$$

for all $\omega < \omega_{d_{in}}$, then

$$\frac{|P(j\omega)C(j\omega)|}{|P(j\omega)|} > 10^{B_{d_{in}}/20},$$

for all $\omega < \omega_{d_{in}}$. If $|P(j\omega)C(j\omega)| \gg 1$ for all $\omega < \omega_{d_{in}}$, then

$$\left| \frac{P(j\omega)}{1 + P(j\omega)C(j\omega)} \right| \lesssim 10^{-B_{d_{in}}/20},$$

which implies that

$$|E(j\omega)| \leq 10^{-B_{d_{in}}/20} |D_{in}(j\omega)|$$

for all $\omega < \omega_{d_{in}}$, or in other words that

$$|e(t)| \leq 10^{-B_{d_{in}}/20} |d_{in}(t)|, \quad (16.9)$$

for all $d_{in}(t)$ with frequency content below $\omega_{d_{in}}$, which implies that

$$\gamma_{d_{in}} = 10^{-B_{d_{in}}/20}.$$

16.1.5 Frequency Response Characterization of System Type

Recall that the loop gain $P(s)C(s)$ is said to be

- Type 0 if there are no free integrators,
- Type 1 if there is one free integrator,
- Type 2 if there are two free integrators, etc..

Type 0 System

Note that for a type 0 system

$$\begin{aligned} \lim_{s \rightarrow 0} P(s)C(s) &= \text{constant} \\ \implies \lim_{\omega \rightarrow 0} |P(j\omega)C(j\omega)| &= \text{constant} \\ \implies \lim_{\omega \rightarrow 0} 20 \log_{10} |P(j\omega)C(j\omega)| &= \text{constant}. \end{aligned}$$

In other words, at low frequency, the Bode magnitude plot is as shown in Figure 16.6.

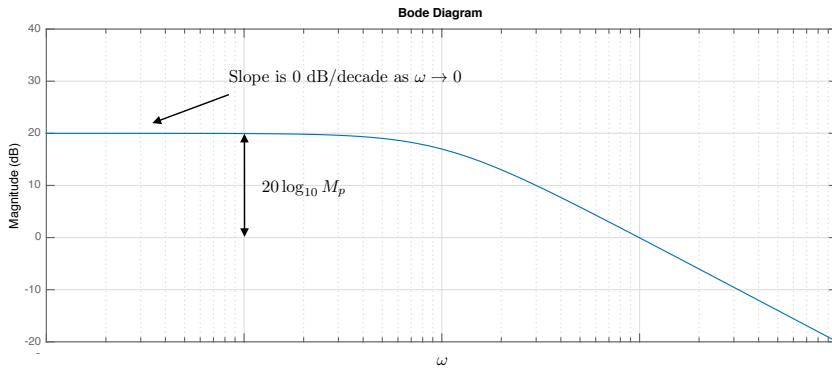


Figure 16.6: Bode plot of a type 0 system.

Recall that for a step input to a type 0 system where $R(s) = A/s$, the steady state error is

$$e_{ss} = \lim_{t \rightarrow 0} e(t) = \lim_{s \rightarrow 0} sE(s) = \lim_{s \rightarrow 0} \frac{A}{1 + P(s)C(s)} = \frac{A}{1 + M_p},$$

where

$$M_p = \lim_{s \rightarrow 0} P(s)C(s) = \lim_{\omega \rightarrow 0} |P(j\omega)C(j\omega)|.$$

As shown in Figure 16.6, $20 \log_{10} M_p$ is the constant limit of the loop gain as $\omega \rightarrow 0$.

Therefore, from the Bode plot of PC for a type 0 system we can compute the tracking error for a step input of size A . Suppose that on the Bode plot

$$\lim_{\omega \rightarrow 0} 20 \log_{10} |P(j\omega)C(j\omega)| = B_0 \text{ dB.}$$

Then

$$M_p = 10^{B_0/20},$$

and

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{1}{1 + M_p} A. \quad (16.10)$$

Type 1 System

For a type 1 system since $P(s)C(s)$ has one free integrator

$$\begin{aligned} & \lim_{s \rightarrow 0} sP(s)C(s) = \text{constant} \\ \implies & \lim_{\omega \rightarrow 0} |(j\omega)P(j\omega)C(j\omega)| = \text{constant} \\ \implies & \lim_{\omega \rightarrow 0} \frac{|P(j\omega)C(j\omega)|}{\left| \frac{1}{j\omega} \right|} = \text{constant} \\ \implies & \lim_{\omega \rightarrow 0} \left[20 \log_{10} |P(j\omega)C(j\omega)| - 20 \log_{10} \left| \frac{1}{j\omega} \right| \right] = \text{constant}. \end{aligned}$$

CHAPTER 16. FREQUENCY DOMAIN SPECIFICATIONS

In other words, at low frequency, the Bode magnitude plot of $20 \log_{10} |P(j\omega)C(j\omega)|$ has a slope of -20 dB/decade as $\omega \rightarrow 0$, as shown in Figure 16.7.

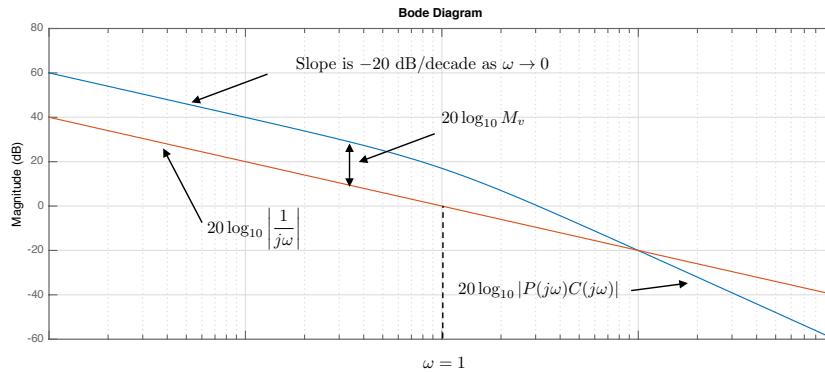


Figure 16.7: Bode plot of a type 1 system.

Recall that for a ramp input to a type 1 system where $R(s) = A/s^2$, the steady state error is

$$e_{ss} = \lim_{t \rightarrow 0} e(t) = \lim_{s \rightarrow 0} sE(s) = \lim_{s \rightarrow 0} \frac{A}{s + sP(s)C(s)} = \frac{A}{M_v},$$

where

$$M_v = \lim_{s \rightarrow 0} sP(s)C(s) = \lim_{\omega \rightarrow 0} |(j\omega)P(j\omega)C(j\omega)|.$$

As shown in Figure 16.7, $20 \log_{10} M_v$ is the amount that the loop gain exceeds the Bode plot of $20 \log_{10} |1/j\omega|$ as $\omega \rightarrow 0$.

Therefore, from the Bode plot of PC for a type 1 system we can compute the tracking error for a ramp input with slope A . Suppose that on the Bode plot

$$\lim_{\omega \rightarrow 0} \left[20 \log_{10} |P(j\omega)C(j\omega)| - 20 \log_{10} \left| \frac{1}{j\omega} \right| \right] = B_1 \text{ dB},$$

then

$$M_v = 10^{B_1/20},$$

and

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{1}{M_v} A. \quad (16.11)$$

Type 2 System

For a type 2 system where $P(s)C(s)$ has two free integrators we have

$$\begin{aligned} & \lim_{s \rightarrow 0} s^2 P(s)C(s) = \text{constant} \\ \implies & \lim_{\omega \rightarrow 0} |(j\omega)^2 P(j\omega)C(j\omega)| = \text{constant} \\ \implies & \lim_{\omega \rightarrow 0} \frac{|P(j\omega)C(j\omega)|}{\left| \frac{1}{j\omega} \right|^2} = \text{constant} \\ \implies & \lim_{\omega \rightarrow 0} \left[20 \log_{10} |P(j\omega)C(j\omega)| - 20 \log_{10} \left| \frac{1}{j\omega} \right|^2 \right] = \text{constant}. \end{aligned}$$

In other words, at low frequency, the Bode magnitude plot of $20 \log_{10} |P(j\omega)C(j\omega)|$ has a slope of -40 dB/decade as $\omega \rightarrow 0$, as shown in Figure 16.8.

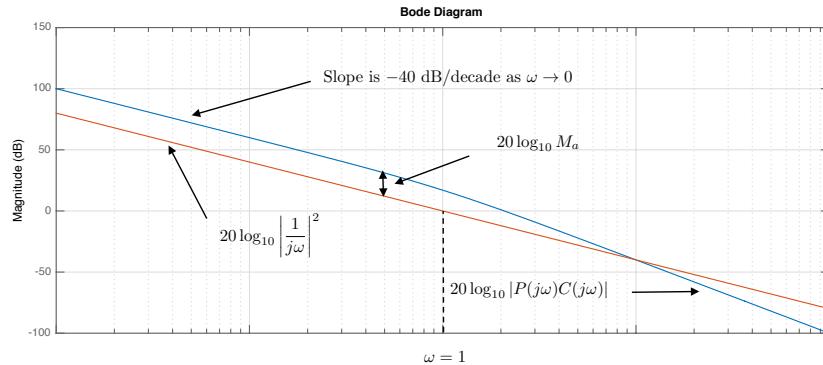


Figure 16.8: Bode plot of a type 2 system.

Recall that for a parabolic input to a type 2 system where $R(s) = A/s^3$, the steady state error is

$$e_{ss} = \lim_{t \rightarrow 0} e(t) = \lim_{s \rightarrow 0} sE(s) = \lim_{s \rightarrow 0} \frac{A}{s^2 + s^2 P(s)C(s)} = \frac{A}{M_a},$$

where

$$M_a = \lim_{s \rightarrow 0} s^2 P(s)C(s) = \lim_{\omega \rightarrow 0} |(j\omega)^2 P(j\omega)C(j\omega)|.$$

As shown in Figure 16.8, $20 \log_{10} M_a$ is the amount that the loop gain exceeds the Bode plot of $20 \log_{10} |1/j\omega|^2$ as $\omega \rightarrow 0$.

Therefore, from the Bode plot of PC for a type 2 system we can compute the tracking error for a parabola input with curvature A . Suppose that on the Bode plot

$$\lim_{\omega \rightarrow 0} \left[20 \log_{10} |P(j\omega)C(j\omega)| - 20 \log_{10} \left| \frac{1}{j\omega} \right|^2 \right] = B_2 \text{ dB}.$$

Then

$$M_a = 10^{B_2/20},$$

and

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{1}{M_a} A. \quad (16.12)$$

16.2 Design Study A. Single Link Robot Arm



Homework Problem A.16

For the single link robot arm, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, (2) the plant under PID control, using the control gains calculated in HW A.10.

- (a) To what percent error can the closed loop system under PID control track the desired input if all of the frequency content of $\theta_r(t)$ is below $\omega_r = 0.4$ radians per second?
- (b) If the reference input is $\theta_r(t) = 5t^2$ for $t \geq 0$, what will be the steady state tracking error to this input when using PID control?
- (c) If all of the frequency content of the input disturbance $d_{in}(t)$ is below $\omega_{d_{in}} = 0.01$ radians per second, what percentage of the input disturbance shows up in the output θ under PID control?
- (d) If all of the frequency content of the noise $n(t)$ is greater than $\omega_{no} = 100$ radians per second, what percentage of the noise shows up in the output signal θ ?

Solution

- (a) The Bode plot of the plant $P(s)$, and the loop gain with PID control $P(s)C_{PID}(s)$ is shown in Figure 16.9.

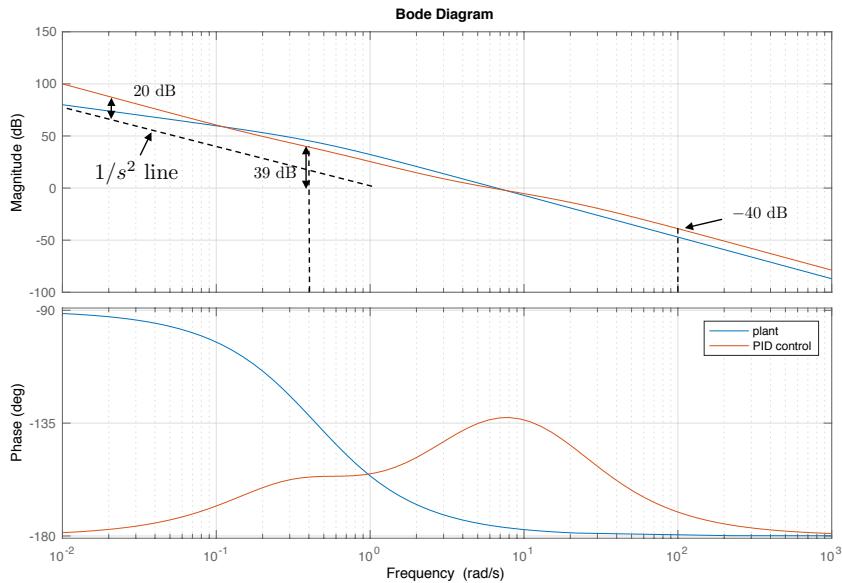


Figure 16.9: Bode plot for single link arm, plant only and under PID control.

From Figure 16.9 we see that below $\omega_r = 0.4$ rad/sec, the loop gain is above $B_r = 39$ dB. Therefore, from Equation (16.5) we have that

$$|e(t)| \leq \gamma_r |r(t)|,$$

where $\gamma_r = 10^{-39/20} = 0.1059$, which implies that the tracking error will be 10.6% of the magnitude of the input.

(b) Suppose now that the desired reference input is $\theta^d(t) = 5t^2$. Under PID control, the slope of the loop gain as $\omega \rightarrow 0$ is -40 dB/dec, which implies that the system is type 2 and will track a step and a ramp with zero steady state error. For a parabola, there will be a finite error. Since the loop gain under PID control is $B_2 = 20$ dB above the $1/s^2$ line, from Equation (??) the steady state error satisfies

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{A}{M_a} = 5 \cdot 10^{-20/20} = 0.5.$$

(c) If the input disturbance is below $\omega_{d_{in}} = 0.01$ rad/s, then the difference between the loop gain and the plant is $B_{d_{in}} = 20$ dB at $\omega_{d_{in}} = 0.01$ rad/s, therefore, from Equation (16.9) we see that

$$\gamma_{d_{in}} = 10^{-20/20} = 0.1,$$

implying that 10% of the input disturbance will show up in the output.

(d) For noise greater than $\omega_{no} = 100$ rad/sec, we see from Figure ?? that $B_n = 40$ dB. Therefore, $\gamma_n = 10^{-40/20} = 0.01$ which implies that 1% of the noise will show up in the output signal.²⁰¹

16.3 Design Study B. Inverted Pendulum



Homework Problem B.16

For the inner loop of the inverted pendulum, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, and (2) the plant under PD control, using the control gains calculated in Homework B.10.

- (a) To what percent error can the closed loop system track the desired input if all of the frequency content of $\theta_r(t)$ is below $\omega_r = 1.0$ radians per second?
- (b) If all of the frequency content of the sensor noise on the inner $n(t)$ is greater than $\omega_{no} = 200$ radians per second, what percentage of the noise shows up in the output signal θ ?

For the outer loop of the inverted pendulum, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for the plant, and the plant under PID, using the control gains calculated in Homework B.10.

- (c) If the reference signal $y_r(t)$ has frequency content below $\omega_r = 0.001$ radians/second, what is the tracking error under PID control if $|r(t)| \leq 50$?

Solution

- (a) The Bode plot of the plant $P(s)$, and the loop gain with PD control $P(s)C_{PD}(s)$ are shown in Figure 16.10.

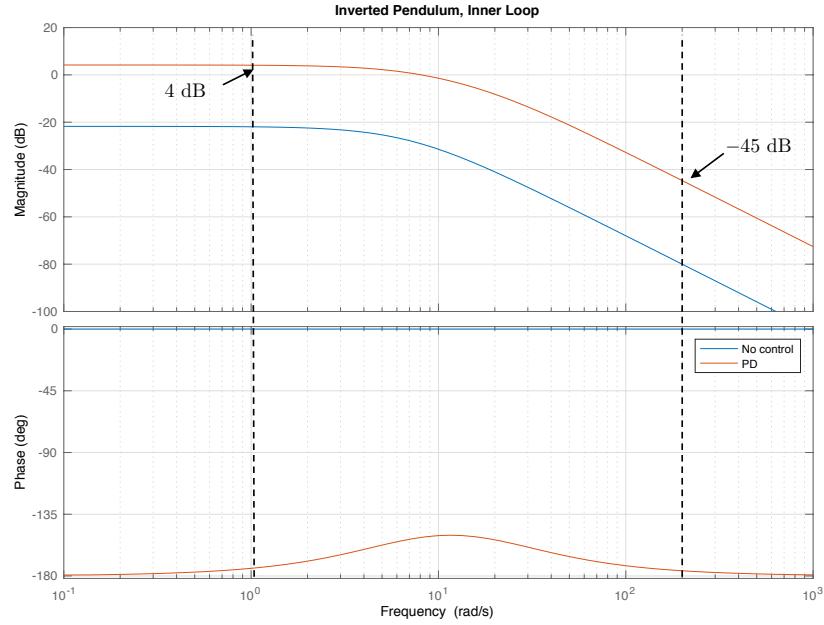


Figure 16.10: Bode plot for inner loop of the inverted pendulum, plant only, and under PD control.

From Figure 16.10 we see that below $\omega_r = 1.0$ rad/sec, the loop gain is above $B_r = 4$ dB. Therefore, from Equation (16.5) we have that

$$|e(t)| \leq \gamma_r |r(t)|,$$

where $\gamma_r = 10^{-4/20} = 0.63$, which implies that the tracking error will be 63% of the magnitude of the input.

(b) For noise greater than $\omega_{no} = 200$ rad/sec, we see from Figure 16.10 that $B_n = 45$ dB. Therefore, $\gamma_n = 10^{-45/20} = 0.0056$ which implies that 0.56% of the noise will show up in the output signal.

(c) The Bode plot of the outer loop $P(s)$, and the loop gain with PID control $P(s)C_{PID}(s)$ is shown in Figure 16.11.

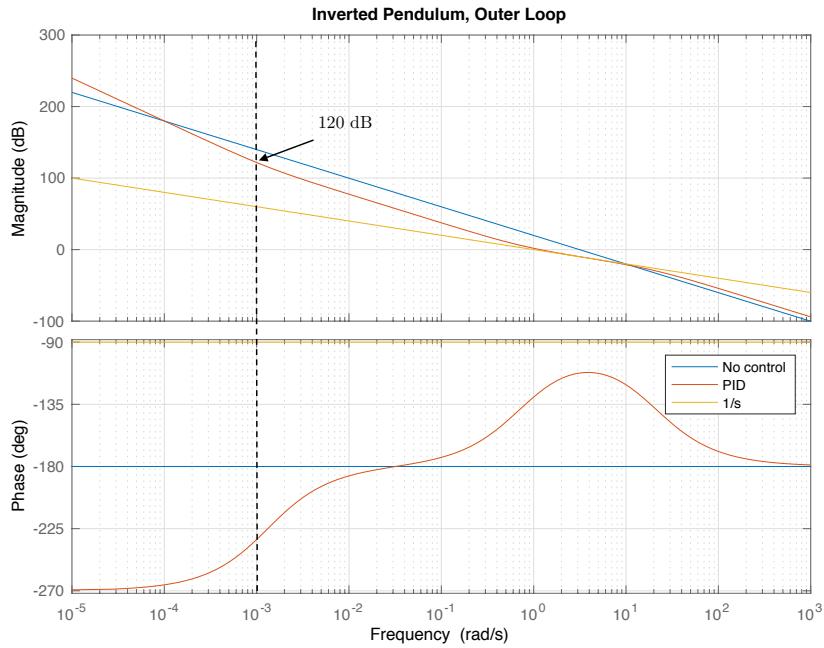


Figure 16.11: Bode plot for outer loop of the inverted pendulum, plant only, and under PID control.

From Figure 16.11 it can be seen that the loopgain under PID control for the outer loop is above $B_r = 120$ dB for all $\omega < \omega_r = 0.001$ radians/second. Therefore, from Equation (16.5) the tracking error satisfies

$$|e(t)| \leq \gamma_r |r(t)|,$$

where $\gamma_r = 10^{-120/20} = 1.0e-06$. Therefore, if $|r(t)| \leq 50$, then $|e(t)| \leq 50e-06$.

16.4 Design Study C. Satellite Attitude Control



Homework Problem C.16

For the inner loop of the satellite attitude control, use the Matlab bode command to create a graph that simultaneously displays the Bode plots for (1) the plant, and (2) the plant under PD control, using the control gains calculated in Homework C.10.

- (a) To what percent error can the closed loop system under PD control track a step in $\theta_r(t)$ of 20 degrees?

- (b) If the input disturbance has frequency content below $\omega_{d_{in}} = 0.1$ radians per second, what percentage of the input disturbance appears in the output.

For the outer loop of the satellite attitude control, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, (2) the plant under PID control using the control gains calculated in Homework C.10.

- (c) If all of the frequency content of the noise $n(t)$ is greater than $\omega_{no} = 10$ radians per second, what percentage of the noise shows up in the output signal ϕ , using PID control?

Solution

- (a) The Bode plot of the plant $P(s)$, and the loop gain with PD control $P(s)C_{PD}(s)$ are shown in Figure 16.12.

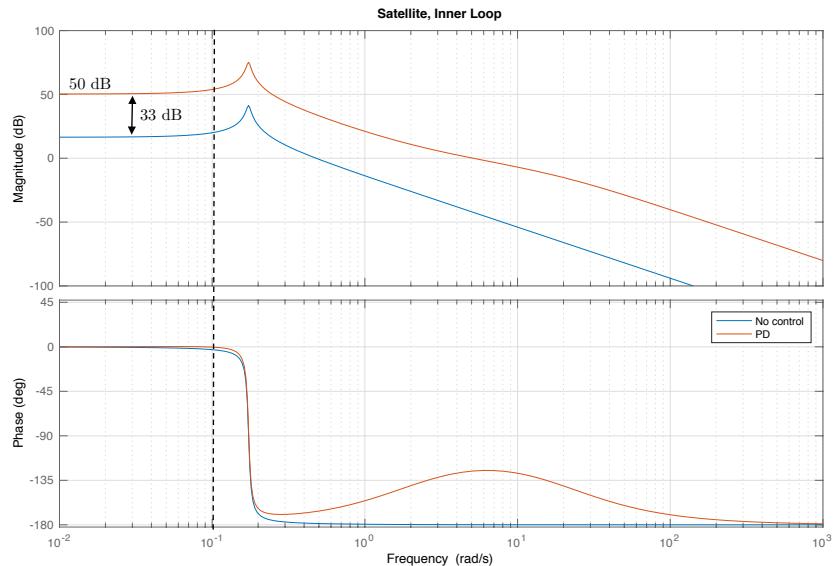


Figure 16.12: Bode plot for inner loop of the satellite attitude control, plant only, and under PD control.

From Figure 16.12 it can be seen that under PD control the inner loop is type 0 and that the loop gain as $\omega \rightarrow 0$ is $B_0 = 50$ dB. Therefore from Equation (16.10) the steady state error to a step of $\theta_d = 20$ degrees is

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{1}{1 + M_p} A = \frac{20}{1 + 10^{50/20}} = 0.0630 \text{ degrees.}$$

(b) If the input disturbance is below $\omega_{d_{in}} = 0.1$ rad/s, then the difference between the loop gain and the plant is $B_{d_{in}} = 33$ dB, therefore, from Equation (16.9) we see that

$$\gamma_{d_{in}} = 10^{-33/20} = 0.0224,$$

implying that 2.2% of the input disturbance will show up in the output.

(c) The Bode plot of outer loop $P_{out}(s)$, and the loop gain with PI control $P_{out}(s)C_{PI}(s)$ are shown in Figure 16.13.

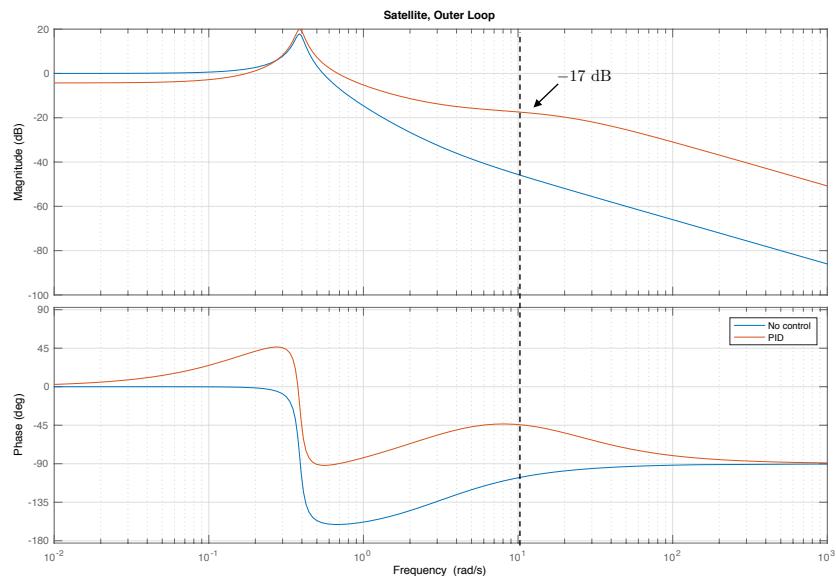


Figure 16.13: Bode plot for outer loop of the satellite attitude control, plant only, P control, and PI control.

For noise greater than $\omega_{no} = 1$ rad/sec, we see from Figure 16.13 that $B_n = 17$ dB. Therefore, $\gamma_n = 10^{-17/20} = 0.1413$ which implies that 14% of the noise will show up in the output signal.

Notes and References

Chapter 17

Stability and Robustness Margins

17.1 Theory

17.1.1 Phase and Gain Margins

In the previous chapter we did not talk about closed loop stability. To achieve the objectives discussed in Chapter 16, the closed loop system must be stable. Recall that tracking performance and disturbance rejection characteristics are determined by the loopshape at low frequencies when $|P(j\omega)C(j\omega)| \gg 1$. Similarly, noise attenuation properties are determined by the loopshape at high frequencies when $|P(j\omega)C(j\omega)| \ll 1$. In both cases we ignored the phase of $P(j\omega)C(j\omega)$. It turns out that stability of the closed-loop system is determined by the phase $P(j\omega)C(j\omega)$ when $P(j\omega)C(j\omega) = 1$, as depicted in Figure 17.1.

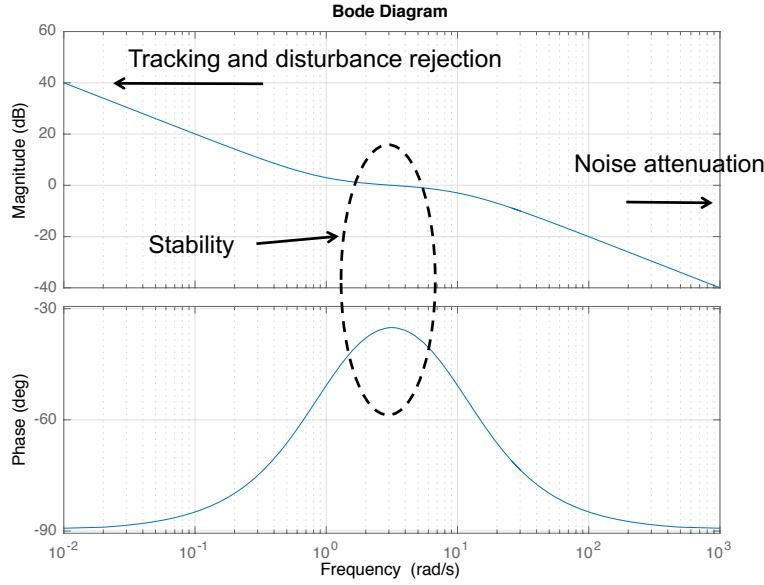


Figure 17.1: The shape of the magnitude plot primarily determines the tracking and disturbance rejection characteristics at low frequencies, and the noise attenuation at high frequencies. Stability is primarily determined by the phase at the crossover frequency.

Define the *crossover frequency* ω_{co} to be the frequency where $|P(j\omega_{co})C(j\omega_{co})| = 1$, or equivalently where $20 \log_{10} |P(j\omega_{co})C(j\omega_{co})| = 0$ dB. The phase $\angle P(j\omega_{co})C(j\omega_{co})$ at the crossover frequency plays a critical role in determining the stability of the system. To better understand why this is the case, consider the transfer functions from R , D_{in} , D_{out} , and N , to E in Equation (16.3). The denominator in each transfer function is $1 + P(s)C(s)$. Therefore, the closed loop poles are given by the closed loop characteristic equation

$$1 + P(s)C(s) = 0,$$

or equivalently where

$$P(s)C(s) = -1.$$

Recall that for any s , $P(s)C(s)$ is a complex number with magnitude and phase. Therefore, given any s_0 we can draw $1 + P(s_0)C(s_0)$ in the complex plane as shown in Figure 17.2.

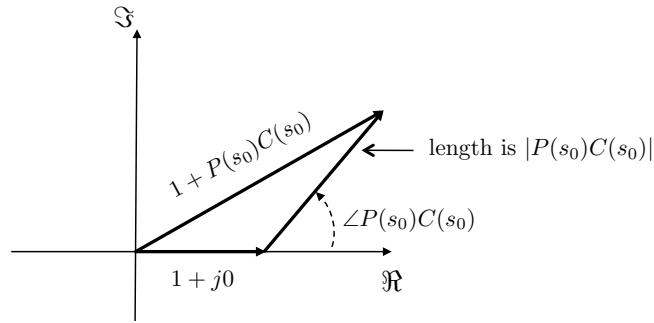


Figure 17.2: The plot of $1 + P(s_0)C(s_0)$ in the complex plane, is the sum of the complex number $1 + j0$ and $P(s_0)C(s_0)$.

Since the physical response of the system is determined by $P(s)C(s)$ evaluated on the $j\omega$ -axis, there are stability problems if there exists and ω_0 such that $P(j\omega_0)C(j\omega_0) = -1$, or in other words when

$$|P(j\omega_0)C(j\omega_0)| = 1 \quad \text{and} \quad \angle P(j\omega_0)C(j\omega_0) = -180^\circ,$$

as shown in Figure 17.3.

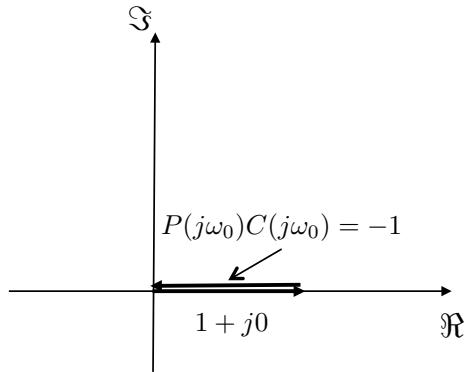


Figure 17.3: The plot of $1 + P(j\omega_0)C(j\omega_0)$ in the complex plane when $P(j\omega_0)C(j\omega_0) = -1$.

Since the magnitude of the loop gain is equal to one at the crossover frequency ω_{co} , it turns out that the quality of stability is determined by the phase of the loop gain at the crossover frequency. Define the *phase margin PM* as

$$PM = \angle P(j\omega_{co})C(j\omega_{co}) - 180^\circ.$$

Figure 17.4 depicts the phase margin on the Bode plot.

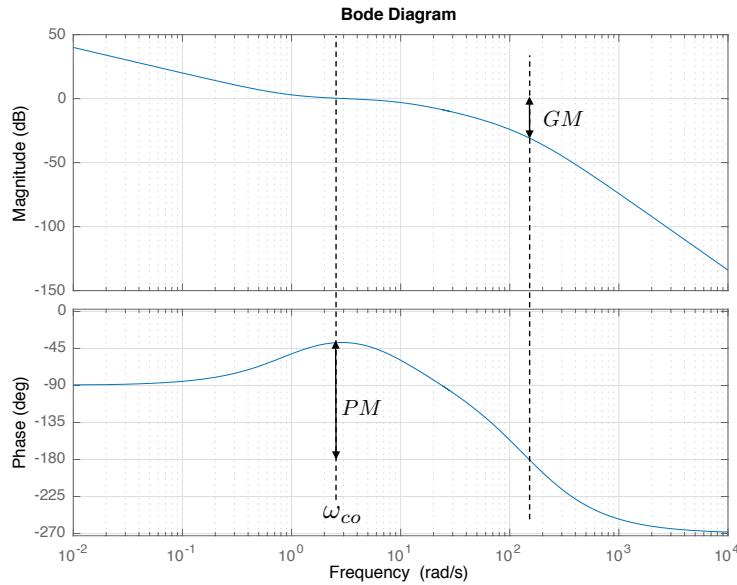


Figure 17.4: The phase margin PM is the difference between the phase of $P(j\omega)C(j\omega)$ and -180 degrees at the crossover frequency ω_{co} . The gain margin GM is the magnitude of $P(j\omega)C(j\omega)$ when the phase crosses -180 degrees.

Another way to understand the phase margin is shown in Figure 17.5. The phase margin indicates the angular distance that $P(j\omega_{co})C(j\omega_{co})$ is from -1 .

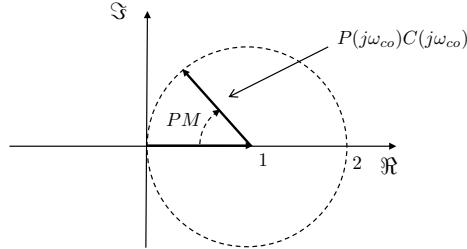


Figure 17.5: Alternative graphical view of the phase margin PM .

A related quantity is the *Gain Margin GM*, which measures how far the magnitude $|P(j\omega)C(j\omega)|$ is from unity when the phase $\angle P(j\omega)C(j\omega) = -180$ degrees. The gain margin is depicted on the Bode plot in Figure 17.4. An alternative view is shown in Figure 17.6.

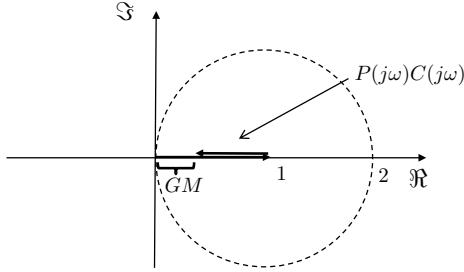


Figure 17.6: Alternative graphical view of the gain margin GM .

A good control design will have large phase and gain margins. A phase margin of $PM = 60$ degrees is considered to be very good and is optimal for many applications. A gain margin of $GM = 10$ dB would be considered very good for most applications.

17.1.2 Open and Closed Loop Frequency Response

From Equation (16.1), the transfer function from the reference $R(s)$ to the output $Y(s)$ is

$$Y(s) = \frac{P(s)C(s)}{1 + P(s)C(s)}R(s)$$

Note that

- when $|P(j\omega)C(j\omega)| \gg 1$, then $\left| \frac{P(j\omega)C(j\omega)}{1+P(j\omega)C(j\omega)} \right| \approx 1$,
- when $|P(j\omega)C(j\omega)| \ll 1$, then $\left| \frac{P(j\omega)C(j\omega)}{1+P(j\omega)C(j\omega)} \right| \approx |P(j\omega)C(j\omega)|$.

When $|P(j\omega)C(j\omega)| = 1$, i.e., when $\omega = \omega_{co}$, then $\left| \frac{P(j\omega_{co})C(j\omega_{co})}{1+P(j\omega_{co})C(j\omega_{co})} \right|$ is a complex number divided by a complex number, and the magnitude will depend on the phase of the $P(j\omega_{co})C(j\omega_{co})$, or in other words the phase margin PM . As shown in Figures 17.5, PM will determine the magnitude of $|1 + P(j\omega_{co})C(j\omega_{co})|$. When PM is small, $|1 + P(j\omega_{co})C(j\omega_{co})|$ will be small and $\left| \frac{P(j\omega_{co})C(j\omega_{co})}{1+P(j\omega_{co})C(j\omega_{co})} \right|$ will be larger than one. When PM is larger than 90 degrees, $|1 + P(j\omega_{co})C(j\omega_{co})|$ will be larger than $|P(j\omega_{co})C(j\omega_{co})|$ and $\left| \frac{P(j\omega_{co})C(j\omega_{co})}{1+P(j\omega_{co})C(j\omega_{co})} \right|$ will be less than one.

Figure 17.7 shows the closed loop Bode plot superimposed on the open loop Bode plot for different values of the phase margin. Note that when the phase margin is $PM = 60$ degrees, the closed loop frequency response looks very similar to the closed-loop frequency response for a second order system when $\zeta = 0.707$, as shown in Figure 15.9. Phase margins smaller than 60 degrees have a peaking response similar to second order system with $\zeta < 0.707$. Note also that the bandwidth of the closed loop system is approximately the crossover frequency of the open loop system. In general, the crossover frequency ω_{co} plays

CHAPTER 17. STABILITY AND ROBUSTNESS MARGINS

a similar role for general systems that the natural frequency ω_n plays for second order systems. Figure 17.7 also shows the corresponding step response for the

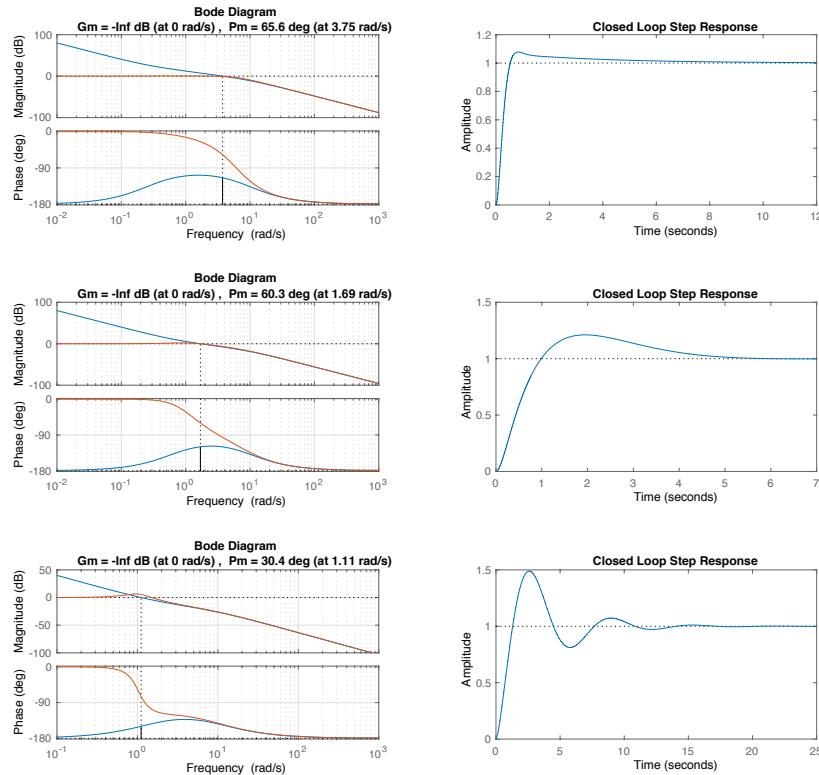


Figure 17.7: Closed loop Bode plot superimposed on the open loop Bode plot, together with the closed loop step response, for different phase margins.

closed loop system. The step response looks similar to the step response for second order systems shown in Figure 8.3, where again, the step response for $PM = 60$ degrees in Figure 17.7 is roughly equivalent to the step response for $\zeta = 0.707$ in Figure 8.4.

For a given reference signal, the size of the control signal $u(t)$ is determined by the crossover frequency. A large crossover frequency will result in a larger control signal $u(t)$. Figure 17.8 shows the open and closed loop transfer functions for a plant equal to $P(s) = 1/s(s+1)$ and for three different controllers that are tuned for equivalent phase margins but with cross over frequency at 2, 20, and 200 respectively. The corresponding control signal when the reference is a unit step is also shown in Figure 17.8. Note that as ω_{co} increases, the magnitude of the control signal increases and the speed of the response also increases. In fact, the integral of the control signals will be the same for each ω_{co} . Therefore,

CHAPTER 17. STABILITY AND ROBUSTNESS MARGINS

saturation constraints on $u(t)$ will necessarily limit the size of the cross over frequency ω_{co} .

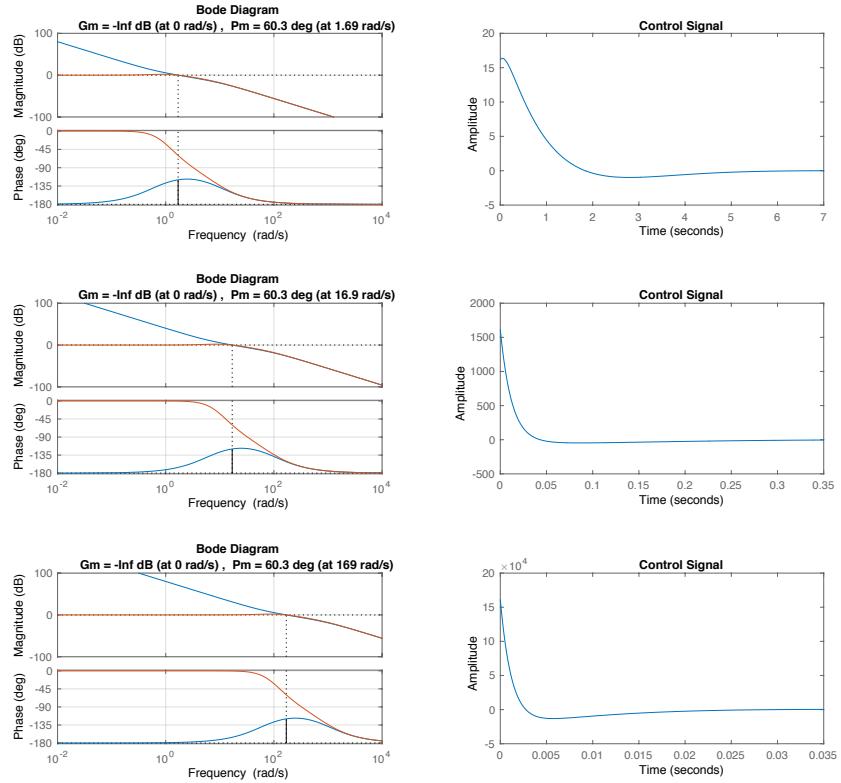


Figure 17.8: Closed loop Bode plot superimposed on the open loop Bode plot, together with the corresponding control signal, for different cross-over frequencies.

17.1.3 Bode Phase Gain Relationship

In Bode's original work, he proved the following theorem that relates the slope of the magnitude plot to the phase.

Bode's Phase-Gain Theorem. *For any stable, minimum phase (i.e., zeros in the LHP), linear time-invariant system $G(s)$, the phase of $G(j\omega)$ is uniquely related to $20 \log_{10} |G(j\omega)|$ by*

$$\angle G(j\omega_0) = \frac{1}{\pi} \int_{-\infty}^{\infty} \left(\frac{dM}{du} \right) W(u) du, \quad (17.1)$$

where

$$M = 20 \log_{10} |G(j\omega)|,$$

$$u = \log_{10} \left| \frac{\omega}{\omega_0} \right|,$$

$$W(u) = \log_{10} \left(\coth \frac{|u|}{2} \right).$$

The term $\frac{dM}{du}$ in Equation (17.1) is the slope of the magnitude of $G(j\omega)$ on the Bode plot. A plot of $W(u)$ is shown in Figure 17.9, where it can be seen that $W(u)$ heavily weights the slope of the Bode magnitude around ω_0 , but also include bleed-over from the slope for roughly a decade before and after ω_0 . This implies that the phase of $P(j\omega)C(j\omega)$ at $\omega = \omega_0$ is determined by the slope of the Bode magnitude plot in a region about ω_0 .

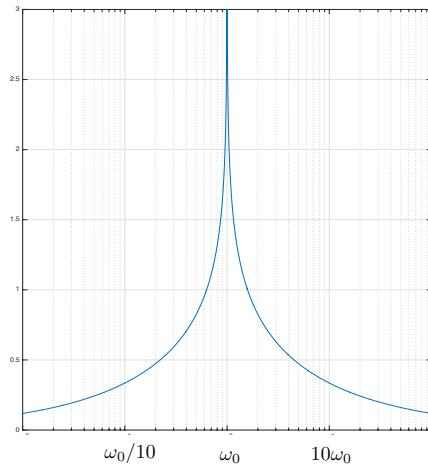


Figure 17.9: The weighting function $W(u)$ in the Bode gain-phase relationship heavily weights the slope around ω_0 .

Recall that the phase for an integrator $1/s$ is -90 degrees, and that the phase for a differentiator is 90 degrees. Also recall that the phase for a constant is 0 degrees. Therefore, we can approximate Equation (17.1) as follows:

CHAPTER 17. STABILITY AND ROBUSTNESS MARGINS

- Slope of $20 \log_{10} |G(j\omega_0)| \approx +20 \text{ dB/dec} \implies \angle G(j\omega_0) \approx +90 \text{ deg}$,
- Slope of $20 \log_{10} |G(j\omega_0)| \approx 0 \text{ dB/dec} \implies \angle G(j\omega_0) \approx 0 \text{ deg}$,
- Slope of $20 \log_{10} |G(j\omega_0)| \approx -20 \text{ dB/dec} \implies \angle G(j\omega_0) \approx -90 \text{ deg}$,
- Slope of $20 \log_{10} |G(j\omega_0)| \approx -40 \text{ dB/dec} \implies \angle G(j\omega_0) \approx -180 \text{ deg}$,
- Slope of $20 \log_{10} |G(j\omega_0)| \approx -60 \text{ dB/dec} \implies \angle G(j\omega_0) \approx -270 \text{ deg}$.

Of course, these values are approximate, and require that the slope around ω_0 persist for approximately a decade before and after ω_0 .

While we do not directly use Equation (17.1) in practice, it has clear implications for feedback design. In particular, the take-away message is that to have a good phase margin, the slope of the Bode magnitude plot at the crossover frequency ω_{co} cannot be too steep, and that it needs to be reasonably shallow for roughly a decade before and after crossover. As a rule of thumb, a phase margin of $PM = 60$ degrees requires that the slope at crossover is between -20 dB/dec and -40 dB/dec . In other words, the ideal loop shape will need to look something like Figure 17.10. It is also important to understand that the Bode phase-gain theorem requires that there is adequate frequency separation between the frequency content of signals that we are tracking or rejecting, and the frequency content of the sensor noise that is to be attenuated.

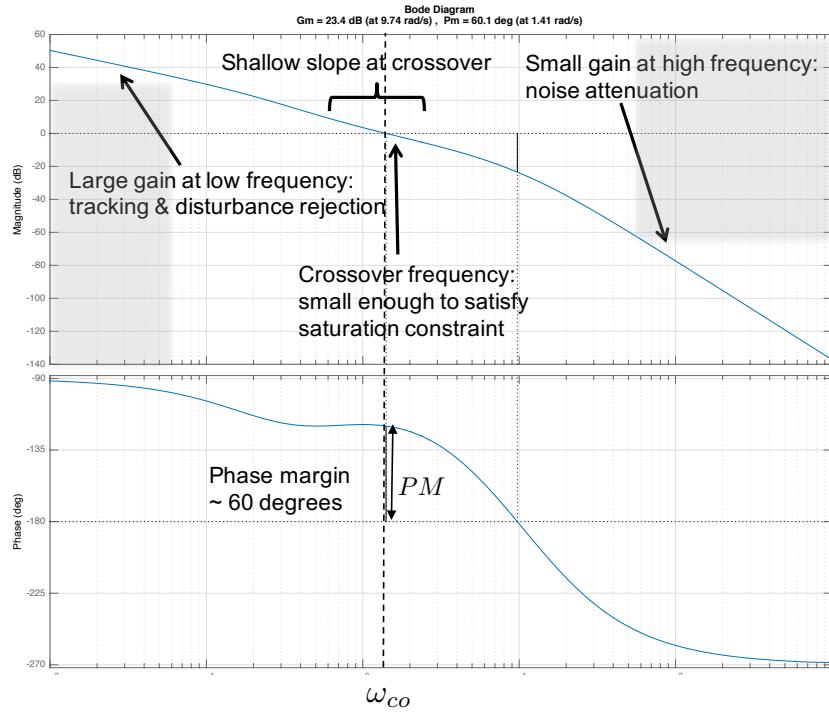


Figure 17.10: The ideal loopshape is large at low frequencies to achieve good tracking and disturbance rejection, is small at high frequencies to attenuate sensor noise, has a shallow slope at the crossover frequency to achieve a good phase margin, and has a small enough crossover frequency to satisfy the input saturation constraints.

17.2 Design Study A. Single Link Robot Arm



Homework Problem A.17

For the single link robot arm, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the closed loop system under PID control. On the same graph, plot the open loop Bode plot and the closed loop Bode plot. What is the bandwidth of the closed loop system, and how does this relate to the crossover frequency? Use the gains found in HW A.10.

Solution

The Matlab code used to generate the plots is shown below.

CHAPTER 17. STABILITY AND ROBUSTNESS MARGINS

```

1 % transfer functions for plant and controller
2 Plant = tf([2/P.m/P.ell^2],[1, 2*P.b/P.m/P.ell^2, 0]);
3 C_pid = tf([(P.kd+P.kp*P.sigma),(P.kp+P.ki*P.sigma),P.ki],...
4 [P.sigma,1,0]);
5
6 % margin and bode plots for PID control
7 figure(1), clf, margin(Plant*C_pid), grid on, hold on
8 bode(Plant*C_pid/(1+Plant*C_pid))
9 legend('Open Loop', 'Closed Loop')

```

The transfer function for the plant is defined in Line 2. The transfer function for the PID controller is

$$\begin{aligned} C_{PID}(s) &= k_P + \frac{k_I}{s} + \frac{k_D s}{\sigma s + 1} = \frac{s(\sigma s + 1)k_P + (\sigma s + 1)k_I + k_D s^2}{s(\sigma s + 1)} \\ &= \frac{(k_D + \sigma k_P)s^2 + (k_P + \sigma k_I)s + k_I}{s(\sigma s + 1)}, \end{aligned}$$

and is defined in Line 3-4. The `margin` command is similar to the `bode` command except that it also annotates the Bode plot with the phase and gain margins. The open loop transfer functions are PC as defined in Line 7, and the closed loop transfer functions are $PC/(1 + PC)$ as defined in Line 8. The results of this code are shown in Figure 17.11.

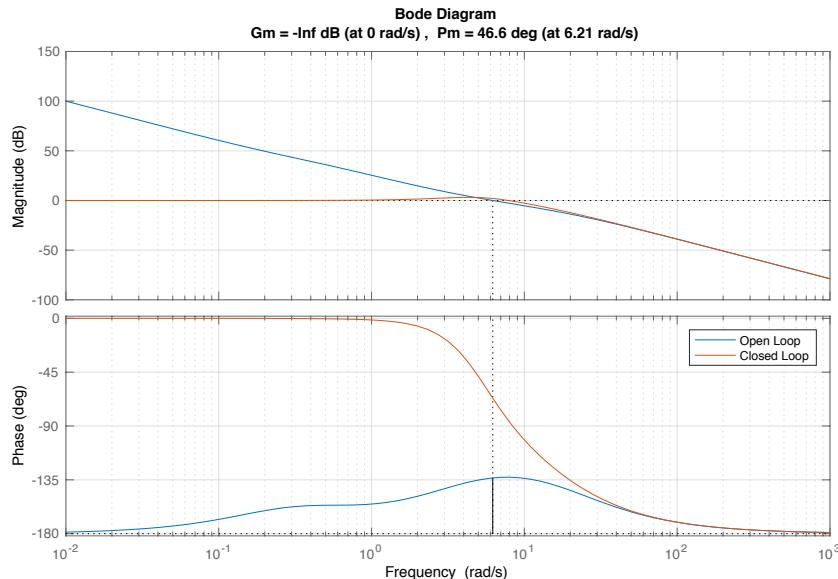


Figure 17.11: The `margin` plot of the open loop system and the `bode` plot of the closed loop system, of the single link robot arm under PID control.

As seen from Figure 17.11 the bandwidth for PID control is approximately

10 rad/sec, which is slightly larger than the cross over frequency of 6.21 rad/sec. The larger bandwidth is due to the small phase margin of $PM = 45.6$ degrees.

17.3 Design Study B. Inverted Pendulum



Homework Problem B.17

For this problem, use the gains found in HW B.10.

- (a) For the inner loop of the inverted pendulum, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the inner loop system under PD control. On the same graph, plot the open loop Bode plot and the closed loop Bode plot. What is the bandwidth of the inner loop, and how does this relate to the crossover frequency?
- (b) For the outer loop of the inverted pendulum, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the outer loop system under PID control. Plot the open and closed loop Bode plots for the outer loop on the same plot as the open and closed loop for the inner loop. What is the bandwidth of the outer loop, and how does this relate to the crossover frequency?
- (c) What is the bandwidth separation between the inner (fast) loop, and the outer (slow) loop. For this design, is successive loop closure justified?

Solution

The Matlab code used to generate the plots is shown below.

```

1 % transfer function for inverted pendulum
2 P_in = tf([-2/P.m2/P.ell],...
3            [1, 0, -2*(P.m1+P.m2)*P.g/P.m2/P.ell]);
4 P_out = tf([-P.m1*P.g/P.m2],[1, P.b/P.m2, 0]);
5 C_in = tf([(P.kd_th+P.sigma*P.kp_th), P.kp_th], [P.sigma, 1]);
6 C_out = tf([(P.kd_z+P.kp_z*P.sigma),...
7              (P.kp_z+P.ki_z*P.sigma),P.ki_z], [P.sigma,1,0]);
8
9 % margin and bode plots
10 figure(1), clf, margin(P_in*C_in), grid on, hold on
11 bode(P_in*C_in/(1+P_in*C_in))
12 margin(P_out*C_out)
13 bode(P_out*C_out/(1+P_out*C_out))
14 legend('Open Loop-Inner', 'Closed Loop-Inner',...
15        'Open Loop-Outer', 'Closed Loop-Outer')

```

CHAPTER 17. STABILITY AND ROBUSTNESS MARGINS

The transfer functions for the inner and outer loop plants and controller are defined in Lines 2–7. For this problem we plot both the inner and outer loop frequency response on the same Bode plot, as implemented in Lines 10–15. The results of this code are shown in Figure 17.12.

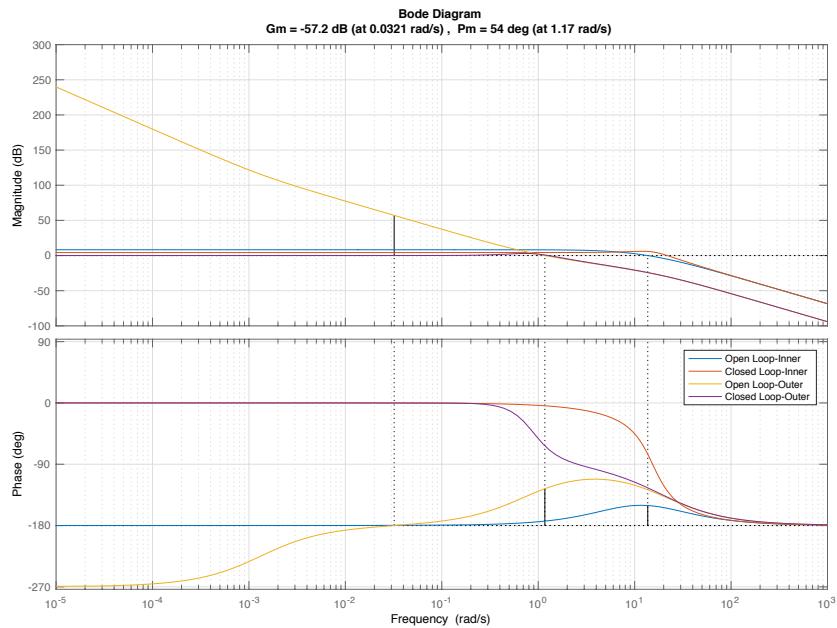


Figure 17.12: The `margin` and `bode` plots for the the open and closed loop systems of both the inner and outer loops of the inverted pendulum system.

As seen from Figure 17.12 the bandwidth of the inner loop is approximately 22 rad/sec, which is slightly larger than the cross over frequency of 14 rad/sec. Similarly, Figure 17.12 indicates that the bandwidth of the outer loop is approximately 1.4 rad/sec, which is slightly smaller than the cross over frequency of 1.2 rad/sec with a phase margin of $PM = 54$ degrees.

The bandwidth separation between the inner and outer loop is about one decade justifying the successive loop closure design approach.

17.4 Design Study C. Satellite Attitude Control



Homework Problem C.17

For this problem, use the gains found in HW C.10.

- (a) For the inner loop of the satellite attitude controller, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the inner loop system under PD control. On the same graph, plot the open loop Bode plot and the closed loop Bode plot. What is the bandwidth of the closed loop system, and how does this relate to the crossover frequency?
- (b) For the outer loop of the satellite attitude controller, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the outer loop system under PI control. Plot the open and closed loop Bode plots for the outer loop on the same plot as the open and closed loop for the inner loop. What is the bandwidth of the closed loop system, and how does this relate to the crossover frequency?
- (c) What is the bandwidth separation between the inner (fast) loop, and the outer (slow) loop. For this design, is successive loop closure justified?

Solution

The Matlab code used to generate the plots is shown below.

```

1 % transfer functions
2 P_in = tf([1/P.Js], [1,P.b/P.Js,P.k/P.Js]);
3 P_out = tf([P.b/P.Jp, P.k/P.Jp], [1,P.b/P.Jp,P.k/P.Jp]);
4
5 C_in = tf([(P.kd_th+P.sigma*P.kp_th), P.kp_th], [P.sigma, 1]);
6 C_out = tf([(P.kd_phi+P.kp_phi*P.sigma),...
7             (P.kp_phi+P.ki_phi*P.sigma),P.ki_phi],...
8             [P.sigma,1,0]);
9
10 % margin and bode plots
11 figure(1), clf, margin(P_in*C_in), grid on, hold on
12 bode(P_in*C_in/(1+P_in*C_in))
13 margin(P_out*C_out)
14 bode(P_out*C_out/(1+P_out*C_out))
15 legend('Open Loop-Inner', 'Closed Loop-Inner',...
16       'Open Loop-Outer', 'Closed Loop-Outer')

```

The transfer functions for the inner and outer loop plants and controller are defined in Lines 2–3. For this problem we plot both the inner and outer loop frequency response on the same Bode plot, as implemented in Lines 10–16. The results of this code are shown in Figure 17.13.

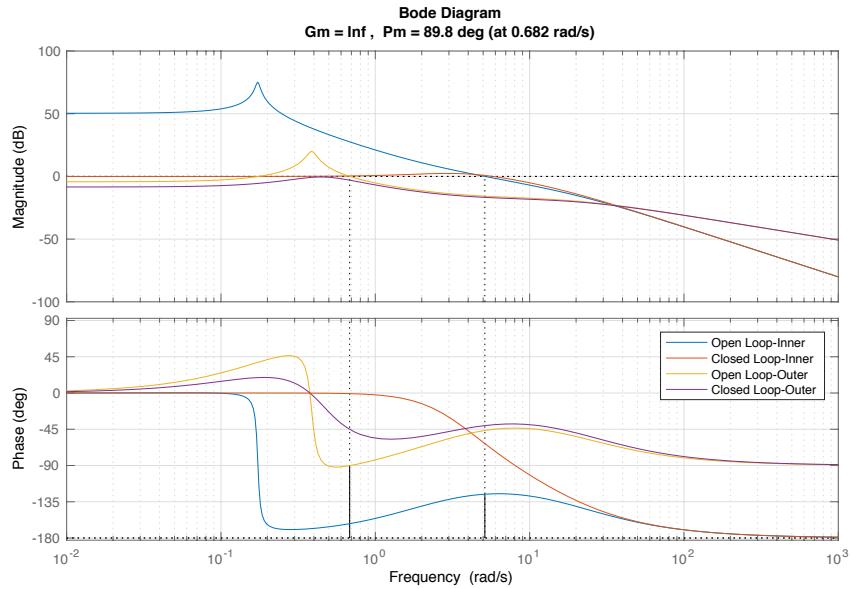


Figure 17.13: The margin and bode plots for the the open and closed loop systems of both the inner and outer loops of the satellite attitude control system.

As seen from Figure 17.13 the bandwidth of the inner loop is approximately 7.8 rad/sec, which is slightly larger then the cross over frequency of 5.2 rad/sec, with a phase margin of $PM = 55$ degrees. Similarly, Figure 17.13 indicates that the bandwidth of the outer loop is approximately the cross over frequency of 0.68 rad/sec. The bandwidth separation between the inner and outer loop is close to a decade and successive loop closure is approximately one far beyond the cross over frequency of the outer loop.

Notes and References

Chapter 18

Compensator Design

18.1 Theory

The loopshaping design methodology is to add elements to the compensator $C(s)$ to achieve a loop shape that satisfies the design specifications and that results in good stability margins, ideally $PM \approx 60$ degrees, all while satisfying input saturation constraints. The idea is that the controller $C(s)$ will be composed of several building blocks:

$$C(s) = C_1(s)C_2(s) \cdots C_q(s).$$

Since the loop gain is $P(s)C(s)$, on the bode plot we have

$$\begin{aligned} 20 \log_{10} |PC| &= 20 \log_{10} |P| + \sum_{i=1}^q 20 \log_{10} |C_i| \\ \angle PC &= \angle P + \sum_{i=1}^q \angle C_i. \end{aligned}$$

Therefore, the total design can be realized by adding different elements to the Bode plot until the design specifications are satisfied. In this chapter we will discuss the following five basic building blocks for loopshaping design:

- **Proportional Gain.** A proportional gain does not have phase (assuming $k_P > 0$) and only affects the magnitude plot, by moving it up and down on the Bode magnitude plot. A proportional gain is used effect the location of the cross over frequency.
- **Integral (PI) Control.** An integrator adds -20 dB of slope and -90 degrees of phase at all frequencies. The addition of an integrator changes the system type. A modified PI controller can be used to affect only the low frequency loop gain, thereby enhancing reference tracking and input/output disturbance rejection.

- **Low Pass Filter.** Low pass filters are used to decrease the loop gain at high frequencies to achieve noise attenuation.
- **Phase Lag Filter.** A phase lag filter can be used to increase the loop gain at low frequencies thereby enhancing reference tracking and input/output disturbance rejection. A phase lag filter differs from integral control in that it does not change the system type.
- **Phase Lead Filter.** A phase lead filter can be used to add phase at the cross over frequency and can thereby stabilize the system and increase the phase margin.

Each of these basic building blocks will be described in more detail in the sections that follow.

18.1.1 Building Block #1: Proportional Gain

The first basic building block for loopshaping design is proportional control where

$$C_P(s) = k_P.$$

Proportional control is used to change the loop gain at all frequencies, without affecting the phase. A Bode plot for proportional control is shown in Figure 18.1 for different values of k_P .

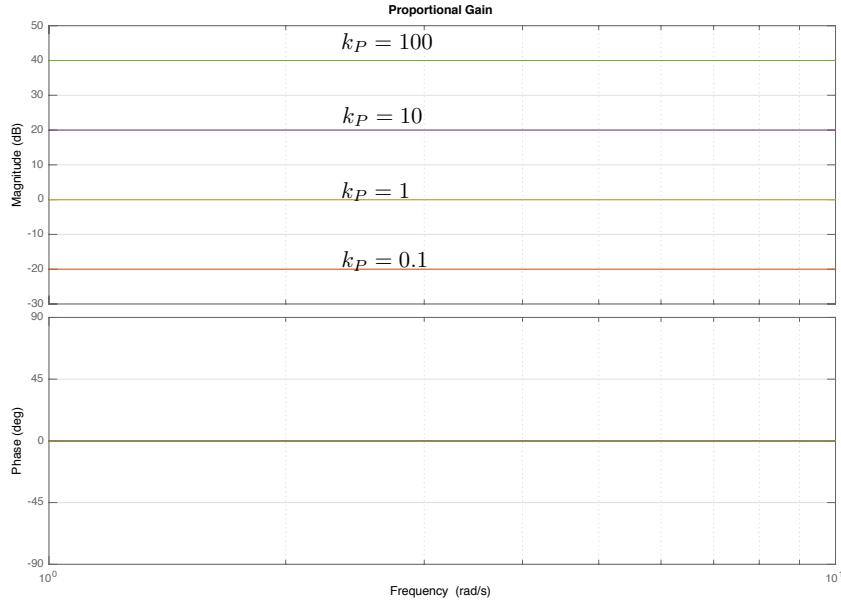


Figure 18.1: The Bode plot for proportional gain at different values of k_P . The proportional gain adds a constant to the loopshape at all frequencies, and does not affect the phase at any frequency.

18.1.2 Building Block #2: Integral Control

The second basic building block for loopshaping design is proportional-integral (PI) control where $k_P = 1$, i.e.,

$$C_{PI}(s) = 1 + \frac{k_I}{s} = \frac{s + k_I}{s}.$$

The Bode plot of C_{PI} for different values of k_I is shown in Figure 18.2. Integral control is used to change the system type by changing the slope of the loop gain by -20 dB/dec at low frequencies. The Bode plot for integral control is shown in Figure 18.2 for different values of k_I . It is clear that integral control changes the slope for frequencies $\omega < k_I$ while not affecting the loop gain when $\omega > k_I$. Note also that integral control decreases the phase at low frequencies, which negatively impacts stability. Therefore, integral control is usually applied at low frequencies, where $k_I \ll \omega_{co}$.

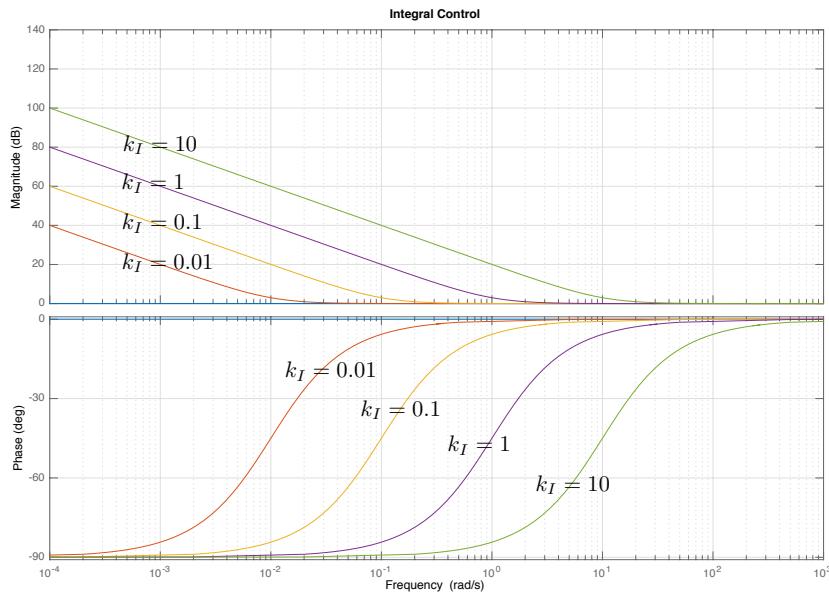


Figure 18.2: The Bode plot for integral control at different values of k_I . Integral control changes the system type by decreasing the slope of the loopshape by -20 dB/dec for frequencies below k_I , while leaving high frequencies unaffected.

18.1.3 Building Block #3: Low Pass Filter

The third basic building block for loopshaping design is a low pass filter where

$$C_{LPF}(s) = \frac{p}{s + p}.$$

Note that the low pass filter is constructed so that the DC-gain is one. The Bode plot for a low pass filter is shown in Figure 18.3 for different values of p . It is evident from Figure 18.3 that a low pass filter decreases the loopgain when $\omega > p$, while leaving the loop gain unaffected when $\omega < p$. Low pass filters are used at high frequencies to enhance noise attenuation.

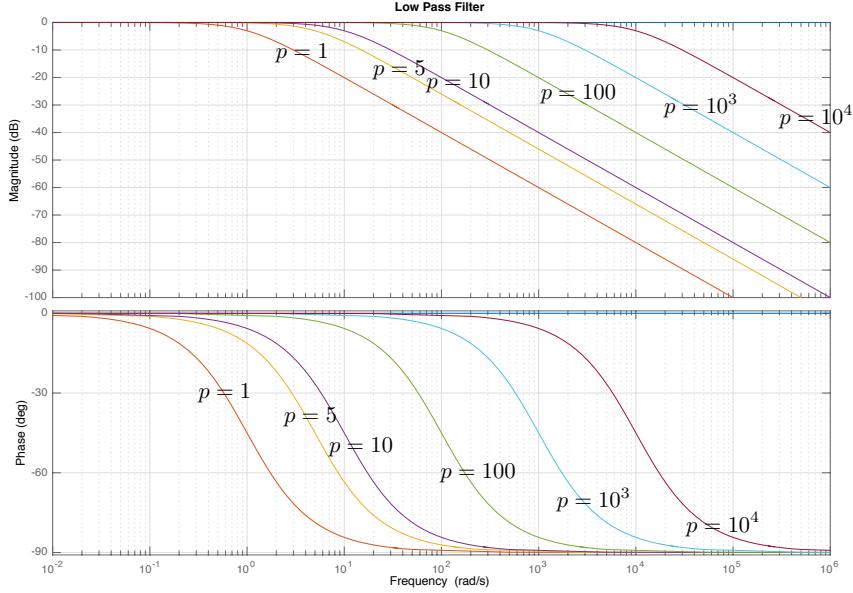


Figure 18.3: The Bode plot for a low pass filter at different values of the pole p . A Low pass filter decreasing the slope of the loopshape by -20 dB/dec for frequencies above p , while leaving low frequencies unaffected.

18.1.4 Building Block #4: Phase Lag Filter

The fourth basic building block for loopshaping design is a phase lag filter where

$$C_{Lag}(s) = \frac{s + z}{s + p}, \quad (18.1)$$

where $z > p > 0$. Note that the phase lag filter is constructed so that

$$\lim_{s \rightarrow \infty} \frac{s + z}{s + p} = \lim_{s \rightarrow \infty} \frac{1 + \frac{z}{s}}{1 + \frac{p}{s}} = 1.$$

Therefore, phase lag filters change the loop gain at low frequencies while leaving high frequencies unaffected. The Bode plot for a phase lag filter is shown in Figure 18.4 when $z = 1$ and when $p = z/M$ for different values of pole-zero separation M . Note that phase-lag filters increase the gain at low frequencies and that the total amount of gain is equal to the separation between the zero and the pole. Specifically, the increased gain on the Bode plot is equal to $20 \log_{10} M$. Phase-lag filter are used to enhance tracking performance and input/output disturbance rejection. Unfortunately phase-lag filters decrease the phase between the zero and the pole and will therefore negatively affect stability if the zero is too close to the crossover frequency. Therefore, phase lag filter are typically added at low frequencies, and are characterized by the zero z where the effect

of the lag ends and the pole zero separation M which indicates the maximum gain at low frequencies. Accordingly, the lag filter (18.1) can be written as

$$C_{Lag}(s) = \frac{s+z}{s+z/M}.$$

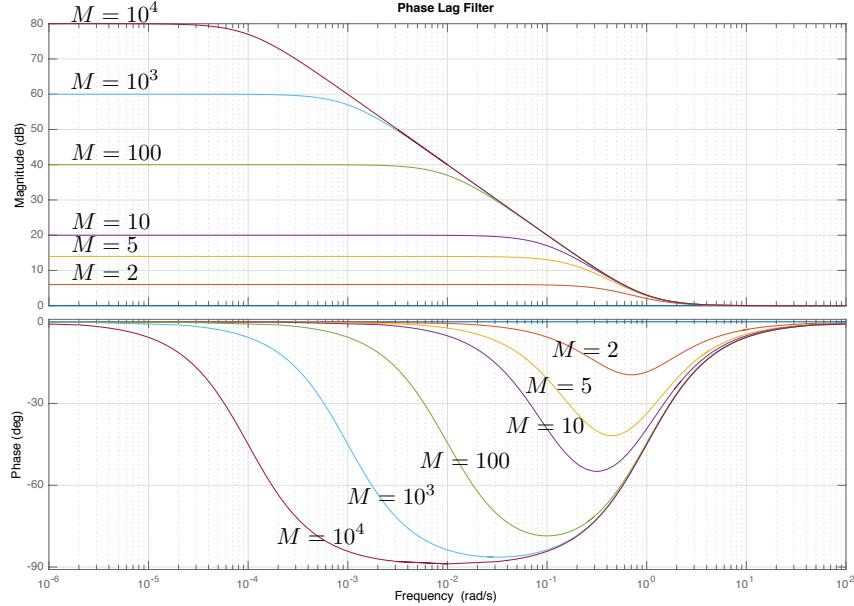


Figure 18.4: The Bode plot for a phase lag filter when the zeros is at $z = 1$ and the pole is at $p = z/M$, for different values of M . A phase lag filter increases the loop gain at low frequencies while leaving high frequencies unaffected.

18.1.5 Building Block #5: Phase Lead Filter

The final basic building block for loopshaping design that we will discuss is a phase lead filter where

$$C_{Lead}(s) = \left(\frac{p}{z}\right) \left(\frac{s+z}{s+p}\right), \quad (18.2)$$

where $p > z > 0$. Note that the phase lag filter is constructed so that

$$\lim_{s \rightarrow 0} \frac{p}{z} \frac{s+z}{s+p} = 1.$$

Therefore, phase lead filters change the loop gain at high frequencies while leaving low frequencies unaffected. The Bode plot for a phase lead filter is shown in Figure 18.5 when $z = 1$ and when $p = Mz$ for different values of pole-zero

CHAPTER 18. COMPENSATOR DESIGN

separation M . Note that phase-lead filters increase the phase for frequencies between the zero and the pole. It can be shown that the peak of the phase plot shown in Figure 18.5 is at

$$\omega_{Lead} = \sqrt{zp}$$

which is the midpoint between z and p on the Bode plot. It can also be shown that the increase in phase at ω_{Lead} is

$$\phi_{Lead} = \tan^{-1} \sqrt{\frac{p}{z}} - \tan^{-1} \sqrt{\frac{z}{p}}.$$

Phase-lead filter are used to increase the phase margin and thereby enhance the stability and robustness of the system. Typically ω_{Lead} is selected to be the cross over frequency ω_{co} to get the maximum benefit to the phase margin. Unfortunately phase-lead filters also increase the loop gain at high frequencies as shown in Figure 18.5, which will make the closed-loop system more sensitive to noise. Therefore, phase lead filters are typically accompanied by a low pass filter at higher frequencies. Since phase-lead filters are typically added at the frequency giving the desired bump in phase, it is convenient to write the transfer function (18.2) in terms of the center frequency ω_{Lead} and the lead ratio M . Using the relationship $p = Mz$ and $\omega_{Lead} = \sqrt{pz}$ we can write Equation (18.2) as

$$C_{Lead}(s) = M \left(\frac{s + \frac{\omega_{Lead}}{\sqrt{M}}}{s + \omega_{Lead}\sqrt{M}} \right)$$

$$C_{Lead}(s) = \frac{\frac{\sqrt{M}}{\omega_{Lead}} s + 1}{\frac{1}{\omega_{Lead}\sqrt{M}} s + 1}.$$

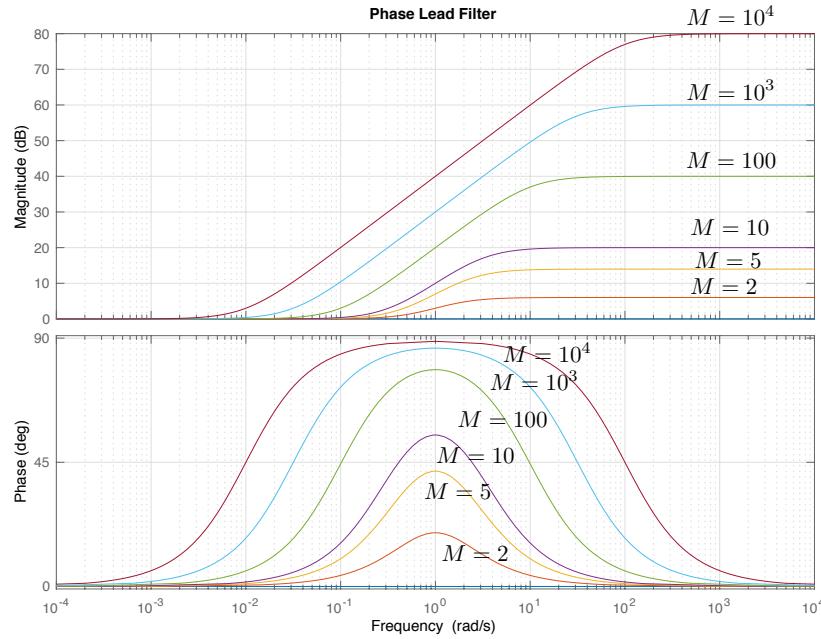


Figure 18.5: The Bode plot for a phase lead filter when $\phi_{Lead} = 1$, for different values of M . A phase lead filter increases the phase around ϕ_{Lead} . It also increases the loop gain at high frequencies.

18.1.6 Simple Example 1

Given the open loop plant

$$P(s) = \frac{1}{s+1},$$

design a controller $C(s)$ so that the closed loop system satisfies the following constraints:

- (1) Tracks a ramp within $\gamma_1 = 0.03$.
- (2) Attenuate noise with frequency content above $\gamma_n = 10$ rad/sec by $\gamma_n = 0.1$.
- (3) Reject input disturbances with frequencies content below $\omega_{d_{in}} = 0.1$ rad/sec by $\gamma_{d_{in}} = 0.1$.
- (4) Phase margin that is approximately 60 degrees.

The first three design specifications can be displayed on the Bode magnitude plot as shown in Figure 18.6. To track a ramp within $\gamma_1 = 0.03$ requires that the loop gain be above

$$B_1 = 20 \log_{10} \left| \frac{1}{0.03} \right| - 20 \log_{10} |\omega|$$

CHAPTER 18. COMPENSATOR DESIGN

as $\omega \rightarrow 0$. Therefore, to satisfy this specification, we need to find C so that the Bode plot for PC rises above the green line on the top left of Figure 18.6.

To attenuate noise with frequency content above $\gamma_n = 10$ rad/sec by $\gamma_n = 0.1$, the loop gain should be below

$$B_n = 20 \log_{10} |0.1| = -20 \text{ dB}$$

for $\omega > 10$ rad/sec. Therefore, we need to design C so that the Bode plot for PC is below the green line in the bottom right of Figure 18.6.

To reject input disturbances with frequencies content below $\omega_{d_{in}} = 0.1$ rad/sec by $\gamma_{d_{in}} = 0.1$, requires that the loop gain is above

$$20 \log_{10} \left| \frac{1}{0.1} \right| + 20 \log_{10} |P(j\omega_{d_{in}})| = 20 - 0 = 20 \text{ dB}$$

for $\omega < 0.1$ rad/sec. Therefore we need to design C so that the Bode plot for PC is above the green line in the left-middle of Figure 18.6.

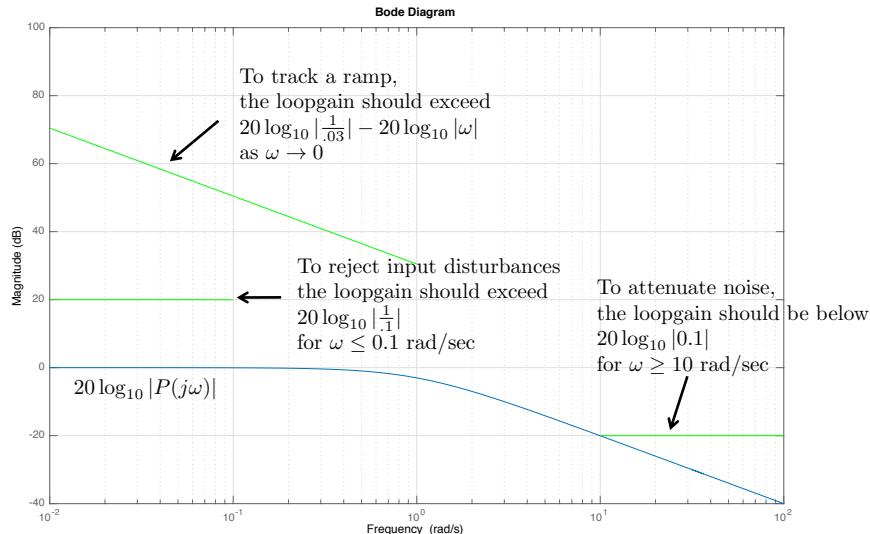


Figure 18.6: The Bode plot for the open loop plant in Example 1, together with the design specifications.

To increase the type of the system so that the slope of PC is -20 dB/dec as $\omega \rightarrow 0$, integral control can be added. Figure 18.7 shows the loopgain of PC when

$$C(s) = C_P(s)C_{PI}(s) = (2) \left(\frac{s + 0.4}{s} \right).$$

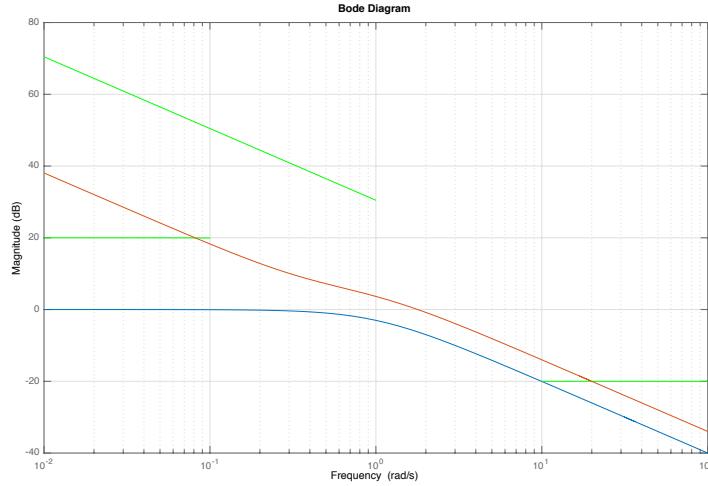


Figure 18.7: The Bode gain plot of P and PC when C includes proportional and integral control.

To increase the loop gain a low frequencies to get above the $1/s$ line as $\omega \rightarrow 0$, a phase lag filter can be added. Figure 18.8 shows the loopgain of PC when

$$C(s) = C_P(s)C_{PI}(s)C_{Lag}(s) = (2) \left(\frac{s + 0.4}{s} \right) \left(\frac{s + 0.8}{s + 0.8/50} \right).$$

From Figure 18.8 we see that the closed loop system will satisfy the tracking and input disturbance specifications.

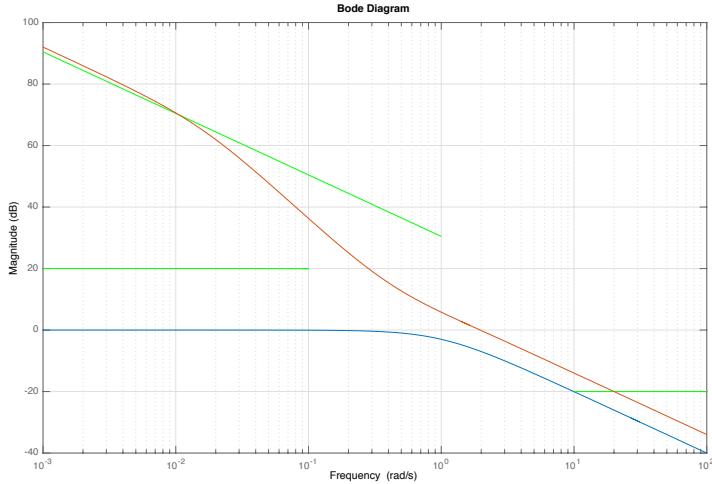


Figure 18.8: The Bode gain plot of P and PC when C includes proportional, integral, and lag filters.

To decrease the loop gain at high frequencies to achieve the noise attenuation specification, a low pass filter can be added at $p = 5$. Figure 18.9 shows the loopgain of PC when

$$C(s) = C_P(s)C_{PI}(s)C_{Lag}(s)C_{LPF}(s) = (2) \left(\frac{s + 0.4}{s} \right) \left(\frac{s + 0.8}{s + 0.8/50} \right) \left(\frac{5}{s + 5} \right).$$

The phase margin for this design is $PM = 63$ degrees and so all of the design specifications are satisfied, as shown in Figure 18.9.

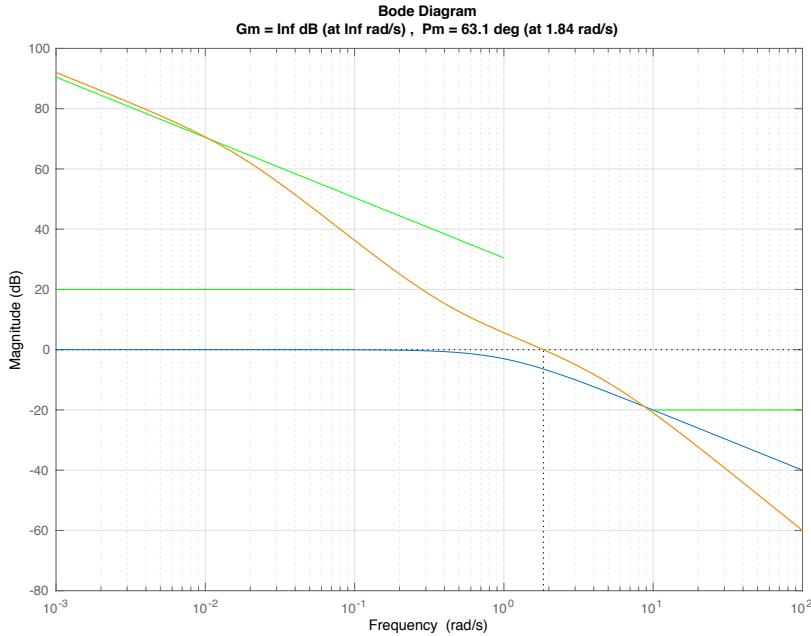


Figure 18.9: The Bode gain plot of P and PC when C includes proportional, integral, lag, and low pass filters.

Matlab code that implements this example is shown below.

```

1 Plant = tf(1,[1,1]); % Plant transfer function
2 % initialize bode magnitude plot
3 figure(2), clf, bodemag(Plant), hold on, grid on
4 % input disturbance specification
5 omega_din = 10^-1; % reject dist below this frequency
6 gamma_din = 0.1; % amount of input disturbance in output
7 w = logspace(log10(omega_din)-2,log10(omega_din));
8 Pmag=bode(Plant,w);
9 for i=1:size(Pmag,3), Pmag_(i)=Pmag(1,1,i); end
10 plot(w,20*log10(1/gamma_din)*ones(1,length(Pmag_))...
11 +20*log10(Pmag_), 'g')
12 % noise specification
13 omega_n = 10^1; % attenuate noise above this frequency
14 gamma_n = .1; % attenuate noise by this amount
15 w = logspace(log10(omega_n),2*log10(omega_n));
16 plot(w,20*log10(gamma_n)*ones(size(w)), 'm')
17 % steady state tracking of ramp
18 gamma_1 = .03;
19 w = logspace(-4, 0);
20 plot(w,20*log10(1/gamma_1)-20*log10(w), 'g')
21
22 %%%%%%%%%%%%%%
23

```

```

24 % Control Design
25 C = 1;
26 % proportional control
27 kp = 2;
28 C = C*kp;
29 % integral control
30 k_I = .4;
31 Integrator = tf([1,k_I],[1,0]);
32 C = C*Integrator;
33 % phase lag
34 z = .8;
35 M = 50;
36 Lag = tf([1,z],[1,z/M]);
37 C = C*Lag;
38 % low pass filter
39 p = 5;
40 LPF = tf(p,[1,p]);
41 C = C*LPF;
42
43 margin(Plant*C) % update plot

```

18.1.7 Controller Implementation

The loopshaping design procedure produces a controller $C(s)$ that will have multiple elements including integrators, lead and lag filters, and low pass filters. Each of these elements are straightforward to implement using analog circuits. However, if the controller is to be implemented on digital hardware using computer code, what is the best way to proceed? While there are many possibilities, one option that is straightforward to implement and that fits well with the previous chapters in the book, is to implement the controller using state space equations. The controller

$$U(s) = C(s)E(s)$$

is first converted to continuous time state space equations using, for example, control canonical form, to produce the equations

$$\dot{z}_C = A_C z_C + B_C e \quad (18.3)$$

$$u = C_C z_C + D_C e \quad (18.4)$$

where $e(t) = r(t) - y(t)$ is the input to the state space equations, and the control signal $u(t)$ is the output. In the example of the previous section we get

$$C(s) = \frac{10s^2 + 12s + 3.2}{s^3 + 5.016s^2 + 0.08s}.$$

Therefore, using control canonic form, the controller implementation is

$$\begin{aligned} \dot{z}_C &= \begin{pmatrix} -5.016 & 0.08 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} z_C + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} e \\ u &= (10 \quad 12 \quad 3.2) z_C. \end{aligned}$$

In Matlab, the commands to find the state space equations are given by

```
1 [num,den] = tfdata(C, 'v');
2 [A_C,B_C,C_C,D_C]=tf2ss(num,den);
```

The controller is then implemented by integrating the differential equations between sample times. If T_s is the sample time, and $N = 10$ Euler steps are used between each sample, then Matlab code that implements the controller is given by

```
1 function u=ctrl(in,P)
2     r      = in(1); % reference input
3     y      = in(2); % plant output
4     t      = in(3); % current time
5     % define and initialize persistent variables
6     persistent z_C % state of the controller
7     if t==0,
8         z_C = zeros(size(P.A_C,1),1);
9     end
10    % error signal
11    error = r - y;
12    N = 10; % number of Euler integration steps per sample
13    for i=1:N, % solve differential equation for controller
14        z_C = z_C + P.Ts/N*( P.A_C*z_C + P.B_C*error );
15    end
16    % output equation for the controller
17    u = P.C_C*z_C + P.D_C*error;
18 end
```

The inputs to the controller are the reference signal r , the output y , and the current time t . In Lines 6–9 the state of the controller z_C is initialized to the zero vector. The error signal is computed on Line 11. The control dynamics (18.3) are integrated using N Euler steps in Lines 12–15, and the control output equation is computed in Line 17.

18.1.8 Prefilter Design

There are applications where it is very difficult to meet the design specifications at both low and high frequencies and still achieve a phase margin that is close to 60 degrees. If for example, the phase margin is close to 45 degrees, then there will be a peak in the closed loop Bode plot. As shown in Figure 17.7, the peak in the closed loop Bode plot produces overshoot and ringing in the closed-loop step response. One method for reducing the overshoot in the closed-loop step response is to add a prefilter $F(s)$ as shown in Figure 18.10.

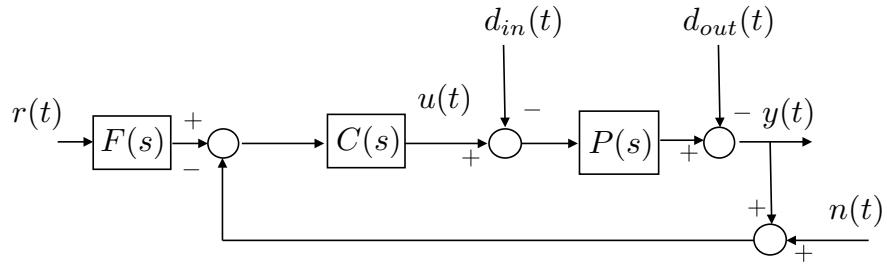


Figure 18.10: The closed loop system with a prefilter $F(s)$ on the reference command r .

The role of the prefilter $F(s)$ is to attenuate frequencies in $r(t)$ that resonate with the closed loop system and produce the overshoot. From a Bode plot perspective, the goal of the prefilter is to flatten the closed loop Bode plot so that the peak is removed. There are several standard forms used for the prefilter. These include a notch filter and a low pass filter. For simplicity, we will discuss the use of a low pass filter below. As explained previously, the transfer function for a low pass filter is given by

$$F(s) = \frac{p}{s + p}, \quad (18.5)$$

where the DC-gain is equal to one.

Just as the controller $C(s)$ was implemented using state space equations, the prefilter can also be implemented using state space equations. The prefilter

$$R_f(s) = F(s)R(s)$$

is converted to continuous time state space equations using, for example, control canonical form, to produce the equations

$$\dot{z}_F = A_F z_F + B_F r \quad (18.6)$$

$$r_f = C_F z_F + D_F r \quad (18.7)$$

where the reference $r(t)$ is the input to the state space equations, and the filtered reference signal $r_f(t)$ is the output. For the transfer function given in Equation (18.5) we get

$$\begin{aligned} \dot{z}_F &= -p z_F + r \\ r_f &= p z_F. \end{aligned}$$

In Matlab, the commands to find the state space equations are given by

```

1 [num,den] = tfdata(F, 'v');
2 [A_F,B_F,C_F,D_F]=tf2ss(num,den);

```

The prefilter is then implemented by integrating the differential equations between sample times. If T_s is the sample time, and $N = 10$ Euler steps are used between each sample, then Matlab code that implements the controller plus prefilter is given by

```

1  function u=ctrl(in,P)
2      r      = in(1);
3      y      = in(2);
4      t      = in(3);
5      % define and initialize persistent variables
6      persistent z_C % state of the controller
7      persistent z_F % state of the prefilter
8      if t==0,
9          z_C = zeros(size(P.A_C,1),1);
10         z_F = zeros(size(P.A_F,1),1);
11     end
12     N = 10; % number of Euler integration steps per sample
13     for i=1:N, % solve differential equation for prefilter
14         % prefilter the reference command
15         z_F = z_F + P.Ts/N*( P.A_F*z_F + P.B_F*r );
16         r_filtered = P.C_F*z_F + P.D_F*r;
17         % error signal uses filtered reference
18         error = r_filtered - y;
19         % update control state using the error
20         z_C = z_C + P.Ts/N*( P.A_C*z_C + P.B_C*error );
21     end
22     % output equation for the controller
23     u = P.C_C*z_C + P.D_C*error;
24 end

```

Prefilter Example

As an example, consider the open loop system PC shown in Figure 18.11. The phase margin is $PM = 42.5$ degrees resulting in a peak in the closed loop frequency response, as shown in Figure 18.11. The corresponding step response is shown in Figure 18.12, where it can be seen that there is significant ringing in the step response.

To remove the peak in the closed loop frequency response, a prefilter is added to the system as shown in Figure 18.10, where the low pass filter pole is $p = 9$, resulting in

$$F(s) = \frac{9}{s+9}.$$

The Bode magnitude of the prefilter F is shown in Figure 18.11, as well as the prefiltered closed loop response $FPC/(1 + PC)$. As can be seen from Figure 18.11, the prefilter removes the peak in the closed loop frequency response. The corresponding step response is shown in Figure 18.12, where it can be seen that the ringing has been significantly reduced.

CHAPTER 18. COMPENSATOR DESIGN

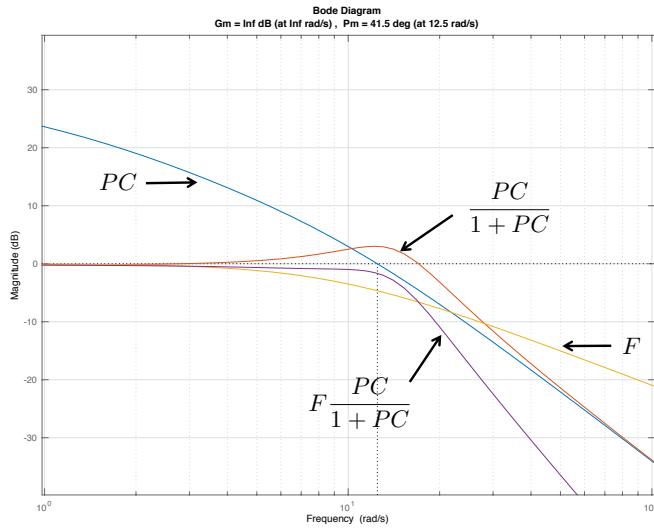


Figure 18.11: The loopgain for the open loop system PC , the closed loop system $PC/(1 + PC)$, the prefilter F , and the closed loop system with the prefilter $FPC/(1 + PC)$.

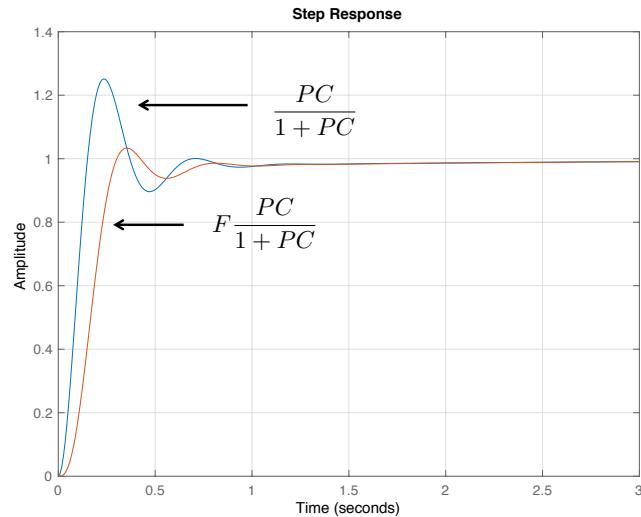


Figure 18.12: The step response of the closed loop system $PC/(1 + PC)$ and the closed loop system with the prefilter $FPC/(1 + PC)$.

18.1.9 Successive Loop Closure Design

When loopshaping is used to design an inner and outer loop controller in a successive loop closure design, the design of the outer loop can take into account the complete dynamics of the inner loop. This is in contrast to the design of PID controllers in Chapter 8 where the inner loop was replaced with the DC-gain. Figure 18.13 shows the system architecture for loopshaping design. The inner and outer loop controller are accompanied by the prefilter $F(s)$ which is only on the outer loop.

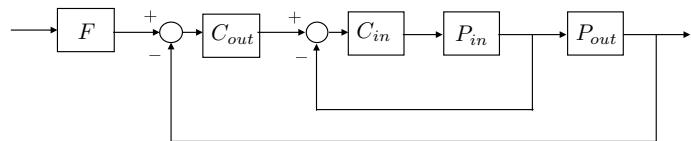


Figure 18.13: The control architecture for a successive loop closure design, with prefilter.

The first step is to design the inner loop controller $C_{in}(s)$ using the plant $P_{in}(s)$. However, for the design of the outer loop controller $C_{out}(s)$, we can use the outer loop plant in cascade with inner closed loop system. Therefore, the plant for the outer loop design is given by

$$P = P_{out} \left(\frac{P_{in}C_{in}}{1 + P_{in}C_{in}} \right).$$

Examples will be given in the sections below.

18.2 Design Study A. Single Link Robot Arm



Homework Problem A.18

For this homework assignment we will use loopshaping to improve the PID controllers developed in HW A.10. Let $C_{pid}(s)$ be the PID controller designed in HW A.10. The final control will be $C(s) = C_{pid}(s)C_l(s)$ where C_l is designed using loopshaping techniques.

(a) Design $C_l(s)$ to meet the following objectives:

- (1) Improve tracking and disturbance rejection by a factor of 10 for reference signals and disturbances below 0.07 rad/sec,
- (2) Improve noise attenuation by a factor of 10 for frequencies above 1000 radians/sec.
- (3) Phase margin that is approximately $PM = 60$ degrees.

- (b) Add zero mean Gaussian noise with standard deviation $\sigma^2 = 0.01$ to the Simulink diagram developed in HW A.10.
- (c) Implement the controller $C(s)$ in Simulink using its state space equivalent.
- (d) Note that despite having a good phase margin, there is still significant overshoot, due in part to the windup effect in the phase lag filter. This can be mitigated by adding a prefilter, that essentially modifies the hard step input into the system. Add a low pass filter for $F(s)$ as a prefilter to flatten out the closed loop Bode response and implement in Simulink using its discrete time state space equivalent.

Solution

The first two design specifications can be displayed on the Bode magnitude plot as shown in Figure 18.14. To improve the tracking and disturbance rejection by a factor of 10 for reference signals and disturbances below $\omega_r = 0.07$ rad/sec, the loop gain must be 20 dB above $P(s)C_{PID}(s)$, as shown by the green line in the top left of Figure 18.14. To improve noise attenuation by a factor of 10 for frequencies above $\omega_n = 1000$ rad/sec, the loop gain must be 20 dB below $P(s)C_{PID}(s)$, as shown by the green line in the bottom right of Figure 18.14.

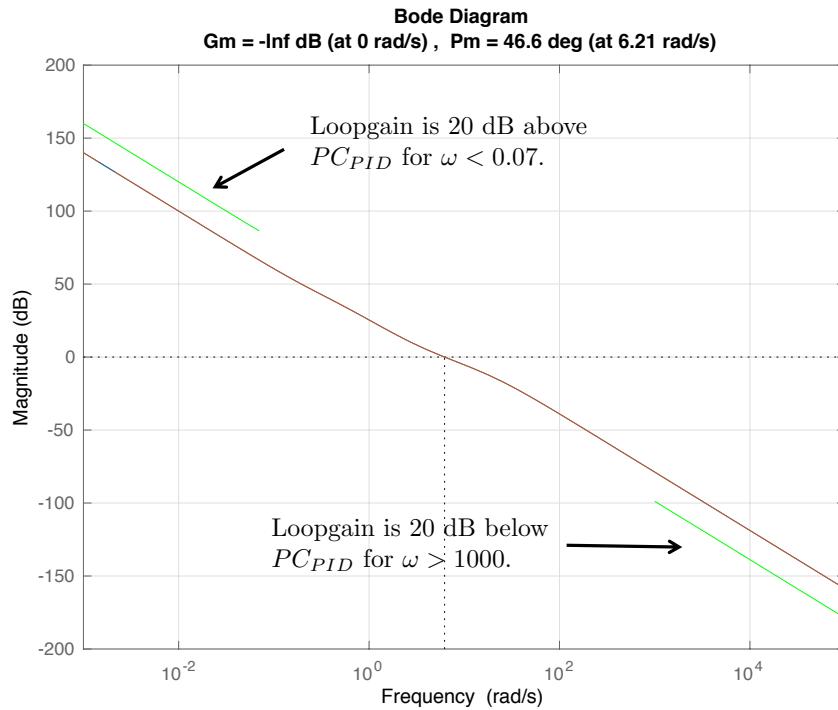


Figure 18.14: The Bode plot for the open loop plant in HW A.18, together with the design specifications.

To increase the loop gain below $\omega_r = 0.07$ rad/sec, a phase lag filter can be added with zero at $z = 0.7$ with a separation of $M = 10$. Figure 18.15 shows the loopgain of PC when

$$C(s) = C_{PID}(s)C_{Lag}(s) = \left(\frac{0.1036s^2 + 0.3108s + 0.1}{0.05s^2 + s} \right) \left(\frac{s + 0.7}{s + 0.07} \right).$$

From Figure 18.15 we see that the closed loop system will satisfy the tracking and input disturbance specifications.

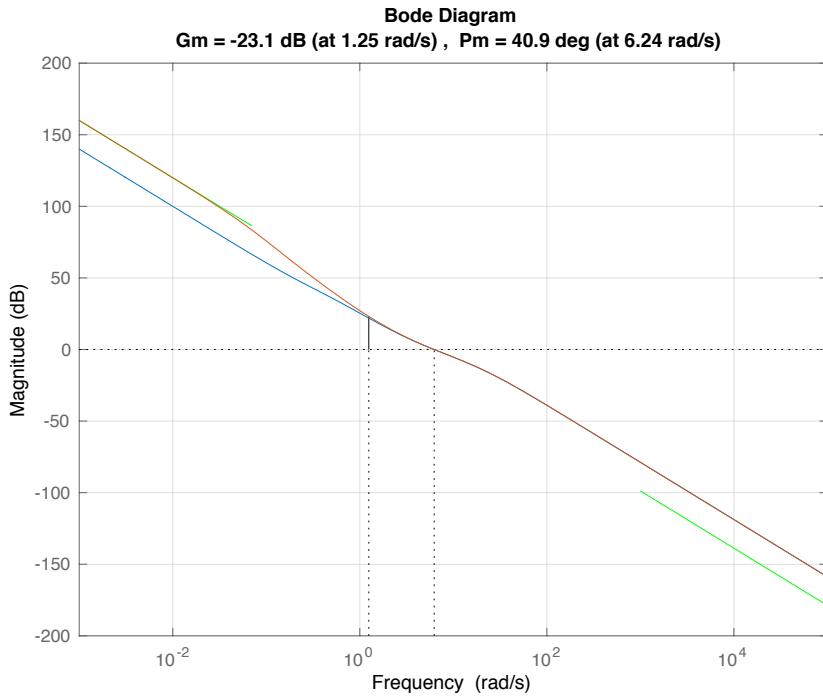


Figure 18.15: The Bode plot for HW A.18 where $C = C_{PID}C_{lag}$.

The phase margin after adding the lag filter is $PM = 40.9$ degrees. To increase the phase margin, a phase lead filter is added with center frequency $\omega_c = 30$ rad/sec and a separation of $M = 10$. The center frequency for the lead filter is selected to be above the cross over frequency so as not to change the location of the cross over frequency, thereby keeping the closed loop bandwidth, and therefore the control effort, roughly the same as with the PID controller. After adding the lead filter, the controller is

$$\begin{aligned} C(s) &= C_{PID}(s)C_{Lag}(s)C_{Lead}(s) \\ &= \left(\frac{0.1036s^2 + 0.3108s + 0.1}{0.05s^2 + s} \right) \left(\frac{s + 0.7}{s + 0.07} \right) \left(\frac{10s + 94.87}{s + 94.87} \right), \end{aligned}$$

and the corresponding loop gain is shown in Figure 18.16.

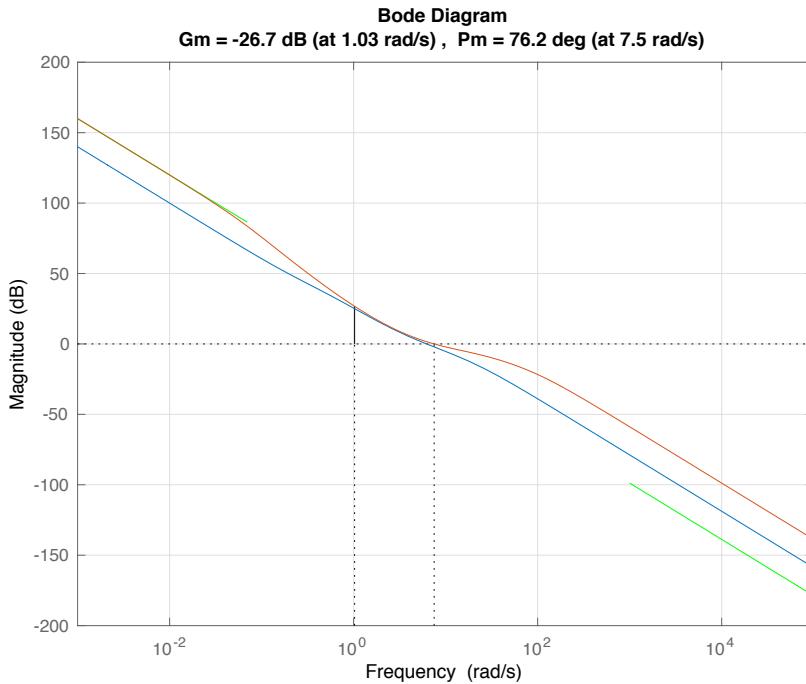


Figure 18.16: The Bode plot for HW A.18 where $C = C_{PID}C_{lag}C_{lead}$.

In order to satisfy the noise attenuation specification, we start by adding a low pass filter at $p = 50$. The corresponding loop gain is shown in Figure 18.17, from which we see that the noise specification is not yet satisfied. Therefore, we add an additional low pass filter at $p = 150$ to obtain the loopgain in Figure 18.18. The phase margin is $PM = 64$ degrees. The corresponding controller is

$$\begin{aligned} C(s) &= C_{PID}(s)C_{Lag}(s)C_{Lead}(s) \\ &= \left(\frac{0.1036s^2 + 0.3108s + 0.1}{0.05s^2 + s} \right) \left(\frac{s + 0.7}{s + 0.07} \right) \left(\frac{10s + 94.87}{s + 94.87} \right) \left(\frac{50}{s + 50} \right) \left(\frac{150}{s + 150} \right). \end{aligned}$$

Note that we have selected the filter values to leave the cross over frequency the same as with the original PID controller.

CHAPTER 18. COMPENSATOR DESIGN

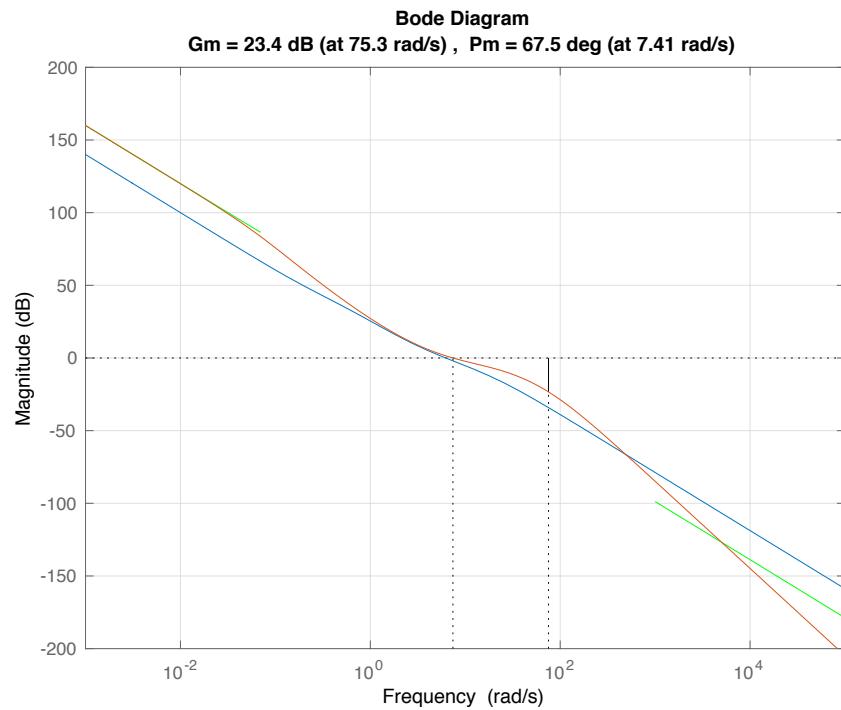


Figure 18.17: The Bode plot for HW A.18 where $C = C_{PID}C_{lag}C_{lead}C_{lpf}$.

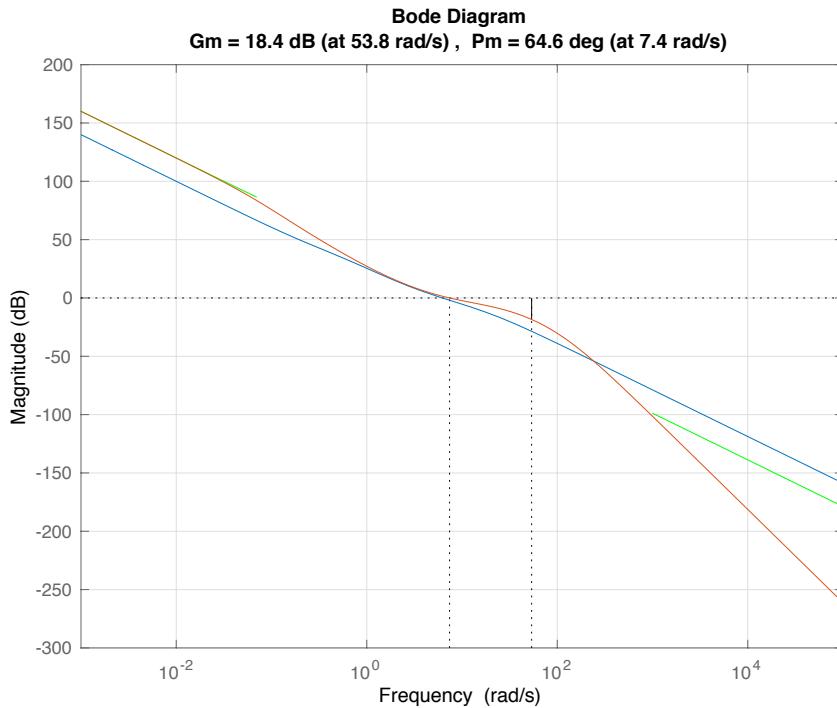


Figure 18.18: The Bode plot for HW A.18 where $C = C_{PID}C_{lag}C_{lead}C_{lpf}C_{lpf2}$.

The closed loop frequency response $PC/(1 + PC)$ and the corresponding step response and control effort are shown by the blue lines in Figure 18.19. The overshoot in the step response is caused by the small amount of peaking on the closed loop bode plot. The peaking can be reduced by adding a prefilter, which in this case is a low pass filter with pole $p = 3$. The prefiltered closed loop frequency response $FPC/(1 + PC)$ and the corresponding step response and control effort are shown by the red lines in Figure 18.19. As shown by Figure 18.19, the prefilter reduces the overshoot and lowers the control effort.

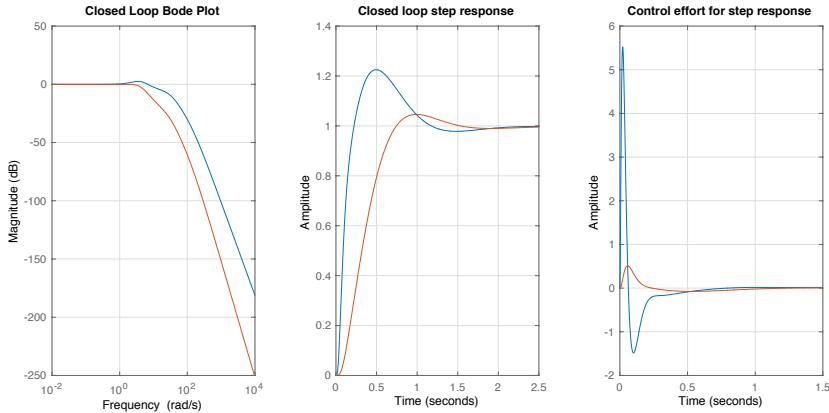


Figure 18.19: On the left is the closed loop frequency response in blue, and the prefiltered closed loop frequency response in red. In the middle are the associated closed-loop step response. On the right is the associated control effort.

Matlab code for the design of the controller is shown below.

```

1
2 % start with pid control designed in problem 10
3 figure(2), clf
4 bodemag(Plant*C_pid,logspace(-3,5))
5 hold on
6 grid on
7
8 % add constraints
9
10 % increase tracking by factor of 10 below omega_r=0.007
11 omega_r = 0.07; % reject input dist. below this frequency
12 gamma_r = 0.1; % amountn of input disturbance in output
13 w = logspace(log10(omega_r)-2,log10(omega_r));
14 Pmag=bode(Plant*C_pid,w);
15 for i=1:size(Pmag,3), Pmag_(i)=Pmag(1,1,i); end
16 plot(w,20*log10(1/gamma_r)*ones(1,length(Pmag_))...
17 +20*log10(Pmag_), 'g')
18
19 % attenuate noise by factor of gamma_n above omega_n
20 omega_n = 1000; % attenuate noise above this frequency
21 gamma_n = 0.1; % amount of noise attenuation in output
22 w = logspace(log10(omega_n),log10(omega_n)+2);
23 Pmag=bode(Plant*C_pid,w);
24 for i=1:size(Pmag,3), Pmag_(i)=Pmag(1,1,i); end
25 plot(w,20*log10(gamma_n)*ones(1,length(Pmag_))...
26 +20*log10(Pmag_), 'g')
27 figure(2), margin(Plant*C_pid)
28 %pause % hw_arm_compensator_design_1
29

```

CHAPTER 18. COMPENSATOR DESIGN

```

30 %%%%%%
31 % Control Design
32 C = C_pid;
33 %%%%%%
34
35 % phase lag (|p|<|z|): add gain at low frequency
36 % (tracking, dist rejection)
37 % low frequency gain = K*z/p
38 % high frequency gain = K
39 z = .7;
40 p = z/10;
41 Lag = tf([1,z],[1,p]);
42 C = C*Lag;
43 figure(2), margin(Plant*C)
44 %pause % hw_arm_compensator_design_2
45
46 % phase lead (|p|>|z|): increase PM (stability)
47 % low frequency gain = K*z/p
48 % high frequency gain = K
49 wmax = 30; % location of maximum frequency bump
50 M = 10; % separation between zero and pole
51 Lead = tf(M*[1,wmax/sqrt(M)],[1,wmax*sqrt(M)]);
52 C = C*Lead;
53 figure(2), margin(Plant*C)
54 %pause % hw_arm_compensator_design_3
55
56 % low pass filter: decrease gain at high frequency (noise)
57 p = 50;
58 LPF = tf(p,[1,p]);
59 C = C*LPF;
60 figure(2), margin(Plant*C)
61 %pause % hw_arm_compensator_design_4
62
63 % low pass filter: decrease gain at high frequency (noise)
64 p = 150;
65 LPF = tf(p,[1,p]);
66 C = C*LPF;
67 figure(2), margin(Plant*C)
68 %pause % hw_arm_compensator_design_5
69
70 %%%%%%
71 % add a prefilter to eliminate the overshoot
72 %%%%%%
73 F = 1;
74 % low pass filter
75 p = 3;
76 LPF = tf(p,[1,p]);
77 F = F*LPF;
78
79 %%%%%%
80 % Create plots
81 %%%%%%
82 % Open-loop tranfer function
83 OPEN = Plant*C;
84 % closed loop transfer function from R to Y
85 CLOSED_R_to_Y = (Plant*C/(1+Plant*C));
86 % closed loop transfer function from R to U

```

```

87  CLOSED_R_to_U = (C/(1+C*Plant));
88
89  figure(3), clf
90      subplot(1,3,1),
91          bodemag(CLOSED_R_to_Y), hold on
92          bodemag(CLOSED_R_to_Y*F)
93          title('Closed Loop Bode Plot'), grid on
94      subplot(1,3,2),
95          step(CLOSED_R_to_Y), hold on
96          step(CLOSED_R_to_Y*F)
97          title('Closed loop step response'), grid on
98      subplot(1,3,3),
99          step(CLOSED_R_to_U), hold on
100         step(CLOSED_R_to_U*F)
101         title('Control effort for step response'), grid on
102
103 %%%%%%%%%%%%%%
104 % Convert controller to state space equations
105 %%%%%%%%%%%%%%
106 [num,den] = tfdata(C, 'v');
107 [P.A_C, P.B_C, P.C_C, P.D_C]=tf2ss(num,den);
108
109 [num,den] = tfdata(F, 'v');
110 [P.A_F, P.B_F, P.C_F, P.D_F] = tf2ss(num,den);

```

18.3 Design Study B. Inverted Pendulum



Homework Problem B.18

For this homework assignment we will use the loopshaping design technique to design a successive loop closure controller for the inverted pendulum.

- First consider the inner loop, where $P_{in}(s)$ is the transfer function of the inner loop derived in HW B.5. Using a proportional and phase-lead controller, design $C_{in}(s)$ to stabilize the system with a phase margin close to 60 deg, and to ensure that noise above $\omega_{no} = 200$ rad/s is rejected by $\gamma_{no} = 0.1$. We want the closed-loop bandwidth of the inner loop to be approximately 25 rad/s. Note that since the gain on $P_{in}(s)$ is negative, the proportional gain will also need to be negative.
- Now consider the design of the controller for the outer loop system. The 'plant' for the design of the outer loop controller is

$$P = P_{out} \frac{P_{in} C_{in}}{1 + P_{in} C_{in}}.$$

Design the outer loop controller C_{out} so that the system is stable with phase margin close to $PM = 60$ degrees, and so that reference signals with frequency below $\omega_r = 0.01$ radians/sec are tracked with error $\gamma_r = 0.0001$,

and noise with frequency content above $\omega_{no} = 400$ radians/sec are rejected with $\gamma_{no} = 0.001$. Design the crossover frequency so that the rise time of the system is about 2 sec.

Solution

Figure 18.21 shows the Bode plot of the plant $P_{in}(s)$ together with the design specification on the noise attenuation.

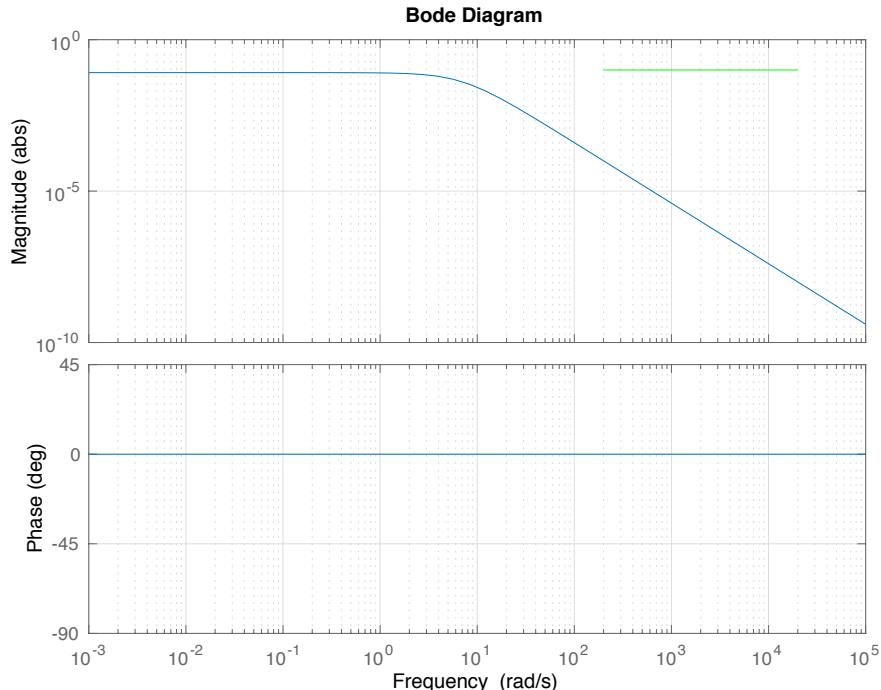


Figure 18.20: The Bode plot for the inner loop plant in HW B.18, together with the design specification.

The first step is to add a negative proportional gain of 1 to account for the negative sign in the plant transfer function. Once this is applied we can see that the phase fo the plant alone is -180 deg over all frequencies so that the open loop response at low frequency is above 0 dB. Figure 18.21 shows the corresponding Bode plot for a proportional gain of $K = -1$.

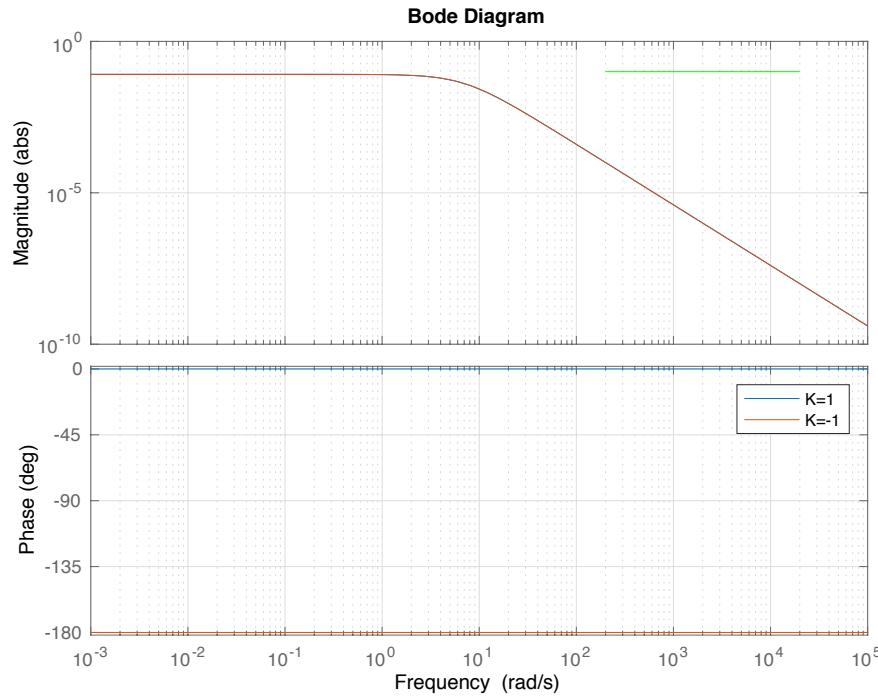


Figure 18.21: The Bode plot for the inner loop plant in HW B.18, with gain of -1 applied.

We want the bandwidth of the inner loop to be approximately 25 rad/s and so we will set the crossover frequency to be 25 rad/s. To stabilize the system, we will add lead compensation centered at 25 rad/s with a lead ratio of 13.9 that corresponds to the desired phase margin of 60 deg. Figure 18.22 shows the addition of the phase lead filter

$$C_{lead} = -45.1 \left(\frac{\frac{s}{6.70} + 1}{\frac{s}{93.3} + 1} \right),$$

which has a phase margin of $PM = 60$ degrees. The gain 45.1 sets the crossover frequency to be 25 rad/s. The lead compensation alone also satisfies the noise specification.

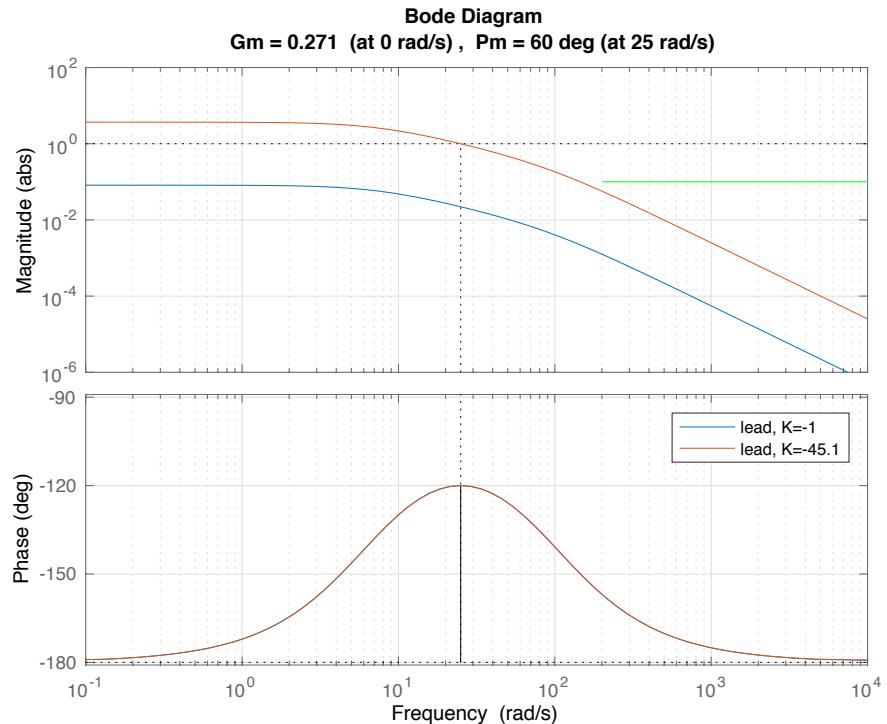


Figure 18.22: The Bode plot for the inner loop system in HW B.18, with proportional gain and phase lead compensation.

Notice that the DC-gain is not equal to one. The closed-loop magnitude response for the inner-loop subsystem

$$\frac{P_{in}C_{in}}{1 + P_{in}C_{in}},$$

as well as the unit step response for the output and control signal are all shown in Figure 18.23.

CHAPTER 18. COMPENSATOR DESIGN

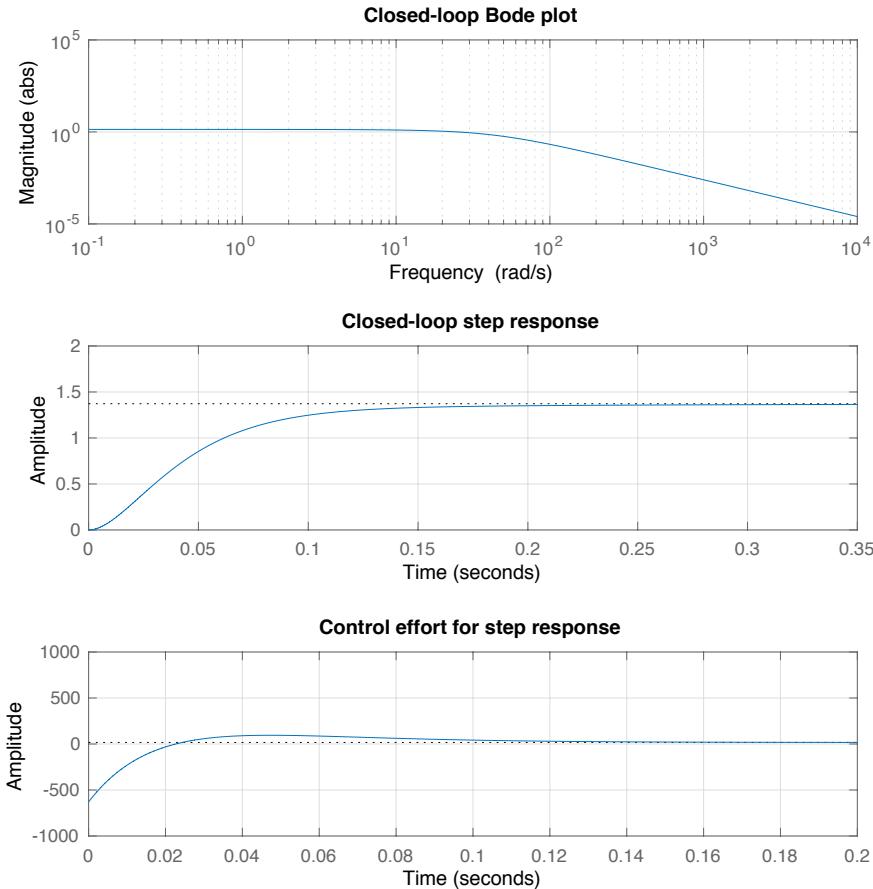


Figure 18.23: The closed loop bode response, the unit step response for the output, and the unit step response for the input of the inner loop design in HW B.18.

The Matlab code used to design the inner loop is shown below.

```

1 param
2
3 Plant = P_in;
4 figure(2), clf, hold on
5
6 % Define Design Specifications
7 %—— noise specification ——
8 omega_n = 200; % attenuate noise above this frequency
9 gamma_n = 0.1; % attenuate noise by this amount
10 w = logspace(log10(omega_n),2+log10(omega_n));
11 plot(w,gamma_n*ones(size(w)), 'g')
12
```

CHAPTER 18. COMPENSATOR DESIGN

```

14     figure(2), bode(Plant,logspace(-3,5)), grid on
15
16 %%%%%%
17 % Control Design
18 C = 1;
19 %%%%%%
20
21 % proportional control: correct for negative sign in plant
22 K = -1;
23 C = C*K;
24 figure(2), bode(Plant*C)
25 legend('K=1','K=-1')
26
27 % phase lead: increase PM (stability)
28 w_max = 25; % location of maximum frequency bump (desired crossover)
29 phi_max = 60*pi/180;
30 M = (1+sin(phi_max))/(1-sin(phi_max)); % lead ratio
31 z = w_max/sqrt(M)
32 p = w_max*sqrt(M)
33 Lead = tf([1/z 1],[1/p 1]);
34 C = C*Lead;
35 figure(2), bode(Plant*C), hold on, grid % update plot
36
37 % find gain to set crossover at w_max = 25 rad/s
38 [m,p] = bode(Plant*C,25);
39 K = 1/m;
40 C = K*C;
41 figure(2), margin(Plant*C) % update plot
42 legend('lead, K=1','lead, K=-45.1')
43
44 %%%%%%
45 % Create plots for analysis
46 %%%%%%
47
48 % Open-loop tranfer function
49 OPEN = Plant*C;
50 % closed loop transfer function from R to Y
51 CLOSED_R_to_Y = minreal((Plant*C/(1+Plant*C)));
52 % closed loop transfer function from R to U
53 CLOSED_R_to_U = minreal((C/(1+C*Plant)));
54
55 figure(3), clf
56 subplot(3,1,1),
57 bodemag(CLOSED_R_to_Y)
58 title('Closed-loop Bode plot'), grid on
59 subplot(3,1,2),
60 step(CLOSED_R_to_Y)
61 title('Closed-loop step response'), grid on
62 subplot(3,1,3),
63 step(CLOSED_R_to_U)
64 title('Control effort for step response'), grid on
65
66 %%%%%%
67 % Convert controller to state space equations for implementation
68 %%%%%%
69 [num,den] = tfdata(C,'v');
70 [P.Ain_C,P.Bin_C,P.Cin_C,P.Din_C]=tf2ss(num,den);

```

```

71
72 %%%%%%
73 % Convert controller to discrete transfer functions for implementation
74 %%%%%%
75 C_in_d = c2d(C,P.Ts,'tustin')
76 [P.Cin_d_num,P.Cin_d_den] = tfdata(C_in_d,'v');
77
78 C_in = C;

```

For the outer loop design, Figure 18.24 shows the Bode plot of the plant $P = P_{out} \frac{P_{in}C_{in}}{1+P_{in}C_{in}}$ together with the design specification on reference tracking and noise attenuation.

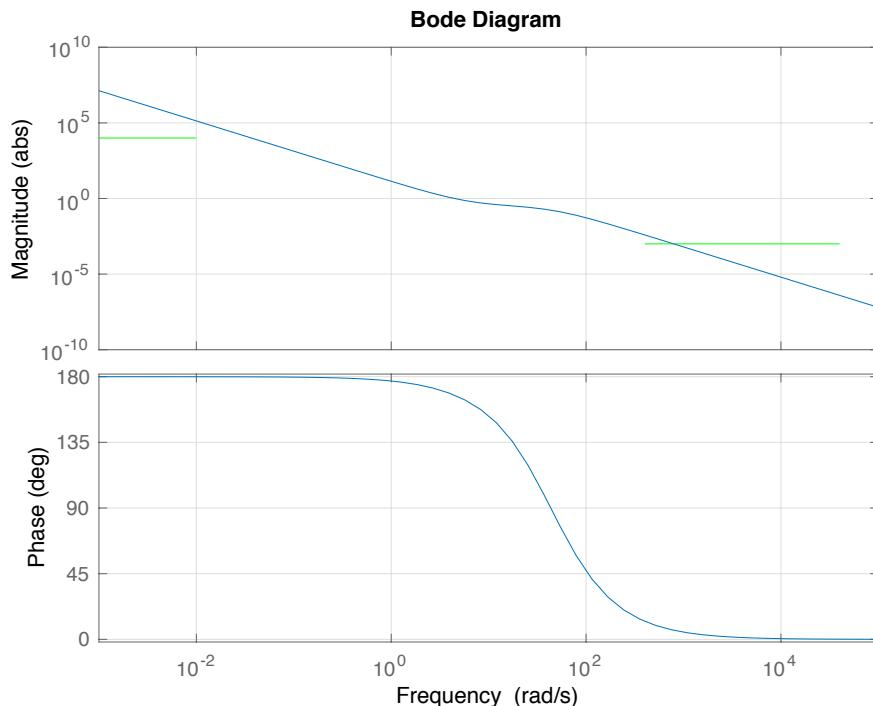


Figure 18.24: The Bode plot for the outer loop plant in HW B.18, together with the design specification.

We want a rise time of 2 sec for the system, so this leads us to select a crossover frequency of 1.1 rad/s. At 1.1 rad/s, the plant alone has a small negative phase margin (remember that 180 deg and -180 deg are equivalent), so to bring the phase margin up above 60 deg, we will use a lead compensator centered at 1.1 rad/s with a lead ratio of 32 to add 70 deg of phase. (A lead ratio of 32 is large. An alternative would be to add two leads that each contribute 35 deg of phase margin.) As shown in Figure 18.25, with the lead applied, the crossover frequency is pushed out beyond 100 rad/s. By applying a proportional gain of 0.0154, we set the crossover at 1.1 rad/s.

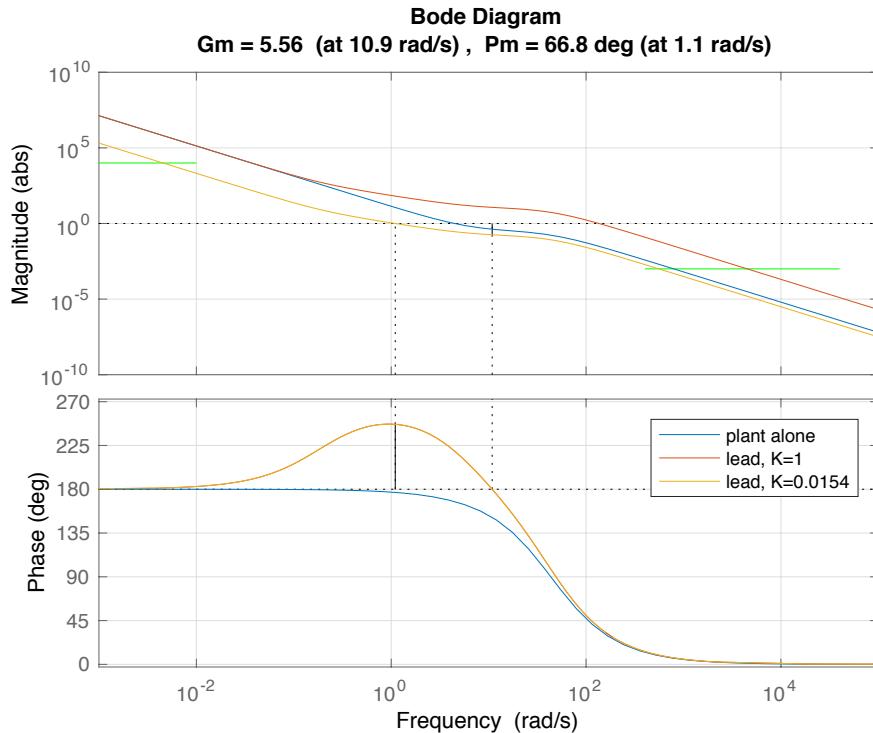


Figure 18.25: The Bode plot for the outer loop system in HW B.18, with lead compensation and proportional gain.

As can be seen from Figure 18.25, with the lead applied and crossover set at 1.1 rad/s, neither the tracking constraint nor the noise attenuation constraint are satisfied. We can satisfy the tracking constraint by applying lag compensation to boost the low-frequency gain near the constraint. Checking the gain of the lead compensated system at the tracking constraint frequency ($\omega_r = 0.01$ rad/s), we can see that the gain is too low by a factor of 4.8. Applying a lag compensator with a lag ratio of 8 ensures that this tracking constraint is satisfied as shown in Figure 18.26.

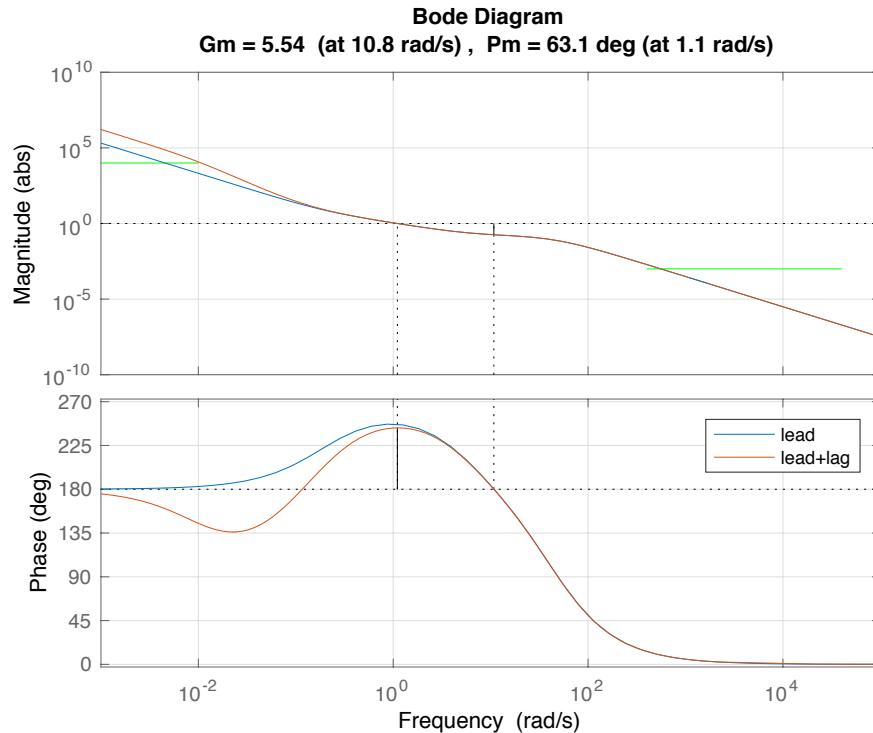


Figure 18.26: The Bode plot for the outer loop system in HW B.18, with lead, and lag control.

Finally, we can apply a low-pass filter to reduce the effects of sensor noise in the system and satisfy the noise attenuation constraint. To satisfy the constraint, we need to drop the gain of the open-loop system by a factor of 2 at $\omega_{no} = 400$ rad/s. We can accomplish this with a low-pass filter with a cut-off frequency of 230 rad/s. This is shown in Figure 18.27 below.

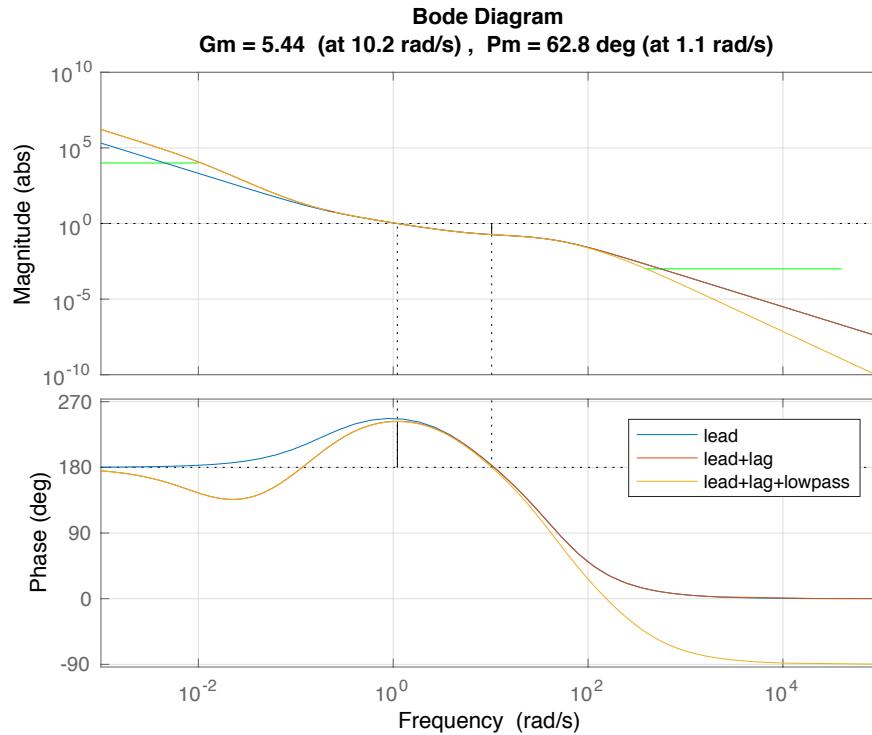


Figure 18.27: The Bode plot for the outer loop system in HW B.18, with lead, lag, and low-pass compensation applied.

The resulting compensator is

$$C_{out}(s) = 0.50 \left(\frac{s + 0.194}{s + 6.24} \right) \left(\frac{s + 0.08}{s + 0.01} \right) \left(\frac{230}{s + 230} \right).$$

The closed-loop response for both the inner and outer loop systems, as well as the unit-step response for the output and control signal of the outer loops are all show in Figure 18.28.

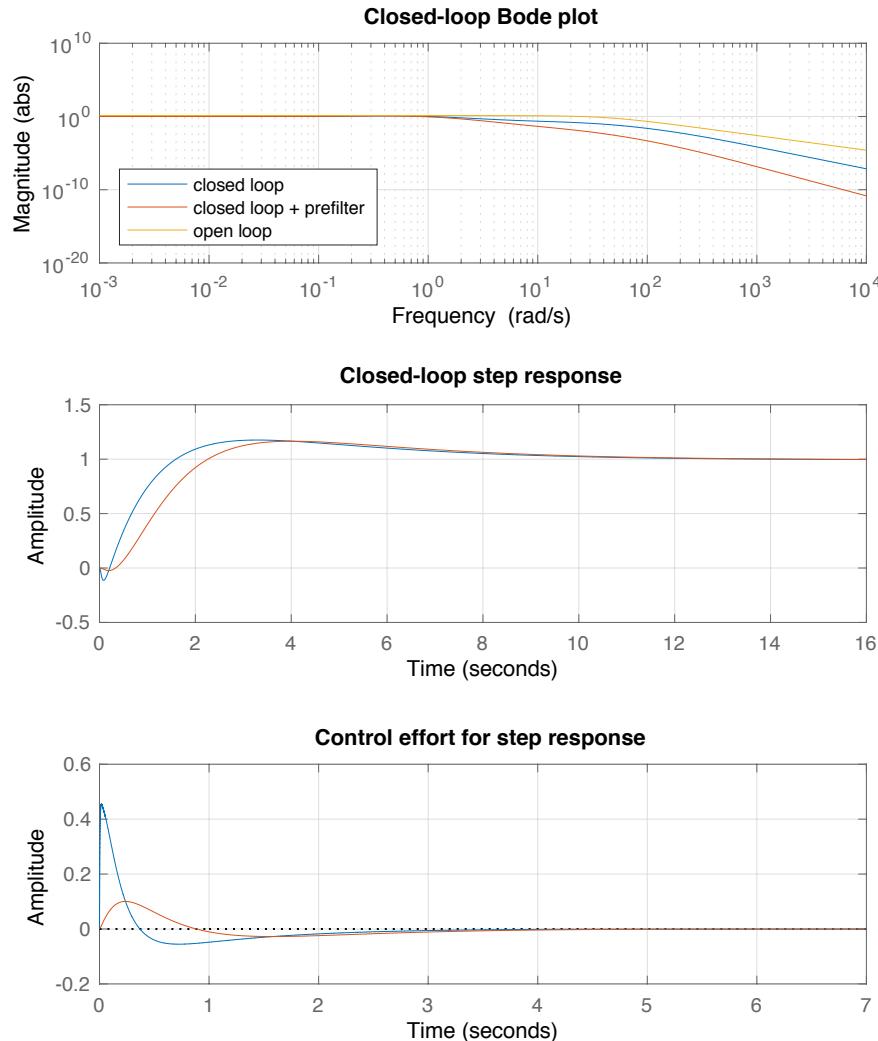


Figure 18.28: The closed-loop bode response, the unit-step response for the output, and the unit step response for the input of the inner loop design in HW B.18.

A prefilter

$$F(s) = \frac{2}{s+2}$$

has been added to the system. Note the significant decrease in the control effort required when prefiltering is used with the step input.

The Matlab code used to design the outer loop is shown below.

```
1 Plant = minreal(P_out*(P_in*C_in/(1+P_in*C_in)));
```

CHAPTER 18. COMPENSATOR DESIGN

```

2 figure(2), clf
3 hold on
4 grid on
5
6 %%%%%%
7 % Define Design Specifications
8 %%%%%%
9
10 %—— general tracking specification ——
11 omega_r = 0.01; % track signals below this frequency
12 gamma_r = 0.0001; % tracking error below this value
13 w = logspace(log10(omega_r)-2,log10(omega_r));
14 plot(w,(1/gamma_r)*ones(size(w)), 'g')
15
16 %—— noise specification ——
17 omega_n = 400; % attenuate noise above this frequency
18 gamma_n = 0.001; % attenuate noise by this amount
19 w = logspace(log10(omega_n),2+log10(omega_n));
20 plot(w,gamma_n*ones(size(w)), 'g')
21
22 %%%%%%
23 % Control Design
24 C = tf([1],[1]);
25 %%%%%%
26
27 % phase lead: increase PM (stability)
28 % At desired crossover frequency, PM = -3
29 % Add 70 deg of PM with lead
30 w_max = 1.1; % location of maximum frequency bump (desired crossover)
31 phi_max = 70*pi/180;
32 M = (1+sin(phi_max))/(1-sin(phi_max)) % lead ratio
33 z = w_max/sqrt(M)
34 p = w_max*sqrt(M)
35 Lead = tf([1/z 1],[1/p 1]);
36 C = C*Lead;
37
38 % find gain to set crossover at w_max = 1.1 rad/s
39 [m,p] = bode(Plant*C,1.1);
40 K = 1/m;
41 C = K*C;
42
43 % Tracking constraint not satisfied — add lag compensation to boost low-frequency gain
44
45 % Find gain increase needed at omega_r
46 [m,p] = bode(Plant*C,omega_r);
47 gain_increase_needed = 1/gamma_r/m
48 % Minimum gain increase at low frequencies is 4.8. Let lag ratio be 8.
49 M = 8;
50 p = omega_r; % Set pole at omega_r
51 z = M*p; % Set zero at M*omega_r
52 Lag = tf(M*[1/z 1],[1/p 1]);
53 C = C*Lag;
54
55 % Noise attenuation constraint not quite satisfied
56 % Can be satisfied by reducing gain at 400 rad/s by a factor of 2
57 % Use a low-pass filter
58 m = 0.5; % attenuation factor

```

```

59         a = m*omega_n*sqrt(1/(1-m^2));
60         lpf = tf(a,[1 a]);
61         C = lpf*C;
62
63 %%%%%%%%%%%%%%
64 % Prefilter Design
65 F = tf([1],[1]);
66 %%%%%%%%%%%%%%
67
68 % low pass filter
69 p = 2; % frequency to start the LPF
70 LPF = tf(p,[1 p]);
71 F = F*LPF
72
73 %%%%%%%%%%%%%%
74 % Convert controller to state space equations for implementation
75 %%%%%%%%%%%%%%
76 C=minreal(C);
77 [num,den] = tfdata(C,'v');
78 [P.Aout_C,P.Bout_C,P.Cout_C,P.Dout_C]=tf2ss(num,den);
79
80 [num,den] = tfdata(F,'v');
81 [P.Aout_F, P.Bout_F, P.Cout_F, P.Dout_F] = tf2ss(num,den);
82
83 %%%%%%%%%%%%%%
84 % Convert controller to discrete transfer functions for implementation
85 %%%%%%%%%%%%%%
86 C_out_d = c2d(C,P.Ts,'tustin')
87 [P.Cout_d_num,P.Cout_d_den] = tfdata(C_out_d,'v');
88
89 F_d = c2d(F,P.Ts,'tustin');
90 [P.F_d_num,P.F_d_den] = tfdata(F_d,'v');
91
92 C_out = C;

```

See <http://controlbook.byu.edu> for the complete solution.

18.4 Design Study C. Satellite Attitude Control



Homework Problem C.18

For this homework assignment we will use the loopshaping design technique to design a successive loop closure controller for the satellite attitude problem.

- (a) First consider the inner loop, where $P_{in}(s)$ is the transfer function of the inner loop derived in HW C.5. The Bode plot for this system shows an underdamped resonate mode. We have seen in previous chapters that this resonate mode can be removed by introducing rate feedback of the form

$$u = -k_D \dot{y} + u',$$

where y is the output signal and u' is the additional control signal to be designed. Using the derivative gain k_{D_θ} found in HW C.10, find the

revised transfer function from τ' to θ , and use this transfer function for $P_{in}(s)$. Letting

$$\tau'(s) = C_{in}(s)E(s),$$

where $e(t) = \theta_r(t) - \theta(t)$, design $C_{in}(s)$ to meet the following specifications:

- The inner loop will track θ_r with frequency content below $\omega_r = 0.01$ radians/sec to within $\gamma_r = 0.01$.
- The inner loop will attenuate noise on the measurement of θ for all frequencies above $\omega_n = 20$ rad/sec by $\gamma_n = 0.01$.
- The phase margin of the inner loop should be around $PM = 60$ degrees.

- (b) Now consider the design of the controller for the outer loop system. As seen from HW C.5, the transfer function for the outer loop also has a strong resonate mode. Similar to the inner loop, use the derivative gain found in HW C.10 to introduce rate damping, and find the associated transfer function $P_{out}(s)$. The 'plant' for the design of the outer loop controller is

$$P = P_{out} \frac{P_{in}C_{in}}{1 + P_{in}C_{in}}.$$

Design the outer loop controller C_{out} to meet the following specs:

- Track steps in ϕ_r with zero steady state error.
- Reject input disturbances with frequency content below $\omega_{d_{in}} = 0.01$ by $\gamma_{d_{in}} = 0.1$.
- Attenuate noise on the measurement of ϕ with frequency content above $\omega_n = 10$ rad/sec by $\gamma_n = 10^{-4}$.
- The phase margin of the outer loop should be around $PM = 60$ degrees.
- Use a prefilter to reduce any peaking in the closed loop response of the outer loop.

Solution

From HW C.5, the transfer function model for the inner loop is given by

$$\Theta(s) = \frac{\frac{1}{J_s}}{s^2 + \frac{b}{J_s}s + \frac{k}{J_s}}\tau(s).$$

Rate damping is added by designing the control signal so that

$$\tau(s) = -k_D \frac{s}{\sigma s + 1} \Theta(s) + \tau'(s).$$

Solving for $\Theta(s)$ in terms of $\tau'(s)$ gives

$$\Theta(s) = \frac{\sigma s + 1}{\sigma J_s s^3 + (\sigma b + J_s)s^2 + (\sigma k + b + k_D)s + k} \tau'(s).$$

CHAPTER 18. COMPENSATOR DESIGN

Therefore, for the inner loop we will set

$$P_{in} = \frac{\sigma s + 1}{\sigma J_s s^3 + (\sigma b + J_s)s^2 + (\sigma k + b + k_D)s + k}.$$

Figure 18.29 shows the Bode plot of the plant $P_{in}(s)$ together with the design specification on input tracking and noise attenuation.

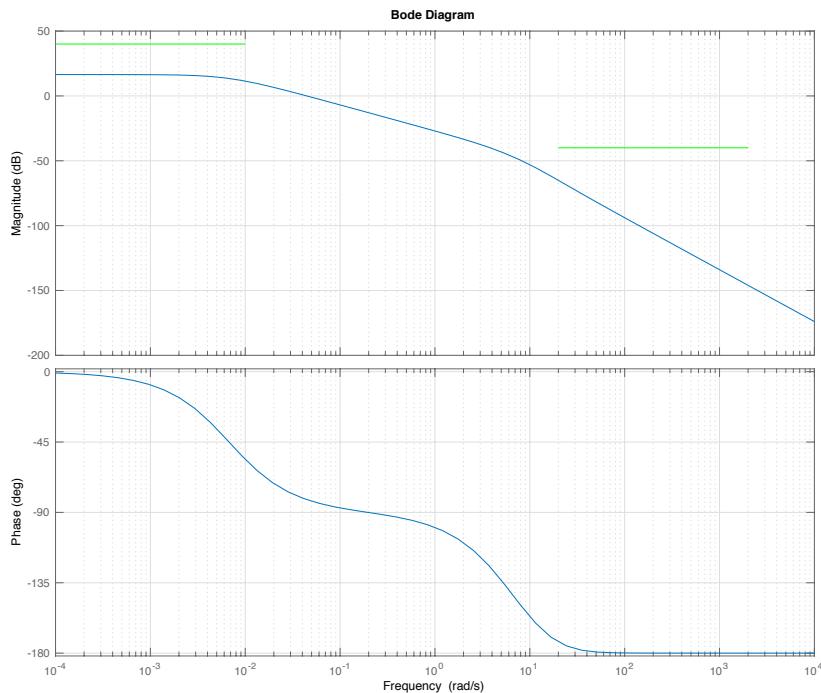


Figure 18.29: The Bode plot for the inner loop plant in HW C.18, together with the design specification.

The first step is to add a proportional gain to satisfy the tracking requirement. Figure 18.30 shows the corresponding Bode plot for a proportional gain of $k_P = 30$.

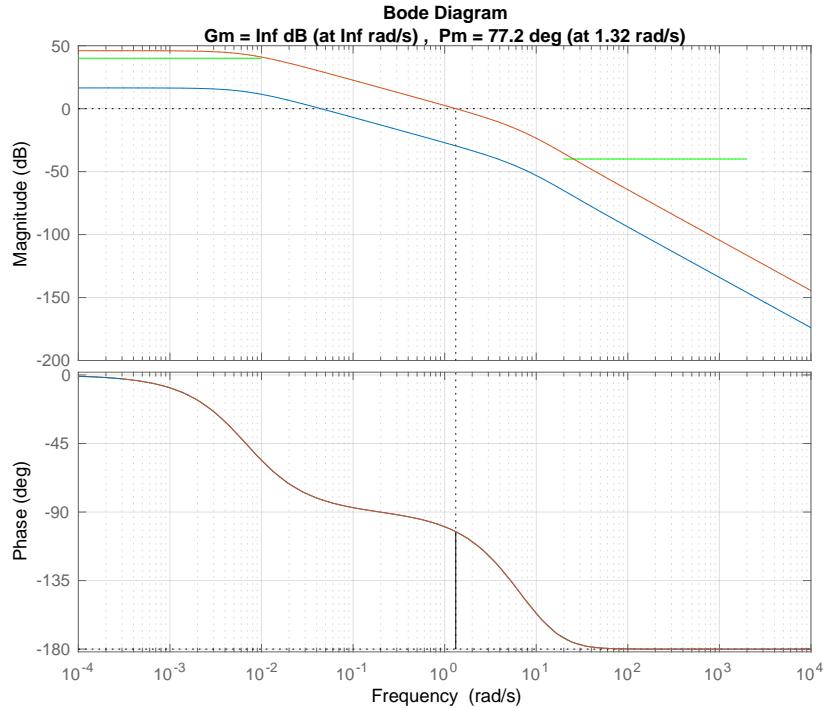


Figure 18.30: The Bode plot for the inner loop plant in HW C.18, with proportional gain.

Since the phase margin is over $PM = 60$ degrees, we add a low pass filter. Figure 18.31 shows the loop gain with the addition of the low pass filter

$$C_{LPF} = \frac{10}{s + 10},$$

which has a phase margin of $PM = 69.8$ degrees, and satisfies the noise specification.

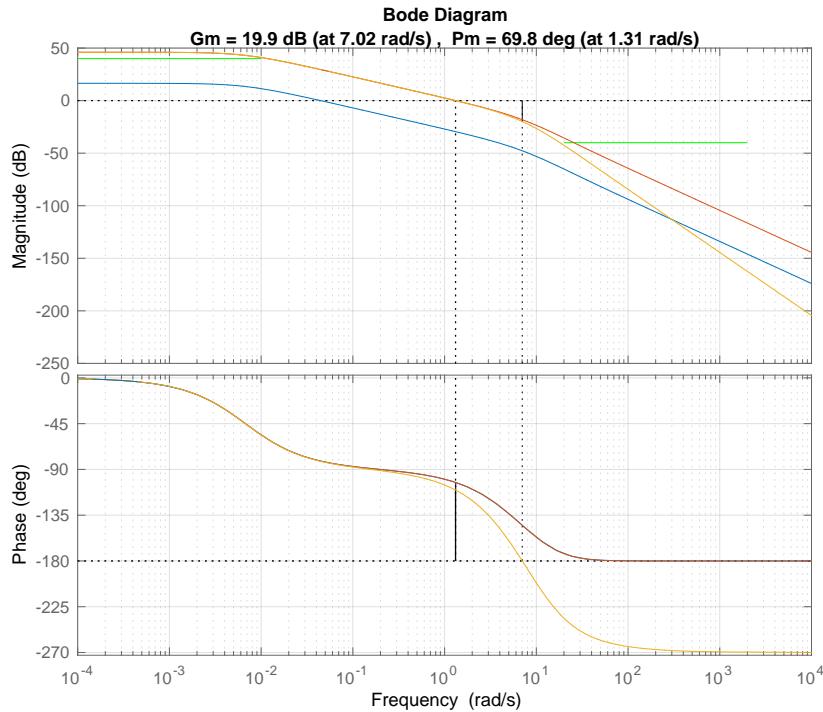


Figure 18.31: The Bode plot for the inner loop system in HW ???.18, with proportional gain and low pass filter.

The resulting compensator is

$$C_{in}(s) = 30 \left(\frac{10}{s + 10} \right).$$

Notice that the DC-gain is not equal to one. The closed loop response for the inner subsystem

$$\frac{P_{in}C_{in}}{1 + P_{in}C_{in}},$$

as well as the unit step response for the output and control signal are all shown in Figure 18.32.

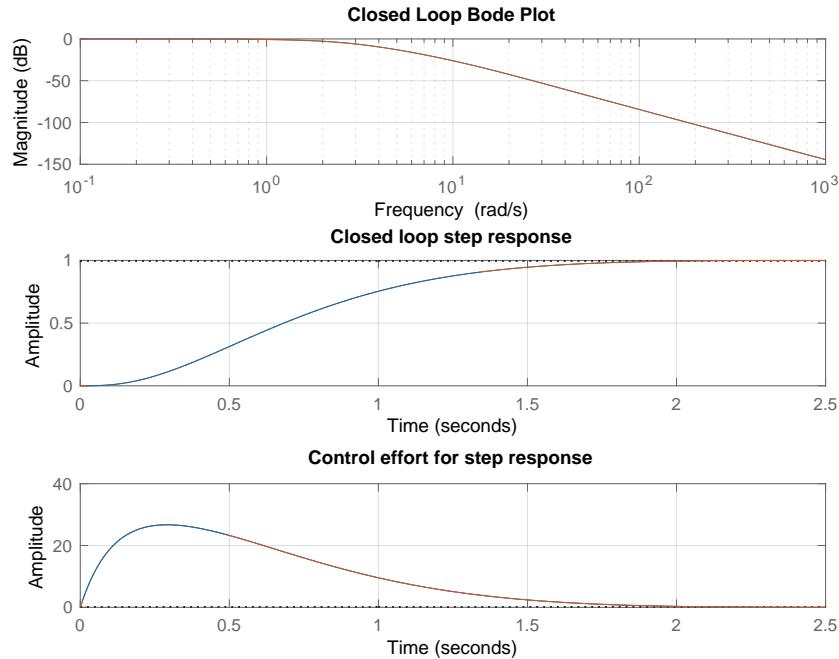


Figure 18.32: The closed loop bode response, the unit step response for the output, and the unit step response for the input of the inner loop design in HW C.18.

The Matlab code used to design the inner loop is shown below.

```

1 Plant = P_in;
2
3 %%%%%%
4 % Define Design Specifications
5 %%%%%%
6 %—— general tracking specification ——
7 omega_r = 10^-2; % track signals below this frequency
8 gamma_r = 10^(-40/20); % tracking error below this value
9 w = logspace(log10(omega_r)-2,log10(omega_r));
10
11 %—— noise specification ——
12 omega_n = 2*10^1; % attenuate noise above this frequency
13 gamma_n = 10^(-40/20); % attenuate noise by this amount
14 w = logspace(log10(omega_n),2+log10(omega_n));
15
16 %%%%%%
17 % Control Design
18 C = 1;
19 %%%%%%
20
21 % proportional control: change cross over frequency

```

CHAPTER 18. COMPENSATOR DESIGN

```

22      kp = 30;
23      C = C*kp;
24
25 % low pass filter: decrease gain at high frequency (noise)
26      p = 10;
27      LPF = tf(p,[1,p]);
28      C = C*LPF;
29
30 %%%%%%%%%%%%%%
31 % Convert controller to state space equations
32 %%%%%%%%%%%%%%
33 [num,den] = tfdata(C,'v');
34 [P.Ain_C,P.Bin_C,P.Cin_C,P.Din_C]=tf2ss(num,den);
35
36 C_in = C;

```

The transfer function model for the outer loop was derived in HW C.5 to be

$$\Phi(s) = \frac{\frac{b}{J_p}s + \frac{k}{J_p}}{s^2 + \frac{b}{J_p}s + \frac{k}{J_p}} \Theta^r(s).$$

Rate damping is added by designing the control signal so that

$$\Theta^r(s) = -k_D \frac{s}{\sigma s + 1} \Phi(s) + \Theta^{r'}(s).$$

Solving for $\Phi(s)$ in terms of $\Theta^{r'}(s)$ gives

$$\Phi(s) = \frac{\sigma b s^2 + (\sigma k + b)s + k}{\sigma J_p s^3 + (\sigma b + J_p + b k_D)s^2 + (\sigma k + b + k k_D)s + k} \Theta^{r'}(s).$$

The plant for the outer loop is therefore

$$P_{out} = \frac{\sigma s + 1}{\sigma J_s s^3 + (\sigma b + J_s)s^2 + (\sigma k + b + k_D)s + k} \frac{P_{in} C_{in}}{1 + P_{in} C_{in}}.$$

Figure 18.33 shows the Bode plot for P_{out} together with the design specification on rejecting input disturbances and attenuating the noise. The requirement that steady state error to a step requires that the slope as $s \rightarrow 0$ is -20 dB/dec, and is not shown in Figure 18.33.

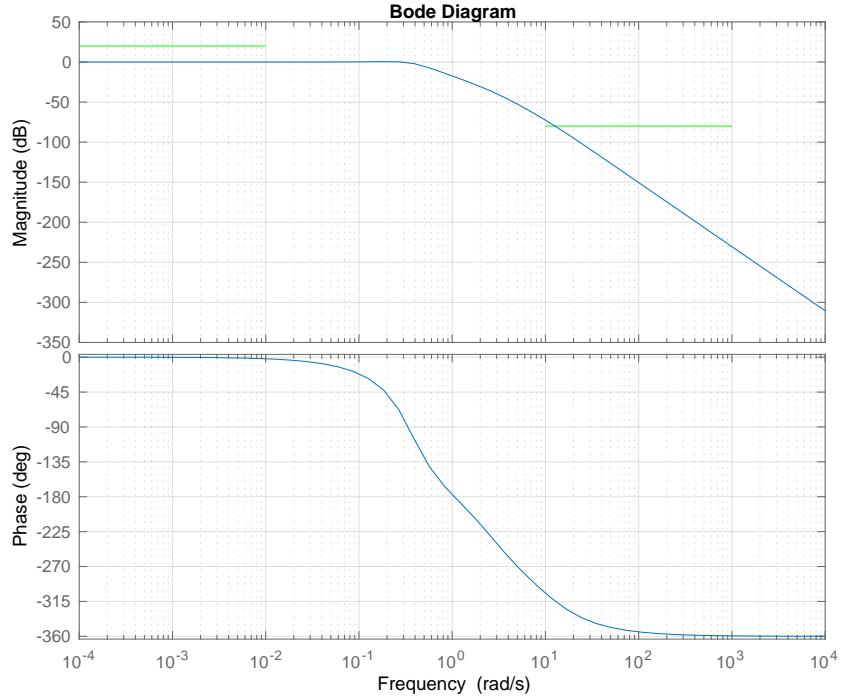


Figure 18.33: The Bode plot for the outer loop plant in HW C.18, together with the design specification.

The first step is to add an integrator to meet the steady state tracking requirement. Figure 18.34 shows the loop gain after adding the integral control

$$C_{int} = \frac{s + 0.4}{s}.$$

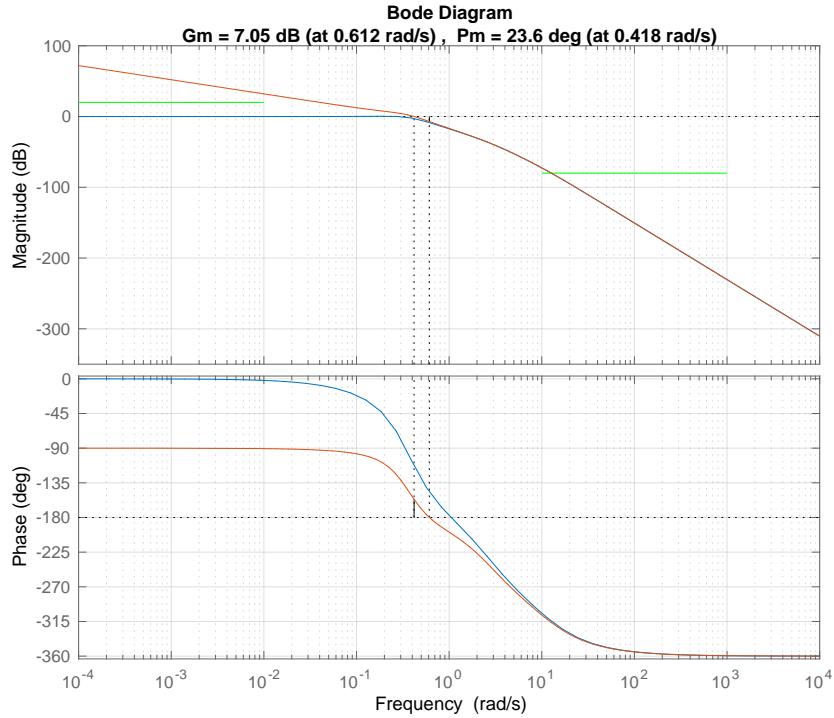


Figure 18.34: The Bode plot for the outer loop system in HW C.18, with integral control.

It is clear from Figure 18.34 that the input disturbance requirement is satisfied with a large buffer. To reduce the bandwidth of the system, and to make it easier to satisfy the noise attenuation requirement, a proportional gain of $k_P = 0.5$ is added to the compensator. The result is shown in Figure 18.35

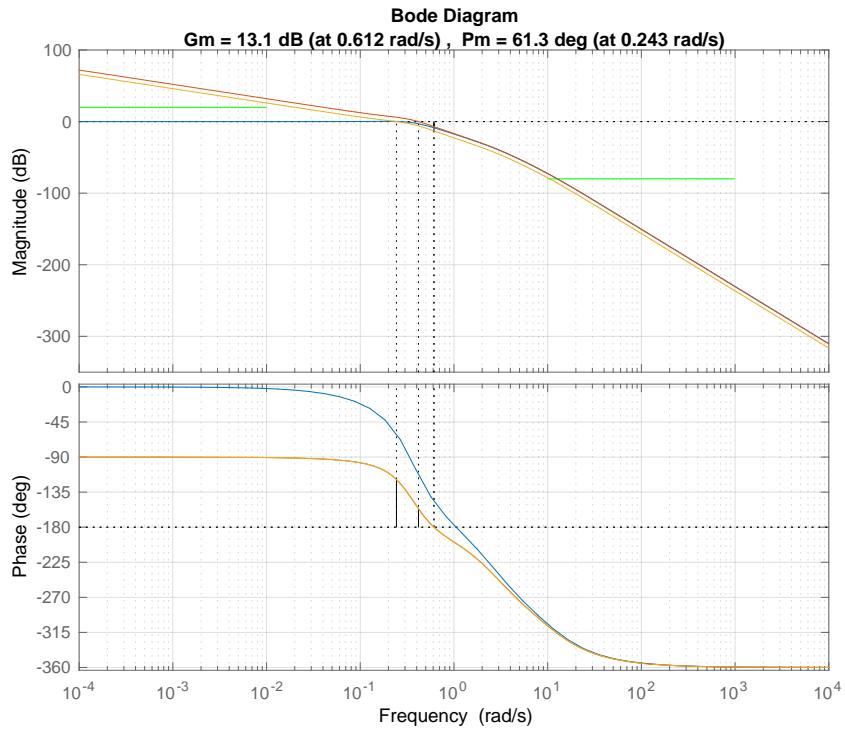


Figure 18.35: The Bode plot for the outer loop system in HW C.18, with proportional and integral control.

To meet the noise specification, the low pass filter

$$C_{lpf} = \frac{3}{s + 3}$$

is added to the compensator, and the resulting loopgain is shown in Figure 18.36.

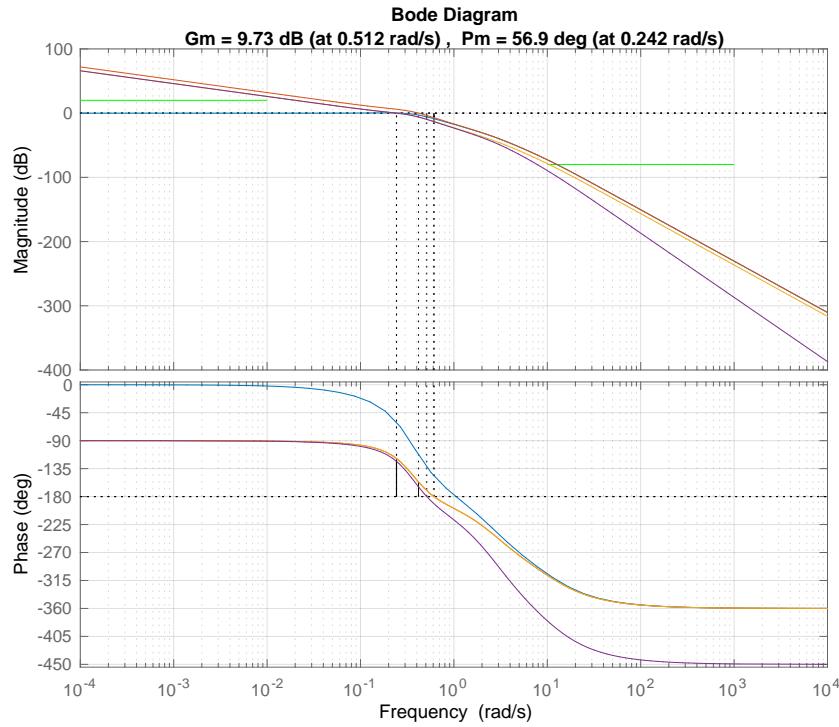


Figure 18.36: The Bode plot for the outer loop system in HW C.18, with proportional gain, integral control, and a low pass filter.

Since the phase margin is $PM = 56.9$ degrees, we consider the phase margin specification to be satisfied. The resulting compensator is

$$C_{out}(s) = 0.5 \left(\frac{s + 0.4}{s} \right) \left(\frac{3}{s + 3} \right).$$

The closed loop response for both the inner and outer loop systems, as well as the unit step response for the output and control signal of the outer loops are all shown in Figure 18.37, where the prefilter

$$F(s) = \frac{0.8}{s + 0.8}$$

has been added to reduce the peaking in the closed loop response.

CHAPTER 18. COMPENSATOR DESIGN

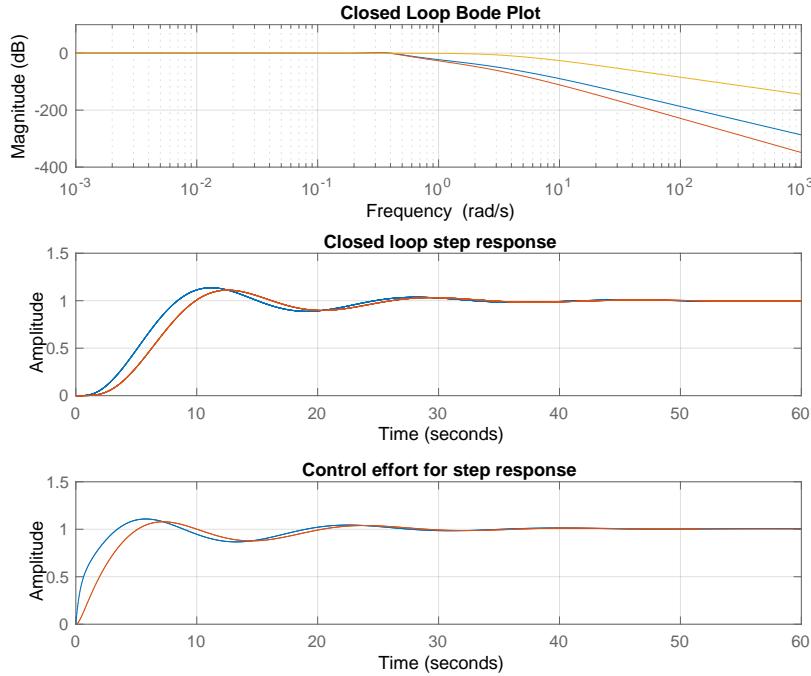


Figure 18.37: The closed loop bode response, the unit step response for the output, and the unit step response for the input of the inner loop design in HW C.18.

The Matlab code used to design the outer loop is shown below.

```

1 Plant = minreal(P_out*(P_in*C_in/(1+P_in*C_in)));
2
3 %%%%%%
4 % Define Design Specifications
5 %%%%%%
6
7 %—— input disturbance specification ——
8 omega_din = 10^-2;
9 gamma_din = 0.1;
10 w = logspace(log10(omega_din)-2,log10(omega_din));
11 Pmag=bode(Plant,w);
12 for i=1:size(Pmag,3), Pmag_(i)=Pmag(1,1,i); end
13
14 %—— noise specification ——
15 omega_n = 10^1;
16 gamma_n = 10^(-80/20);
17 w = logspace(log10(omega_n),2+log10(omega_n));
18 plot(w,20*log10(gamma_n)*ones(size(w)), 'g')
19
20 %%%%%%

```

```

21 % Control Design
22 C = tf([1],[1]);
23 %%%%%%
24 % integral control:
25 k_I = .4; % frequency at which integral action ends
26 Integrator = tf([1,k_I],[1,0]);
27 C = C*Integrator;
28
29 % proportional control: change cross over frequency
30 kp = .5;
31 C = C*kp;
32
33 % low pass filter: decrease gain at high frequency (noise)
34 p = 3;
35 LPF = tf(p,[1,p]);
36 C = C*LPF;
37
38 %%%%%%
39 % Prefilter Design
40 F = tf([1],[1]);
41 %%%%%%
42 % low pass filter
43 p = 0.8; % frequency to start the LPF
44 LPF = tf(p, [1,p]);
45 F = F*LPF;
46
47 %%%%%%
48 % Convert controller to state space equations
49 %%%%%%
50 C=minreal(C);
51 [num,den] = tfdata(C, 'v');
52 [P.Aout_C,P.Bout_C,P.Cout_C,P.Dout_C]=tf2ss(num,den);
53
54 [num,den] = tfdata(F, 'v');
55 [P.Aout_F, P.Bout_F, P.Cout_F, P.Dout_F] = tf2ss(num,den);
56
57 C_out=C;
58
59

```

See <http://controlbook.byu.edu> for the complete solution.

Notes and References

Part VI

Homework Problems

D Design Study: Mass Spring Damper

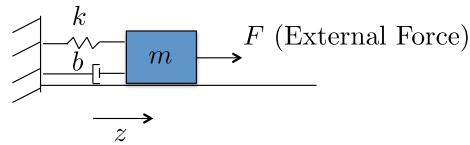


Figure 18.38: Mass Spring Damper

Figure 18.38 shows the mass-spring-damper system. The mass slides along a frictionless level surface and is connected to a wall by a spring with spring constant k and a damper with damping constant b . The position of the mass is given by z , where $z = 0$ is the position where the spring is not stretched. The speed of the mass is given by \dot{z} . An external force F is applied to the mass as shown in Figure 18.38.

Assume the following physical constants: $m = 5 \text{ kg}$, $k = 3 \text{ N/m}$, $b = 0.5 \text{ N-sec/m}$.

Homework Problems:

- | | |
|---|---|
| D.2 Kinetic energy. | D.d Root locus. |
| D.a Simulink animation. | D.10 Digital PID. |
| D.3 Equations of Motion. | D.11 Full state feedback. |
| D.b Simulink S-function. | D.12 Full state with integrator. |
| D.4 Linearize equations of motion. | D.13 Observer based control. |
| D.5 Transfer function model. | D.14 Disturbance observer. |
| D.6 State space model. | D.15 Frequency Response. |
| D.7 Pole placement using PD. | D.16 Loop gain. |
| D.8 Second order design. | D.17 Stability margins. |
| D.9 Integrators and system type. | D.18 Loopshaping design. |



Homework D.2

Using the configuration variable z , write an expression for the kinetic energy of the system.



Homework D.a

Create a simulink animation of the mass-spring-damper system. The input should be a slider for z . Turn in a screen capture of the animation.



Homework D.3

- (a) Find the potential energy for the system.
- (b) Define the generalized coordinates.
- (c) Find the generalized forces and damping forces.
- (d) Derive the equations of motion using the Euler-Lagrange equations.



Homework D.b

Modify the simulink, Matlab, or python model created in homework D.a by creating a function that implements the equations of motion. The input to the function should be a slider for force. The output should go to the animation developed in homework D.a.



Homework D.4

For the mass spring damper:

- (a) Find the equilibria of the system.
- (b) Linearize the system about the equilibria using Jacobian linearization.
- (c) Linearize the system using feedback linearization.



Homework D.5

For the equations of motion for the mass spring damper,

- (a) Using the Laplace transform, convert from time domain to the s-domain.
- (b) Find the transfer function from the input force F to the mass position z .
- (c) Draw the associated block diagram.



Homework D.6

For the equations of motion for the mass spring damper, define the states as $x = (z, \dot{z})^\top$, and the input as $u = F$, and the measured output as $y = z$. Find the linear state space equations in the form

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du.\end{aligned}$$



Homework D.7

- (a) Given the open loop transfer function from force to position in problem D.5, find the open loop poles of the system, when the equilibrium position is $z_e = 0$.
- (b) Using PD control architecture shown in Figure 7.2, find the closed loop transfer function from z_r to z and find the closed loop poles as a function of k_P and k_D .
- (c) Select k_P and k_D to place the closed loop poles at $p_1 = -1$ and $p_2 = -1.5$.
- (d) Using the gains from part (c), implement the PD control for the mass spring damper in Simulink and plot the response of the system to a 1 m step input.



Homework D.8

For the mass spring, do the following:

- (a) Suppose that the design requirements are that the rise time is $t_r \approx 2$ seconds, with a damping ratio of $\zeta = 0.7$. Find the desired closed loop characteristic polynomial $\Delta_{cl}^d(s)$, and the associated pole locations. Find the proportional and derivative gains k_P and k_D to achieve these specifications, and modify the Simulink simulation from HW D.7 to verify that the step response satisfies the requirements.
- (b) Suppose that the size of the input force is limited to $F_{\max} = 2$ N. Modify the Simulink diagram to include a saturation block on the torque F . Using the rise time t_r as a tuning parameter, tune the PD control gains so that the input just saturates when a step of size 1 meter is placed on z^r . Plot the step response showing that saturation does not occur for a 1 meter step input for the new control gains.



Homework D.9

- (a) When the controller for the mass spring damper is PD control, what is the system type? Characterize the steady state error when the reference input is a step, a ramp, and a parabola. How does this change if you add an integrator?
- (b) Consider the case where a constant disturbance acts at the input to the plant (for example an inaccurate knowledge of the spring constant in this case). What is the steady state error to a constant input disturbance when the integrator is not present, and when it is present?



Homework D.e

Adding an integrator to obtain PID control for the mass spring damper, put the characteristic equation in Evan's form and use the Matlab `rlocus` command to plot the root locus versus the integrator gain k_I . Select a value for k_I that does not significantly change the other locations of the closed loop poles.



Homework D.10

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Modify the system dynamics file so that the parameters m , k , and b vary by up to 20% of their nominal value each time they are run (uncertainty parameter = 0.2).

- (b) Rearrange the block diagram so that the controller is implemented as an m-function implemented at the sample rate of $T_s = 0.01$. The controller should only assume knowledge of the position z and the reference position z_r .
- (c) Implement the PID controller designed in Problems D.8 using an m-function called `mass_ctrl.m`. If your simulation is in Matlab or Python then implement the PID controller as a separate class or function. Use the dirty derivative gain of $\tau = 0.05$. Tune the integrator to remove the steady state error caused by the uncertain parameters.



Homework D.11

The objective of this problem is to implement state feedback controller for the mass spring damper using the full state. Start with the simulation files developed in Homework D.10.

- (a) Select the closed loop poles as the roots of the equations $s^2 + 2\zeta\omega_n + \omega_n^2 = 0$ where ω_n , and ζ were found in Homework D.8.
- (b) Add the state space matrices A , B , C , D derived in Homework D.6 to your param file.
- (e) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}_{A,B}) = n$.
- (c) Find the feedback gain K such that the eigenvalues of $(A - BK)$ are equal to desired closed loop poles. Find the reference gain k_r so that the DC-gain from z_r to z is equal to one. Note that $K = (k_p, k_d)$ where k_p and k_d are the proportional and derivative gains found in Homework D.8. Why?
- (d) Modify the control code `mass_ctrl.m` to implement the state feedback controller. To construct the state $x = (z, \dot{z})^\top$ use a digital differentiator to estimate \dot{z} .



Homework D.12

- (a) Modify the state feedback solution developed in Homework D.11 to add an integrator with anti-windup to the feedback loop for z .
- (b) Add a constant input disturbance of 0.25 Newtons to the input of the plant and allow the plant parameters to vary up to 20%.
- (c) Tune the integrator pole (and other gains if necessary) to get good tracking performance.



Homework D.13

The objective of this problem is to design an observer that estimates the state of the system and to use the estimated state in the controller designed in Homework D.12.

- (a) Modify the simulink diagram from HW D.12 as described in the description of HW A.13 to add a plot of the true state x , and estimated state \hat{x} , and the observation error $x - \hat{x}$.
- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, without an input disturbance. Modify `mass_dynamics.m` so that the parameters known to the controller are the actual plant parameters (uncertainty parameter $\alpha = 0$).
- (d) In the control block, add an observer to estimate the state \hat{x} , and use the estimate of the state in your feedback controller. Tune the poles of the controller and observer to obtain good performance.
- (e) As motivation for the next chapter, add an input disturbance to the system of 0.25 and observe that there is steady state error in the observation error. In the next chapter we will show how to remove the steady state error in the observation error.



Homework D.14

- (a) Modify your solution from HW D.13 so that the uncertainty parameter in `mass_dynamics.m` is $\alpha = 0.2$, representing 20% inaccuracy in the knowledge of the system parameters, and so that the input disturbance is 0.25. Also, add noise to the output channels z_m and θ_m with standard deviation of 0.001.
- (b) Add a disturbance observer to the controller, and verify that the steady state error in the estimator has been removed. Tune the system to get good response.



Homework D.15

Draw by hand the Bode plot of the mass spring damper from force \tilde{F} to position \tilde{z} . Use the Matlab `bode` command and compare your results.



Homework D.16

For the mass spring, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, and (2) the plant under PID control, using the control gains calculated in Homework D.10.

- What is the tracking error to a unit ramp under PID control?
- If the frequency content of the input disturbance $d_{in}(t)$ is below $\omega_{d_{in}} = 0.1$ radians per second, what percentage of the input disturbance shows up in the output z under PID control?
- If all of the frequency content of the noise $n(t)$ is greater than $\omega_{no} = 100$ radians per second, what percentage of the noise shows up in the output signal z ?



Homework D.17

For the mass spring damper, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the closed loop system under PID control. On the same graph, plot the open loop Bode plot and the closed loop Bode plot. What is the bandwidth of the closed loop system, and how does this relate to the crossover frequency? Use the gains found in HW D.10.



Homework D.18

For this homework assignment we will use the loopshaping design technique to design a controller for the mass spring damper problem with dynamics given by

$$Z(s) = \left(\frac{\frac{1}{m}}{s^2 + \frac{b}{m}s + \frac{k}{m}} \right) F(s).$$

The controller designed in this problem will take the form

$$F = C(s)E(s),$$

where the error signal is $e = z_f^r - z$, where z_f^r is the prefiltered reference command. Find the controller $C(s)$ so that the closed loop system meets the following specifications.

- Reject constant input disturbances.
- Track reference signals with frequency content below $\omega_r = 0.1$ rad/s to within $\gamma_r = 0.03$.

- Attenuate noise on the measurement of z for all frequencies above $\omega_n = 500$ rad/s by $\gamma_n = 0.001$.
- The phase margin is close to $PM = 60$ degrees.
- Use a prefilter to reduce any peaking in the closed loop response.

E Design Study: Ball on Beam

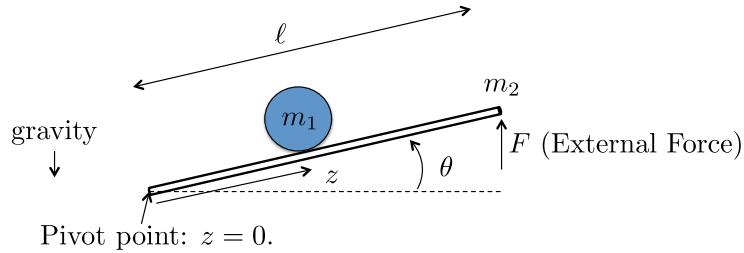


Figure 18.39: Ball on Beam Problem

Figure 18.39 shows the system. The position of the ball measured from the pivot point is z and the speed of the ball along the direction of the beam is \dot{z} . The angle of the beam from level is θ and the angular speed of the beam is $\dot{\theta}$. Gravity acts in the down direction. The mass of the ball is m_1 and the mass of the beam is m_2 . The length of the beam is ℓ . An external force is applied at the end of the beam as shown in Figure 18.39.

Use the following physical parameters: $m_1 = 0.35 \text{ kg}$, $m_2 = 2 \text{ kg}$, $\ell = 0.5 \text{ m}$, $g = 9.8 \text{ m/s}^2$.

Homework Problems:

- | | |
|---|---|
| E.2 Kinetic energy. | E.10 Digital PID. |
| E.a Simulink animation. | E.11 Full state feedback. |
| E.3 Equations of Motion. | E.12 Full state with integrator. |
| E.b Simulink S-function. | E.13 Observer based control. |
| E.4 Linearize equations of motion. | E.14 Disturbance observer. |
| E.5 Transfer function model. | E.15 Frequency Response. |
| E.6 State space model. | E.16 Loop gain. |
| E.8 Successive loop closure. | E.17 Stability margins. |
| E.9 Integrators and system type. | E.18 Loopshaping design. |
| E.d Root locus. | |



Homework E.2

Using the configuration variable z and θ , write an expression for the kinetic energy of the system. Assume that the ball is a point mass without rolling inertia. In general, this term will be small and will not have a large effect on the dynamics of the ball-beam system.



Homework E.a

Create a simulink animation of the ball on beam. The inputs should be sliders for z and θ . Turn in a screen capture of the animation.



Homework E.3

- (a) Find the potential energy for the system.
- (b) Define the generalized coordinates.
- (c) Find the generalized forces and damping forces.
- (d) Derive the equations of motion using the Euler-Lagrange equations.



Homework E.b

Modify the simulink, Matlab, or Python model created in homework E.b by creating a function that implements the equations of motion. The input to the function should be a slider for force. The output should go to the animation developed in homework E.b.



Homework E.4

For the ball on beam system:

- (a) Find the equilibria of the system.
- (b) Linearize the system about the equilibria using Jacobian linearization.
- (c) If possible, linearize the system using feedback linearization.



Homework E.5

For the inverted pendulum:

- (a) Start with the linearized equations and use the Laplace transform to convert the equations of motion to the s-domain.
- (b) Find the transfer functions from the input $\tilde{F}(s)$ to the outputs $\tilde{Z}(s)$ and $\tilde{\Theta}(s)$. Find the transfer function from the input $\tilde{\Theta}(s)$ to the output $\tilde{Z}(s)$.
- (c) Assume that deviations in the gravity torque away from the equilibrium torque due to motions of the ball along the beam are small compared to the torque required to hold up the beam. This will allow you to ignore the $m_1 g \tilde{z}$ term in the second equation of motion. How does this simplify your transfer functions?
- (d) Draw a block diagram of your simplified system transfer functions in a cascade from the input $\tilde{F}(s)$ to the fast intermediate output $\tilde{\Theta}(s)$ and then to slower output $\tilde{Z}(s)$.



Homework E.6

Defining the states as $\tilde{x} = (\tilde{z}, \tilde{\theta}, \dot{\tilde{z}}, \dot{\tilde{\theta}})^\top$, the input as $\tilde{u} = \tilde{F}$, and the measured output as $\tilde{y} = (\tilde{z}, \tilde{\theta})^\top$, find the linear state space equations in the form

$$\begin{aligned}\dot{\tilde{x}} &= A\tilde{x} + B\tilde{u} \\ \tilde{y} &= C\tilde{x} + D\tilde{u}.\end{aligned}$$



Homework E.8

For the ball and beam problem, do the following:

- (a) Using the principle of successive loop closure, draw a block diagram that uses PD control for both inner loop control and outer loop control. For design purposes, let the equilibrium position for the ball be $z_e = \frac{\ell}{2}$. The input to the outer loop controller is the desired ball position z_r and the output of the controller is the desired beam angle $\tilde{\theta}_r$. The input to the inner loop controller is the desired beam angle $\tilde{\theta}_r$ and the output is the force \tilde{F} on the end of the beam.

- (b) Focusing on the inner loop, find the PD gains k_{P_θ} and k_{D_θ} so that the rise time of the inner loop is $t_{r_\theta} = 1$ second, and the damping ratio is $\zeta_\theta = 0.707$.
- (c) Find the DC gain k_{DC_θ} of the inner loop.
- (d) Replacing the inner loop by its DC-gain, find the PD gains k_{P_z} and k_{D_z} so that the rise time of the outer loop is $t_{r_z} = 10t_{r_\theta}$ and the damping ratio is $\zeta_z = 0.707$.
- (e) Implement the successive loop closure design for the ball and beam in Simulink where the commanded ball position is given by a square wave with magnitude 0.25 ± 0.15 meters and frequency 0.01 Hz. Use the actual position in the ball as a feedback linearizing term for the equilibrium force. The block diagram should look something like Figure 18.40.
- (f) Suppose that the size of the input force on the beam is limited to $F_{\max} = 15$ N. Modify the Simulink diagram to include a saturation block on the force F . Using the rise time of the outer loop, tune the PD control law to get the fastest possible response without input saturation when a step of size 0.25 meter is placed on \tilde{z}^r .

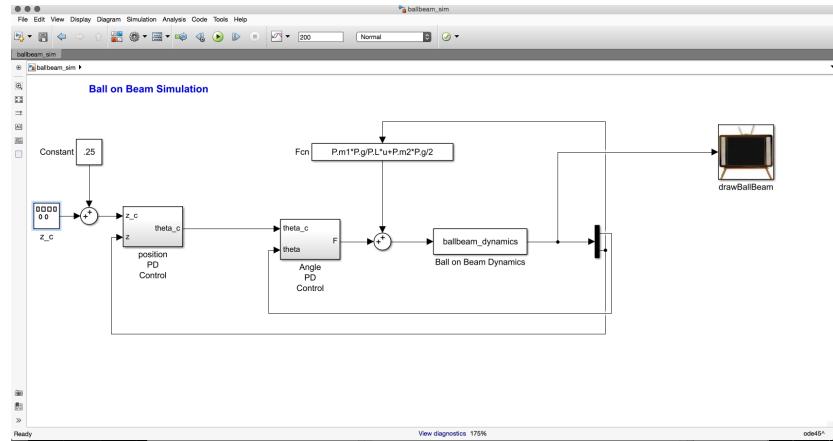


Figure 18.40: Block diagram for homework E.9.



Homework E.9

- (a) When the inner loop controller for the ballbeam system is PD control, what is the system type of the inner loop? Characterize the steady state error

when $\tilde{\theta}^d$ is a step, a ramp, and a parabola. What is the system type with respect to an input disturbance?

- (b) When the outer loop controller for the ballbeam is PD control, what is the system type of the outer loop? Characterize the steady state error when z^d is a step, a ramp, and a parabola. How does this change if you add an integrator? What is the system type with respect to an input disturbance for both PD and PID control?



Homework E.e

Adding an integrator to obtain PID control for the outer loop of the ballbeam system, put the characteristic equation in Evan's form and use the Matlab `rlocus` command to plot the root locus versus the integrator gain k_I . Note that since the integrator gain will be negative, the transfer function in Evan's form must be negated since the Matlab `rlocus` command only plots the return for $k_I > 0$. Select a value for k_I that does not significantly change the other locations of the closed loop poles.



Homework E.10

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Modify the system dynamics file so that the parameters m_1 , m_2 , and ℓ vary by up to 20% of their nominal value each time they are run (uncertainty parameter = 0.2).
- (b) Rearrange the block diagram so that the controller is implemented as an m-function implemented at the sample rate of $T_s = 0.01$. Implement the nested PID loops in Problems E.8 using an m-function called `ballbeam_ctrl.m`. If your simulation is in Matlab or Python then implement the PID controller as a separate class or function. Use the dirty derivative gain of $\tau = 0.05$. Tune the integrators to so that there is no steady state error. The controller should only assume knowledge of the position z , the angle θ , and the reference position z_r .
- (c) Note that the integrator gain will need to be negative which will cause problems for the anti-windup scheme that we have been implementing. Remove the old anti-windup scheme and implement a new scheme where the integrator only winds up when $|\dot{z}|$ is small.



Homework E.11

The objective of this problem is to implement a state feedback controller for the ball and beam problem. Start with the simulation files developed in Homework E.10.

- (a) Using the values for ω_{n_z} , ζ_z , ω_{n_θ} , and ζ_θ selected in Homework E.8, find the desired closed loop poles.
- (b) Add the state space matrices A , B , C , D derived in Homework E.6 to your param file.
- (c) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}_{A,B}) = n$.
- (d) Find the feedback gain K such that the eigenvalues of $(A - BK)$ are equal to desired closed loop poles. Find the reference gain k_r so that the DC-gain from z_r to z is equal to one.
- (e) Implement the state feedback scheme in Simulink and tune the closed loop poles to get good response. You should be able to get much faster response using state space methods.



Homework E.12

- (a) Modify the state feedback solution developed in Homework E.11 to add an integrator with anti-windup to the feedback loop for z .
- (b) Add a constant input disturbance of 1 Newtons to the input of the plant and allow the plant parameters to vary up to 20%.
- (c) Tune the integrator pole (and other gains if necessary) to get good tracking performance.



Homework E.13

The objective of this problem is to design an observer that estimates the state of the system and to use the estimated state in the controller designed in Homework E.12.

- (a) Modify the simulink diagram from HW E.12 as described in the description of HW A.13 to add a plot of the true state x , and estimated state \hat{x} , and the observation error $x - \hat{x}$.
- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, without an input disturbance. Modify `ballbeam_dynamics.m` so that the parameters known to the controller are the actual plant parameters (uncertainty parameter $\alpha = 0$).
- (d) In the control block, add an observer to estimate the state \hat{x} , and use the estimate of the state in your feedback controller. Tune the poles of the controller and observer to obtain good performance.
- (e) As motivation for the next chapter, add an input disturbance to the system of 0.5 and observe that there is steady state error in the observation error. In the next chapter we will show how to remove the steady state error in the observation error.



Homework E.14

- (a) Modify your solution from HW E.13 so that the uncertainty parameter in `ballbeam_dynamics.m` is $\alpha = 0.2$, representing 20% inaccuracy in the knowledge of the system parameters, and so that the input disturbance is 0.5. Also, add noise to the output channels z_m and θ_m with standard deviation of 0.001.
- (b) Add a disturbance observer to the controller, and verify that the steady state error in the estimator has been removed. Tune the system to get good response.



Homework E.15

- (a) Draw by hand the Bode plot of the inner loop transfer function from force \tilde{F} to angle $\tilde{\theta}$ for the ball beam system. Use the Matlab `bode` command and compare your results.
- (b) Draw by hand the Bode plot of the outer loop transfer function from angle $\tilde{\theta}$ to position $\tilde{z}(t)$ for the ball beam system. Use the Matlab `bode` command and compare your results.



Homework E.16

For the inner loop of the ball & beam system, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, and (2) the plant under PD control, using the control gains calculated in Homework E.10.

- (a) To what percent error can the closed loop system under PD control track the desired input if all of the frequency content of $\theta_r(t)$ is below $\omega_r = 1.0$ radians per second?
- (b) If the frequency content of the input disturbance is all contained below $\omega_{d_{in}} = 0.8$ radians per second, what percentage of the input disturbance shows up in the output?
- (c) If all of the frequency content of the noise $n(t)$ is greater than $\omega_{no} = 300$ radians per second, what percentage of the noise shows up in the output signal θ ?

For the outer loop of the ball & beam, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, and (2) the plant under PID control using the control gains calculated in Homework E.10.

- (d) If the frequency content of an output disturbance is contained below $\omega_{d_{out}} = 0.1$ radian/sec, what percentage of the output disturbance will be contained in the output under PID control?
- (e) If the reference signal is $y_r(t) = 2 \sin(0.6t)$ what is the output error under PID control?



Homework E.17

For this problem, use the gains found in HW E.10.

- (a) For the inner loop of the ball & beam system, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the inner loop system under PD control. On the same graph, plot the open loop Bode plot and the closed loop Bode plot. What is the bandwidth of the closed loop system, and how does this relate to the crossover frequency?
- (b) For the outer loop of the ball beam system, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the outer loop system under PID control. Plot the open and closed loop Bode plots for

the outer loop on the same plot as the open and closed loop for the inner loop. What is the bandwidth of the closed loop system, and how does this relate to the crossover frequency?

- (c) What is the bandwidth separation between the inner (fast) loop, and the outer (slow) loop. For this design, is successive loop closure justified?



Homework E.18

For this homework assignment we will use the loopshaping design technique to design a successive loop closure controller for the ball and beam problem.

- (a) First consider the inner loop, where $P_{in}(s)$ is the transfer function of the inner loop derived in HW E.5. Design the inner control system $C_{in}(s)$ to stabilize the system with a phase margin close to 60 degrees, and to ensure that reference inputs with frequency below $\omega_r = 1$ radians/sec have tracking error $\gamma_r = 0.0032$, and to ensure that noise above $\omega_n = 1000$ radians/second is rejected by $\gamma_n = 0.0032$.
- (b) Now consider the design of the controller for the outer loop system. The 'plant' for the design of the outer loop controller is

$$P = P_{out} \frac{P_{in}C_{in}}{1 + P_{in}C_{in}}.$$

Design the outer loop controller C_{out} so that the system is stable with phase margin close to $PM = 60$ degrees, and so that constant input disturbances are rejected, reference signals with frequency below $\omega_r = 0.1$ radians/sec are tracked with error $\gamma_r = 0.01$, and noise with frequency content above $\omega_n = 100$ radians/sec are rejected with $\gamma_n = 0.001$. Add a prefilter to reduce peaking in the closed loop transfer function.

F Design Study: Planar VTOL

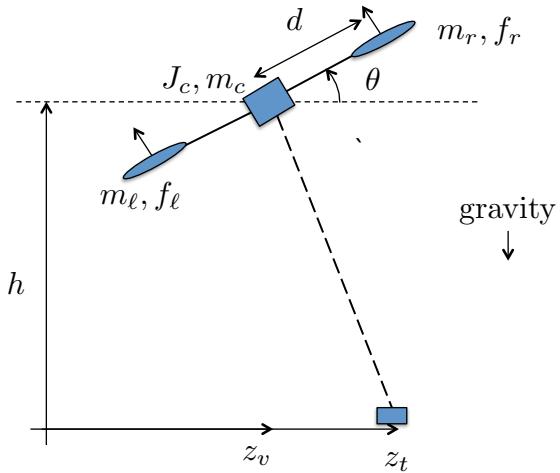


Figure 18.41: Planar vertical take-off and landing (VTOL)

In this design study we will explore the control design for a simplified planar version of a quadrotor following a ground target. In particular, we will constrain the dynamics to be in two dimension plane comprising vertical and one dimension of horizontal, as shown in Figure 18.41. The planar vertical take-off and landing (VTOL) system is comprised of a center pod of mass m_c and inertia J_c , a right motor/rotor that is modeled as a point mass m_r that exerts a force f_r at a distance d from the center of mass, and a left motor/rotor that is modeled as a point mass m_ℓ that exerts a force f_ℓ at a distance $-d$ from the center of mass. The position of the center of mass of the planar VTOL system is given by horizontal position z_v and altitude h . The airflow through the rotor creates a change in the direction of flow of air and causes what is called “momentum drag.” Momentum drag can be modeled as a viscous drag force that is proportional to the horizontal velocity \dot{z}_v . In other words, the drag force is $F_{\text{drag}} = -\mu \dot{z}_v$. The target on the ground will be modeled as an object with position z_t and altitude $h = 0$. We will not explicitly model the dynamics of the target.

Use the following physical parameters: $m_c = 1 \text{ kg}$, $J_c = 0.0042 \text{ kg m}^2$, $m_r = 0.25 \text{ kg}$, $m_\ell = 0.25 \text{ kg}$, $d = 0.3 \text{ m}$, $\mu = 0.1 \text{ kg/s}$, $g = 9.81 \text{ m/s}^2$.

Homework Problems:

- F.2** Kinetic energy.
- F.a** Simulink animation.
- F.3** Equations of Motion.
- F.b** Simulink S-function.

- F.4** Linearize equations of motion.
- F.5** Transfer function model.
- F.6** State space model.
- F.7** Pole Placement using PD.

- F.8** Successive loop closure.
F.9 Integrators and system type.
F.d Root locus.
F.10 Digital PID.
F.11 Full state feedback.
F.12 Full state with integrator.
- F.13** Observer based control.
F.14 Disturbance observer.
F.15 Frequency Response.
F.16 Loop gain.
F.17 Stability margins.
F.18 Loopshaping design.



Homework F.2

Using the configuration variables z_v , h , and θ , write an expression for the kinetic energy of the system.



Homework F.a

Create a simulink animation of the planar VTOL system. The inputs should be sliders for z_v , z_t , h , and θ . Turn in a screen capture of the animation.



Homework F.3

- (a) Find the potential energy for the system.
- (b) Define the generalized coordinates and damping forces.
- (c) Find the generalized forces. Note that the right and left forces are more easily modeled as a total force on the center of mass, and a torque about the center of mass.
- (d) Derive the equations of motion for the planar VTOL system using the Euler-Lagrange equations.



Homework F.b

Modify the simulink, Matlab, or Python model created in homework F.b by creating a function that implements the equations of motion. The input to the function should be a slider for the forces f_r and f_ℓ . The output should go to the animation developed in homework F.b. The target is still maneuvered using a slider bar.



Homework F.4

Since f_r and f_ℓ appear in the equations as either $f_r + f_\ell$ or $d(f_r - f_\ell)$, it is easier to think of the inputs as the total force $F \triangleq f_r + f_\ell$, and the torque

$\tau = d(f_r - f_\ell)$. Note that since

$$\begin{pmatrix} F \\ \tau \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ d & -d \end{pmatrix} \begin{pmatrix} f_r \\ f_\ell \end{pmatrix},$$

that

$$\begin{pmatrix} f_r \\ f_\ell \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ d & -d \end{pmatrix}^{-1} \begin{pmatrix} F \\ \tau \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2d} \\ \frac{1}{2} & -\frac{1}{2d} \end{pmatrix}^{-1} \begin{pmatrix} F \\ \tau \end{pmatrix}.$$

Therefore, in subsequent exercises, if we determine F and τ , then the right and left forces are given by

$$\begin{aligned} f_r &= \frac{1}{2}F + \frac{1}{2d}\tau \\ f_\ell &= \frac{1}{2}F - \frac{1}{2d}\tau. \end{aligned}$$

- (a) Find the equilibria of the system.
- (b) Linearize the equations about the equilibria using Jacobian linearization.
- (c) If possible, linearize the system using feedback linearization.



Homework F.5

For the planar VTOL system, note that the system naturally divides into so-called longitudinal dynamics (up-down motion) with total force \tilde{F} as the input and altitude \tilde{h} as the output, and lateral dynamics (side-to-side motion) with $\tilde{\tau}$ as the input and \tilde{z} as the output, with $\tilde{\theta}$ as an intermediate variable.

- (a) Start with the linearized equations of motion and use the Laplace transform to convert the equations of motion to the s-domain.
- (b) For the longitudinal dynamics, find the transfer function from the input $\tilde{F}(s)$ to the output $\tilde{H}(s)$.
- (c) For the lateral dynamics, find the transfer function from the input $\tilde{\tau}(s)$ to the outputs $\tilde{Z}(s)$ and $\tilde{\Theta}(s)$. Identify the fast and slow subsystem.
- (d) Draw a block diagram of the open loop longitudinal and lateral systems.



Homework F.6

- (a) Defining the longitudinal states as $\tilde{x}_{lon} = (\tilde{h}, \dot{\tilde{h}})^\top$ and the longitudinal input as $\tilde{u}_{lon} = \tilde{F}$ and the longitudinal output as $\tilde{y}_{lon} = \tilde{h}$, find the linear state space equations in the form

$$\begin{aligned}\dot{\tilde{x}}_{lon} &= A\tilde{x}_{lon} + B\tilde{u}_{lon} \\ \tilde{y}_{lon} &= C\tilde{x}_{lon} + D\tilde{u}_{lon}.\end{aligned}$$

- (b) Defining the lateral states as $\tilde{x}_{lat} = (\tilde{z}, \tilde{\theta}, \dot{\tilde{z}}, \dot{\tilde{\theta}})^\top$ and the lateral input as $\tilde{u}_{lat} = \tilde{\tau}$ and the lateral output as $\tilde{y}_{lat} = (\tilde{z}, \tilde{\theta})^\top$, find the linear state space equations in the form

$$\begin{aligned}\dot{\tilde{x}}_{lat} &= A\tilde{x}_{lat} + B\tilde{u}_{lat} \\ \tilde{y}_{lat} &= C\tilde{x}_{lat} + D\tilde{u}_{lat}.\end{aligned}$$



Homework F.7

For this problem we will only consider the longitudinal (up-down) dynamics of the VTOL system.

- (a) Given the open-loop transfer function from total force \tilde{F} to altitude \tilde{h} found in problem F.5, find the open-loop poles of the system.
- (b) Using PD control architecture shown in Figure 7.2, find the closed-loop transfer function from \tilde{h}_r to \tilde{h} and find the closed-loop poles as a function of k_P and k_D .
- (c) Select k_P and k_D to place the closed-loop poles at $p_1 = -0.2$ and $p_2 = -0.3$.
- (d) Using the gains from part (c), implement the PD control for the altitude control in Simulink and plot the step response.



Homework F.8

For the VTOL system, do the following:

- (a) For the longitudinal (altitude) dynamics of the VTOL, suppose that the design requirements are that the rise time is $t_r \approx 8$ seconds, with a damping ratio of $\zeta = 0.707$. Find the desired closed-loop characteristic polynomial $\Delta_{cl}^d(s)$, and the associated pole locations. Find the proportional and derivative gains k_{P_h} and k_{D_h} to achieve these specifications, and modify the Simulink simulation from HW F.7 to verify that the step response satisfies the requirements.

- (b) For lateral dynamics of the VTOL, using the principle of successive loop closure, draw a block diagram that uses PD control for both inner-loop control and outer-loop control. The input to the outer-loop controller is the desired lateral position z_r and the output of the controller is the desired pitch angle θ_r . The input to the inner-loop controller is the desired pitch angle θ_r and the output is the torque τ on the VTOL center.
- (c) Focusing on the inner loop, find the PD gains k_{P_θ} and k_{D_θ} so that the rise time of the inner loop is $t_{r_\theta} = 0.8$ seconds, and the damping ratio is $\zeta_\theta = 0.707$.
- (d) Find the DC gain k_{DC_θ} of the inner loop.
- (e) Replacing the inner loop by its DC-gain, find the PD gains k_{P_z} and k_{D_z} so that the rise time of the outer loop is $t_{r_z} = 10t_{r_\theta}$ and the damping ratio is $\zeta_z = 0.707$.
- (f) Implement the successive loop closure design for the lateral control of the VTOL system in Simulink where the commanded lateral position is given by a square wave with magnitude 3 ± 2.5 meters and frequency 0.08 Hz. The block diagram for the system should look something like Figures 18.42 and 18.43.
- (g) Suppose that the size of the force on each rotor is constrained by $0 \leq f_\ell \leq f_{\max} = 10$ N, and $0 \leq f_r \leq f_{\max} = 10$ N. Modify the Simulink diagram to include saturation blocks on the right and left forces f_ℓ and f_r . Using the rise time for the altitude controller, and the rise time for the outer loop of the lateral controller as tuning parameters, tune the controller to achieve the fastest possible response without input saturation.

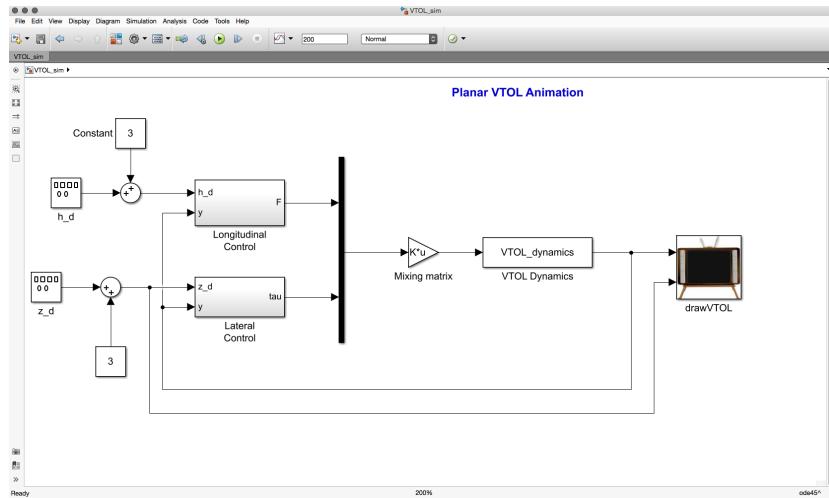


Figure 18.42: Block diagram for homework F.8.

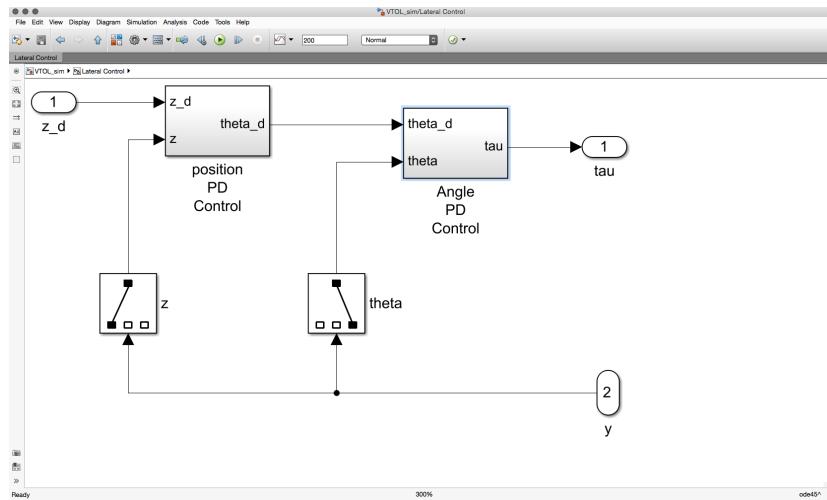


Figure 18.43: Block diagram for lateral controller in homework F.8.



Homework F.9

- (a) When the the longitudinal controller for the VTOL system is PD control, what is the system type? Characterize the steady state error when h^d is a step, a ramp, and a parabola. How does this change if you add an integrator? What is the system type with respect to an input disturbance for both PD and PID control?
- (b) When the inner loop of the lateral controller for the VTOL system is PD control, what is the system type of the inner loop? Characterize the steady state error when $\tilde{\theta}^d$ is a step, a ramp, and a parabola. What is the system type with respect to an input disturbance?
- (c) When the outer loop of the lateral controller for the VTOL system is PD control, what is the system type of the outer loop? Characterize the steady state error when z^d is a step, a ramp, and a parabola. How does this change if you add an integrator? What is the system type with respect to an input disturbance for both PD and PID control?



Homework F.e

- (a) Adding an integrator to obtain PID control for the longitudinal controller for the VTOL system, put the characteristic equation in Evan's form and use the Matlab `rlocus` command to plot the root locus versus the integrator gain k_{I_h} . Select a value for k_{I_h} that does not significantly change the other locations of the closed loop poles.
- (b) Adding an integrator to obtain PID control for the outer loop of the lateral controller for the VTOL system, put the characteristic equation of the outer loop in Evan's form and use the Matlab `rlocus` command to plot the root locus versus the integrator gain k_{I_z} . Select a value for k_{I_z} that does not significantly change the other locations of the closed loop poles.



Homework F.10

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Modify the system dynamics file so that the parameters m_c , J_c , d , and μ vary by up to 20% of their nominal value each time they are run (uncertainty parameter = 0.2).
- (b) Rearrange the block diagram so that both longitudinal and lateral controllers are implemented as an m-function implemented at the sample rate of $T_s = 0.01$. Implement the nested PID loops in Problems F.8 using an m-function called `VTOL_ctrl.m`. If your simulation is in Matlab or Python then implement the PID controller as a separate class or function. Use the dirty derivative gain of $\tau = 0.05$. Tune the integrator to so that there is no steady state error. The controller should only assume knowledge of the position z , the angle θ , the altitude h , the reference position z_r , and the reference altitude h_r .



Homework F.11

The objective of this problem is to implement a state feedback controller for the VTOL system. Start with the simulation files developed in Homework F.10.

- (a) Using the values for ω_{n_h} , ζ_h , ω_{n_z} , ζ_z , ω_{n_θ} , and ζ_θ selected in Homework F.8, find the desired closed loop poles.
- (b) Add the state space matrices A , B , C , D derived in Homework F.6 to your param file.
- (c) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}_{A,B}) = n$.

- (d) Find the feedback gain K such that the eigenvalues of $(A - BK)$ are equal to desired closed loop poles. Find the reference gains k_{r_h} and k_{r_z} so that the DC-gain from h_r to h is equal to one, and the DC-gain from z_r to z is equal to one.
- (e) Implement the state feedback scheme in Simulink and tune the closed loop poles to get good response. You should be able to get a faster response using state space methods.



Homework F.12

- (a) Modify the state feedback solution developed in Homework F.11 to add an integrator with anti-windup to the altitude feedback loop and to the position feedback loop.
- (b) Allow the plant parameters to vary up to 20% and add a constant input disturbance of 0.1 Newtons to the input of position dynamics simulating wind. *Hint: The best place to add the wind force is in VTOL_dynamics.m. For example, one possibility is to modify the z dynamics as*

$$zddot = (- (fr+f1) *sin(theta) +F_wind) / (P.mc+2*P.mr);$$
- (c) Tune the integrator poles on both loops (and other gains if necessary) to get good tracking performance.



Homework F.13

The objective of this problem is to design an observer that estimates the state of the system and to use the estimated state in the controller designed in Homework F.12.

- (a) Modify the simulink diagram from HW F.12 as described in the description of HW A.13 to add a plot of the true state x , and estimated state \hat{x} , and the observation error $x - \hat{x}$.
- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, without an input disturbance. Modify `VTOL_dynamics.m` so that the parameters known to the controller are the actual plant parameters (uncertainty parameter $\alpha = 0$).

- (d) In the control block, add an observer for both the longitudinal and lateral dynamics to estimate the state \hat{x} , and use the estimate of the state in your feedback controller. Tune the poles of both controllers and observers to obtain good performance.



Homework F.14

- (a) Modify your solution from HW F.13 so that the uncertainty parameter in `VTOL_dynamics.m` is $\alpha = 0.2$, representing 20% inaccuracy in the knowledge of the system parameters. Modify `VTOL_dynamics` to add an altitude disturbance of 1.0 and a wind disturbance of 1.0 m/s. Matlab code that implements these disturbances is given by

```

1  % disturbances
2  wind = 1.0;
3  altitude_dist = 1.0;
4
5  % add wind disturbance
6  zdot = zdot + wind;
7
8  zddot = -(fr+f1)*sin(theta)/(mc+2*mr);
9  hddot = (-(mc+2*mr)*g + (fr+f1)*cos(theta))/(mc+2*mr) ...
10   + altitude_dist;
11  thetaddot = d*(fr-f1)/(Jc+2*mr*d^2);

```

Before adding the disturbance observer, run the simulation and note that the controller is not robust to the input disturbance.

- (b) Add a disturbance observer to both controllers, and verify that the steady state error in the estimator has been removed and that the disturbances have been adequately compensated. Tune the system to get good response.



Homework F.15

- (a) Draw by hand the Bode plot of the altitude transfer function from force \tilde{F} to altitude \tilde{h} . Use the Matlab `bode` command and compare your results.
- (b) Draw by hand the Bode plot of the inner loop transfer function for the lateral dynamics from torque τ to angle θ . Use the Matlab `bode` command and compare your results.
- (c) Draw by hand the Bode plot of the outer loop transfer function for the lateral dynamics from angle θ to position $z(t)$. Use the Matlab `bode` command and compare your results.



Homework F.16

For the altitude hold loop of the VTOL system, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, and (2) the plant under PID control using the control gains calculated in Homework F.10.

- (a) What is the tracking error if the reference input is a parabola with curvature 5?
- (b) If all of the frequency content of the noise $n(t)$ is greater than $\omega_{no} = 30$ radians per second, what percentage of the noise shows up in the output signal θ ?

For the inner loop of the lateral controller for the VTOL system, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, and (2) the plant under PD control, using the control gains calculated in Homework F.10.

- (c) If the frequency content of the input disturbance is all contained below $\omega_{din} = 2$ radians per second, what percentage of the input disturbance shows up in the output?
- (d) If a sensor for θ is to be selected, what are the characteristics of the sensor (frequency band, and size) that would result in measurement noise for θ that is less than 0.1 degrees?

For the outer loop of the lateral controller for the VTOL system, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, and (2) the plant under PD control, using the control gains calculated in Homework F.10.

- (e) To what percent error can the closed loop system track the desired input if all of the frequency content of $z_r(t)$ is below $\omega_r = 0.1$ radians per second?
- (f) If the frequency content of an output disturbance is contained below $\omega_{d_{out}} = 0.01$ radian/sec, what percentage of the output disturbance will be contained in the output?



Homework F.17

For this problem, use the gains found in HW F.10.

- (a) For the altitude hold loop of the VTOL system, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the closed loop system under PID control. On the same graph, plot the open loop Bode plot and the closed loop Bode plot. What is the bandwidth of the closed loop system, and how does this relate to the crossover frequency?
- (b) For the inner loop of the lateral control loop, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the inner loop system under PD control. On the same graph, plot the open loop Bode plot and the closed loop Bode plot. What is the bandwidth of the closed loop system, and how does this relate to the crossover frequency?
- (c) For the outer loop of the lateral control loop, use the Matlab `bode` and `margin` commands to find the phase and gain margin for the outer loop system under PD control. Plot the open and closed loop Bode plots for the outer loop on the same plot as the open and closed loop for the inner loop. What is the bandwidth of the closed loop system, and how does this relate to the crossover frequency?
- (d) What is the bandwidth separation between the inner (fast) loop, and the outer (slow) loop. For this design, is successive loop closure justified?



Homework F.18

For this homework assignment we will use the loopshaping design technique to design a controller for the vtol flight control problem.

- (a) First consider the longitudinal or altitude control problem. For this part, the force command will be

$$F = F_e + C_{lon}(s)E_h(s),$$

where F_e is the equilibrium force as found in other homework problems, and the error signal is $e_h = h_f^r - h_m$, where h_f^r is the prefiltered reference altitude command, and h_m is the measured altitude. Find the longitudinal controller $C_{lon}(s)$ so that the closed loop system meets the following specifications.

- Reject constant input disturbances.
- Track reference signals with frequency content below $\omega_r = 0.1$ rad/s to within $\gamma_r = 0.01$.
- Attenuate noise on the measurement of h for all frequencies above $\omega_n = 200$ rad/s by $\gamma_n = 10^{-4}$.
- The phase margin is close to $PM = 60$ deg.
- Use a prefilter to improve the closed loop response.

- (b) Now consider the inner loop of the lateral, or position, dynamics. Find the inner loop controller $C_{lat_{in}}(s)$ so that the inner loop is stable with phase margin approximately $PM = 60$ deg, and with closed loop bandwidth approximately $\omega_{co} = 10$ rad/s. Add a low pass filter to enhance noise rejection.
- (c) For the outer loop of the lateral system, find the controller $C_{lat_{out}}(s)$ to meet the following specifications:
- Cross over frequency is around $\omega_{co} = 1$ rad/s.
 - Reject constant input disturbances.
 - Attenuate noise on the measurement of z for all frequencies above $\omega_{no} = 100$ rad/s by $\gamma_{no} = 10^{-5}$.
 - The phase margin is close to $PM = 60$ deg.
 - Use a prefilter to reduce overshoot in the step response.

Appendix a

Creating Animations in Simulink

In the study of dynamics and control, it is essential to be able to visualize the motion of the physical system. In this appendix we describe how to create animations in Matlab/Simulink.

Appendix a.1 Handle Graphics in Matlab

When a graphics function like `plot` is called in Matlab, the function returns a *handle* to the plot. A graphics handle is similar to a pointer in C/C++ in the sense that all of the properties of the plot can be accessed through the handle. For example, the Matlab command

```
1      >> plot_handle=plot(t,sin(t))
```

returns a pointer, or handle, to the plot of `sin(t)`. Properties of the plot can be changed by using the handle, rather than reissuing the `plot` command. For example, the Matlab command

```
1      >> set(plot_handle, 'YData', cos(t))
```

changes the plot to `cos(t)` without redrawing the axes, title, label, or other objects that may be associated with the plot. If the plot contains drawings of several objects, a handle can be associated with each object. For example,

```
1      >> plot_handle1 = plot(t,sin(t))
2      >> hold on
3      >> plot_handle2 = plot(t,cos(t))
```

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

draws both $\sin(t)$ and $\cos(t)$ on the same plot, with a handle associated with each object. The objects can be manipulated separately without redrawing the other object. For example, to change $\cos(t)$ to $\cos(2t)$, issue the command

```
1      >> set(plot_handle2, 'YData', cos(2*t))
```

We can exploit this property to animate simulations in Simulink by redrawing only the parts of the animation that change in time, and thereby significantly reducing the simulation time. To show how handle graphics can be used to produce animations in Simulink, we will provide five detailed examples, including the three design examples which are 2-D animation, and also a 3-D animation of a spacecraft using lines to produce a stick figure, and a modified version that uses the vertices-faces data construction in Matlab.

Appendix a.2 Design Study A: Single Link Robot Arm



Homework Problem A.a

Create a simulink animation of the single link robot arm. The input should be a slider for θ .

Solution

Consider the image of the single link robot arm shown in Figure a.1, where the configuration is completely specified by the angle θ of the arm from vertical. The physical parameters of the system that impact the animation are the arm length ℓ , and the arm width w . The first step in developing the animation is

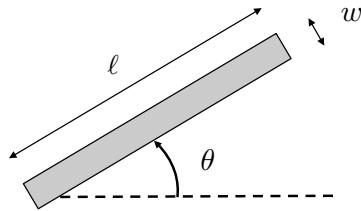


Figure a.1: Drawing for single link robot arm. The first step in developing an animation is to draw a figure of the object to be animated and identify all of the physical parameters.

to determine the position of points that define the animation. For example, for the single link robot arm in Figure a.1, the four corners of the arm, when $\theta = 0$

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

are given by

$$p_0 = \begin{pmatrix} 0 & \ell & \ell & 0 \\ -\frac{w}{2} & -\frac{w}{2} & \frac{w}{2} & \frac{w}{2} \end{pmatrix},$$

where the first column of p_0 is the lower left corner, the second column is the lower right corner, the third column is the upper right corner, and the last column is the upper left corner. When $\theta \neq 0$, the points are rotated by θ to obtain

$$p(\theta) = R(\theta)p_0,$$

where

$$R = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

is a rotation matrix that rotates points in \mathbb{R}^2 by θ .

The Matlab code that draws the arm link and returns a handle to the image of the link is given by

```

1 function handle = drawLink(theta, L, w, handle)
2     pts = [ 0, L, L, 0; -w/2, -w/2, w/2, w/2];
3     R = [cos(theta), -sin(theta); sin(theta), cos(theta)];
4     pts = R*pts;
5     X = pts(1,:);
6     Y = pts(2,:);
7
8     if isempty(handle),
9         handle = fill(X,Y,'b');
10    else
11        set(handle, 'XData',X, 'YData',Y);
12    end
13 end

```

Line 2 defines the points when $\theta = 0$, and line 3 defines the rotation matrix R . The rotated points are given by line 4, and the associated X and Y coordinates are extracted in lines 5 and 6. Note that in Line 1, `handle` is both an input and an output. If an empty array is passed into the function, then the `fill` command is used to plot the arm in Line 9. On the other hand, if a valid handle is passed into the function, then the arm is redrawn using the `set` command in Line 11.

The main routine for the robot arm animation is listed below.

```

1 function drawArm(u)
2     % process inputs to function
3     theta      = u(1);
4     t          = u(2);
5     % drawing parameters
6     L = 1;
7     w = 0.3;
8     % define persistent variables
9     persistent link_handle
10    % at time 0, initialize plot and persistent vars
11    if t==0,

```

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

```

12     figure(1), clf
13     plot([0,L],[0,0], 'k—'); % plot track
14     hold on
15     link_handle = drawLink(theta, L, w, []);
16     axis([-2*L, 2*L, -2*L, 2*L]);
17     % at every other time step, redraw link
18     else
19         drawLink(theta, L, w, link_handle);
20     end
21 end

```

The routine `drawArm` is called from the Simulink file shown in Figure a.2, where there are two inputs: the angle θ , and the time t . Lines 3-4 rename the inputs to θ , and t . Lines 6-7 define the drawing parameters ℓ and w . We require that the graphics handle persist between function calls to `drawArm`. We define a persistent variable `handle` in Line 9. The `if` statement in Lines 11-20 is used to produce the animation. Lines 12-16 are called once at the beginning of the simulation, and draw the initial animation. Line 12 brings the figure 1 window to the front and clears it. Lines 13 and 14 draw the zero angle line as a visual reference. Line 15 calls the `drawLink` routine with an empty handle as input, and returns the handle to the link. Line 16 sets the axes of the figure. After the initial time step, all that needs to be changed are the locations of link. Therefore, in Line 19, the `drawLink` routines are called with the figure handles as inputs.

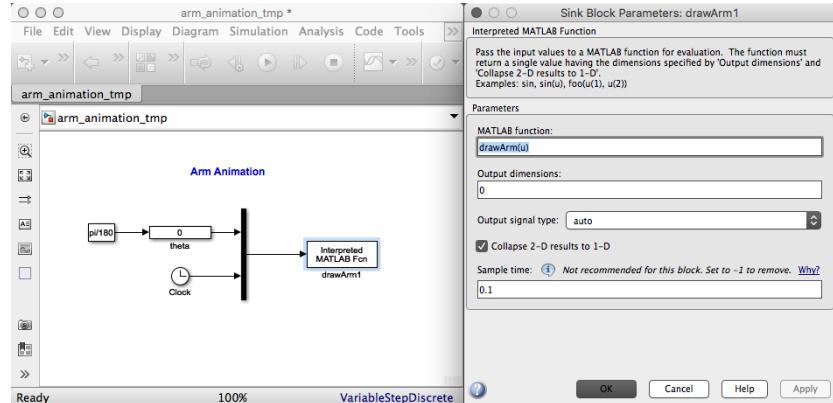


Figure a.2: Simulink file for debugging the arm animation. There are two inputs to the Matlab m-file `drawArm`: the angle θ , and the time t . A slider gain for θ is used to verify the animation.

See <http://controlbook.byu.edu> for the complete solution.

Appendix a.3 Design Study B. Pendulum on Cart



Homework Problem B.a

Create a simulink animation of the pendulum on a cart system. The inputs should be sliders for z and θ .

Solution

Consider the image of the inverted pendulum shown in Figure a.3, where the configuration is completely specified by the position of the cart z , and the angle of the rod from vertical θ . The physical parameters of the system are the rod length L , the base width w , the base height h , and the gap between the base and the track g . The first step in developing the animation is to determine

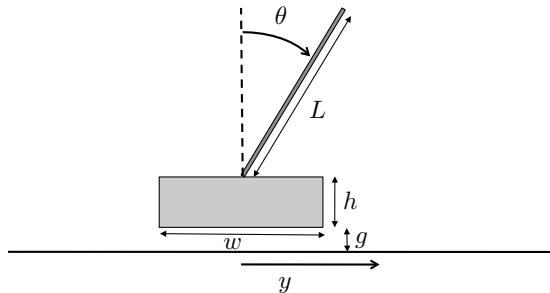


Figure a.3: Drawing for inverted pendulum. The first step in developing an animation is to draw a figure of the object to be animated and identify all of the physical parameters.

the position of points that define the animation. For example, for the inverted pendulum in Figure a.3, the four corners of the base are

$$(z + w/2, g), (z + w/2, g + h), (z - w/2, g + h), \text{ and } (z - w/2, g),$$

and the two ends of the rod are given by

$$(z, g + h) \text{ and } (z + L \sin \theta, g + h + L \cos \theta).$$

Since the base and the rod can move independently, each will need its own figure handle. The `drawBase` command can be implemented with the following Matlab code:

```

1 function handle = drawBase(z, width, height, gap, handle)
2     X = [z-width/2, z+width/2, z+width/2, z-width/2];
3     Y = [gap, gap, gap+height, gap+height];
4     if isempty(handle),

```

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

```

5      handle = fill(X,Y,'m');
6  else
7      set(handle,'XData',X,'YData',Y);
8 end

```

Lines 2 and 3 define the X and Y locations of the corners of the base. Note that in Line 1, `handle` is both an input and an output. If an empty array is passed into the function, then the `fill` command is used to plot the base in Line 5. On the other hand, if a valid handle is passed into the function, then the base is redrawn using the `set` command in Line 7.

The Matlab code for drawing the rod is similar and is listed below.

```

1  function handle = drawRod(z, theta, L, gap, height, handle)
2      X = [z, z+L*sin(theta)];
3      Y = [gap+height, gap + height + L*cos(theta)];
4      if isempty(handle),
5          handle = plot(X, Y, 'g');
6      else
7          set(handle,'XData',X,'YData',Y);
8      end

```

The main routine for the pendulum animation is listed below.

```

1  function drawPendulum(u)
2      % process inputs to function
3      z        = u(1);
4      theta    = u(2);
5      t        = u(3);
6
7      % drawing parameters
8      L = 1;
9      gap = 0.01;
10     width = 1.0;
11     height = 0.1;
12
13     % define persistent variables
14     persistent base_handle
15     persistent rod_handle
16
17     % first time function is called, initialize plot
18     % and persistent vars
19     if t==0,
20         figure(1), clf
21         track_width=3;
22         % plot track
23         plot([-track_width,track_width],[0,0],'k');
24         hold on
25         base_handle = drawBase(z, width, height, gap, []);
26         rod_handle = drawRod(z, theta, L, gap, height, []);
27         axis([-track_width,track_width,-L,2*track_width-L]);
28         % at every other time step, redraw base and rod
29     else
30         drawBase(z, width, height, gap, base_handle);
31         drawRod(z, theta, L, gap, height, rod_handle);

```

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

32

[end](#)

The routine `drawPendulum` is called from the Simulink file shown in Figure a.4, where there are three inputs: the position z , the angle θ , and the time t . Lines 3-5 rename the inputs to z , θ , and t . Lines 8-11 define the drawing parameters. We require that the handle graphics persist between function calls to `drawPendulum`. Since a handle is needed for both the base and the rod, we define two persistent variables in Lines 14 and 15. The `if` statement in Lines 19-32 is used to produce the animation. Lines 20-27 are called once at the beginning of the simulation, and draw the initial animation. Line 20 brings the figure 1 window to the front and clears it. Lines 21 and 23 draw the ground along which the pendulum will move. Line 25 calls the `drawBase` routine with an empty handle as input, and returns the handle `base_handle` to the base. Line 26 calls the `drawRod` routine, and Line 25 sets the axes of the figure. After the initial time step, all that needs to be changed are the locations of the base and rod. Therefore, in Lines 30 and 31, the `drawBase` and `drawRod` routines are called with the figure handles as inputs.

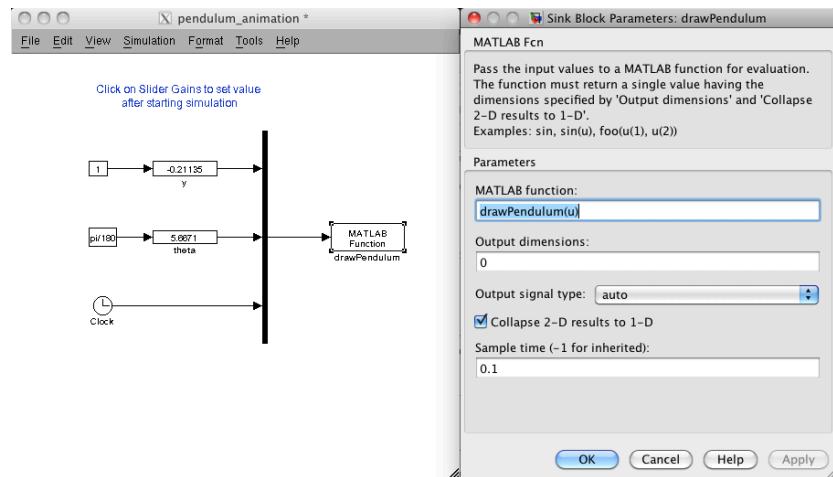


Figure a.4: Simulink file for debugging the pendulum simulation. There are three inputs to the Matlab m-file `drawPendulum`: the position z , the angle θ , and the time t . Slider gains for z and θ are used to verify the animation.

Appendix a.4 Design Study C. Satellite Attitude Control



Homework Problem C.a

Create a simulink animation of the satellite system. The inputs should be sliders for θ and ϕ .

Solution

The m-file that implements the animation for the simple satellite system is listed below.

```

1 function satellite_animation(u, P)
2
3 % process inputs to function
4 theta = u(1);
5 phi = u(2);
6 %thetadot = u(3);
7 %phidot = u(4);
8 t = u(5);
9
10 % define persistent variables
11 persistent base_handle
12 persistent panel_handle
13
14 % first time function is called, initialize plot and
15 % persistent vars
16 if t==0
17 figure(1), clf
18 plot([-2*P.length,2*P.length],[0,0], 'k—'); % plot track
19 hold on
20 base_handle = drawBase(theta, P.width, []);
21 panel_handle = drawPanel(phi, P.width, P.length, []);
22 axis([-2*P.length, 2*P.length, -2*P.length, 2*P.length]);
23
24
25 % at every other time step, redraw base and rod
26 else
27 drawBase(theta, P.width, base_handle);
28 drawPanel(phi, P.width, P.length, panel_handle);
29 end
30 end
31
32
33 %
34 %=====
35 % drawBase
36 % draw the base of the pendulum
37 % return handle if 3rd argument is empty, otherwise use 3rd
38 % arg as handle
39 %=====

```

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

```

40 %
41 function handle = drawBase(theta, w, handle)
42
43 % define points on base (without rotation)
44 pts = [...
45     w/2, -w/2; ...
46     w/2, -w/6; ...
47     w/2+w/6, -w/6; ...
48     w/2+w/6, w/6; ...
49     w/2, w/6; ...
50     w/2, w/2; ...
51     -w/2, w/2; ...
52     -w/2, w/6; ...
53     -w/2-w/6, w/6; ...
54     -w/2-w/6, -w/6; ...
55     -w/2, -w/6; ...
56     -w/2, -w/2; ...
57 ];
58 % define rotation matrix
59 R = [cos(theta), sin(theta); -sin(theta), cos(theta)];
60 % rotate points
61 pts = R*pts;
62 % break into X and Y components
63 X = pts(1,:);
64 Y = pts(2,:);
65
66 if isempty(handle)
67     handle = fill(X,Y,'b');
68 else
69     set(handle,'XData',X,'YData',Y);
70     drawnow
71 end
72 end
73
74 %
75 %=====
76 % drawPanel
77 % draw the solar panel
78 % return handle if 3rd argument is empty, otherwise use 3rd
79 % arg as handle
80 %=====
81 %
82 function handle = drawPanel(phi, w, L, handle)
83
84 % define points on base (without rotation)
85 pts = [...
86     -L, -w/6; ...
87     L, -w/6; ...
88     L, w/6; ...
89     -L, w/6; ...
90 ];
91 % define rotation matrix
92 R = [cos(phi), sin(phi); -sin(phi), cos(phi)];
93 % rotate points
94 pts = R*pts;
95 % break into X and Y components
96 X = pts(1,:);

```

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

```
97     Y = pts(2,:);
98
99     if isempty(handle)
100         handle = fill(X, Y, 'g');
101     else
102         set(handle, 'XData', X, 'YData', Y);
103         drawnow
104     end
105 end
```

See <http://controlbook.byu.edu> for the complete solution.

Appendix a.5 Animation Example: Spacecraft using Lines

The previous sections showed examples of 2-D animation. In this section we discuss a 3-D animation of a spacecraft with six degrees of freedom. Figure a.5 shows a simple line drawing of a spacecraft, where the bottom is meant to denote a solar panel that should be oriented toward the sun.

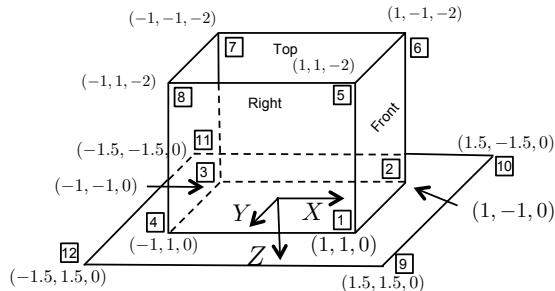


Figure a.5: Drawing used to create spacecraft animation. Standard aeronautics body axes are used, where the x -axis points out the front of the spacecraft, the y -axis points to the right, and the z -axis point out the bottom of the body.

The first step in the animation process is to label the points on the spacecraft and to determine their coordinates in a body-fixed coordinate system. We will use standard aeronautics axes with X pointing out the front of the spacecraft, Y pointing to the right, and Z pointing out the bottom. The points 1 through 12 are labeled in Figure a.5 and specify how coordinates are assigned to each label. To create a line drawing, we need to connect the points in a way that draws each of the desired line segments. To do this as one continuous line, some of the segments will need to be repeated. To draw the spacecraft shown in Figure a.5, we will transition through the following nodes: 1 – 2 – 3 – 4 – 1 – 5 – 6 – 2 – 6 – 7 – 3 – 7 – 8 – 4 – 8 – 5 – 1 – 9 – 10 – 2 – 10 – 11 – 3 – 11 – 12 – 4 – 12 – 9. Matlab code that defines the local coordinates of the spacecraft is given below.

```

1 function XYZ=spacecraftPoints
2 % define points on the spacecraft in NED coordinates
3 XYZ = [...
4     1    1    0;... % point 1
5     1   -1    0;... % point 2
6    -1   -1    0;... % point 3
7    -1    1    0;... % point 4
8     1    1    0;... % point 1
9     1    1   -2;... % point 5
10    1   -1   -2;... % point 6
11    1   -1    0;... % point 2
12    1   -1   -2;... % point 6
13   -1   -1   -2;... % point 7
14   -1   -1    0;... % point 3

```

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

```

15      -1 -1 -2;... % point 7
16      -1 1 -2;... % point 8
17      -1 1 0;... % point 4
18      -1 1 -2;... % point 8
19      1 1 -2;... % point 5
20      1 1 0;... % point 1
21      1.5 1.5 0;... % point 9
22      1.5 -1.5 0;... % point 10
23      1 -1 0;... % point 2
24      1.5 -1.5 0;... % point 10
25      -1.5 -1.5 0;... % point 11
26      -1 -1 0;... % point 3
27      -1.5 -1.5 0;... % point 11
28      -1.5 1.5 0;... % point 12
29      -1 1 0;... % point 4
30      -1.5 1.5 0;... % point 12
31      1.5 1.5 0;... % point 9
32  ]';

```

The configuration of the spacecraft is given by the Euler angles ϕ , θ , and ψ , which represent the roll, pitch, and yaw angles, respectively, and p_n , p_e , p_d , which represent the north, east, and down positions, respectively. The points on the spacecraft can be rotated and translated using the Matlab code listed below.

```

1 function XYZ=rotate(XYZ,phi,theta,psi)
2 % define rotation matrix
3 R_roll = [...
4     1, 0, 0;...
5     0, cos(phi), -sin(phi);...
6     0, sin(phi), cos(phi)];
7 R_pitch = [...
8     cos(theta), 0, sin(theta);...
9     0, 1, 0;...
10    -sin(theta), 0, cos(theta)];
11 R_yaw = [...
12    cos(psi), -sin(psi), 0;...
13    sin(psi), cos(psi), 0;...
14    0, 0, 1];
15 R = R_roll*R_pitch*R_yaw;
16 % rotate vertices
17 XYZ = R*XYZ;

```

```

1 function XYZ = translate(XYZ,pn,pe,pd)
2 XYZ = XYZ + repmat([pn;pe;pd],1,size(XYZ,2));

```

Drawing the spacecraft at the desired location is accomplished using the following Matlab code:

```

1 function handle =...
2     drawSpacecraftBody(pn,pe,pd,phi,theta,psi, handle)
3 % define points on spacecraft in local NED coordinates

```

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

```

4     NED = spacecraftPoints;
5     % rotate spacecraft by phi, theta, psi
6     NED = rotate(NED,phi,theta,psi);
7     % translate spacecraft to [pn; pe; pd]
8     NED = translate(NED,pn,pe, pd);
9     % transform vertices from NED to XYZ (for rendering)
10    R = [...
11        0, 1, 0;...
12        1, 0, 0;...
13        0, 0, -1;...
14    ];
15    XYZ = R*NED;
16    % plot spacecraft
17    if isempty(handle),
18        handle = plot3(XYZ(1,:),XYZ(2,:),XYZ(3,:));
19    else
20        set(handle,'XData',XYZ(1,:),'YData',XYZ(2,:),...
21             'ZData',XYZ(3,:));
22        drawnow
23    end

```

Lines 9–14 are used to transform the coordinates from the north-east-down (NED) coordinate frame to the drawing frame used by Matlab which has the x -axis to the viewer’s right, the y -axis into the screen, and the z -axis up. The `plot3` command is used in Line 17 to render the original drawing, and the `set` command is used to change the `XData`, `YData`, and `ZData` in Line 19. A Simulink file that can be used to debug the animation is on the book website. A rendering of the spacecraft is shown in Figure a.6.

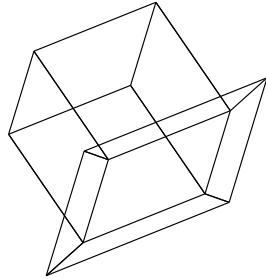


Figure a.6: Rendering of the spacecraft using lines and the `plot3` command.

The disadvantage of implementing the animation using the function `spacecraftPoints` to define the spacecraft points is that this function is called each time the animation is updated. Since the points are static, they only need to be defined once. The Simulink mask function can be used to define the points at the beginning of the simulation. Masking the `drawSpacecraft` m-file in Simulink, and then clicking on `Edit Mask` brings up a window like the one shown in Figure a.7. The spacecraft points can be defined in the initialization window, as shown in

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

Figure a.7, and passed to the `drawSpacecraft` m-file as a parameter.

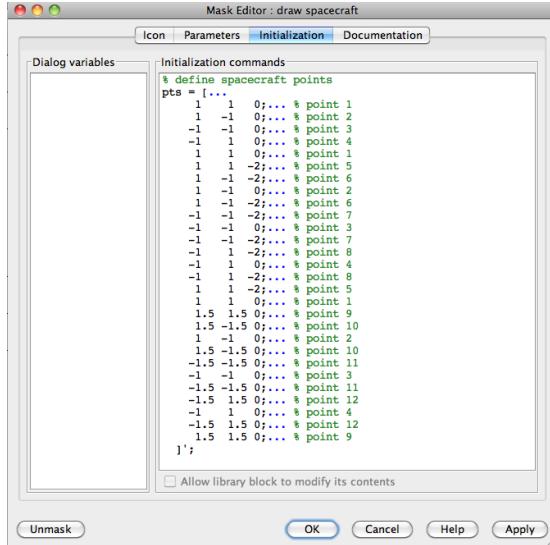


Figure a.7: The mask function in Simulink allows the spacecraft points to be initialized at the beginning of the simulation.

Appendix a.6 Animation Example: Spacecraft Using Vertices and Faces

The stick-figure drawing shown in Figure a.6 can be improved visually by using the vertex-face structure in Matlab. Instead of using the `plot3` command to draw a continuous line, we will use the `patch` command to draw faces defined by vertices and colors. The vertices, faces, and colors for the spacecraft are defined in the Matlab code listed below.

```

1  function [V, F, patchcolors]=spacecraftVFC
2  % Define the vertices (physical location of vertices
3  V = [...
4      1 1 0;... % point 1
5      1 -1 0;... % point 2
6      -1 -1 0;... % point 3
7      -1 1 0;... % point 4
8      1 1 -2;... % point 5
9      1 -1 -2;... % point 6
10     -1 -1 -2;... % point 7
11     -1 1 -2;... % point 8
12     1.5 1.5 0;... % point 9
13     1.5 -1.5 0;... % point 10
14     -1.5 -1.5 0;... % point 11
];

```

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

```

15      -1.5  1.5  0;... % point 12
16  ];
17  % define faces as a list of vertices numbered above
18  F = [...;
19      1, 2, 6, 5;... % front
20      4, 3, 7, 8;... % back
21      1, 5, 8, 4;... % right
22      2, 6, 7, 3;... % left
23      5, 6, 7, 8;... % top
24      9, 10, 11, 12;... % bottom
25  ];
26  % define colors for each face
27  myred    = [1, 0, 0];
28  mygreen   = [0, 1, 0];
29  myblue    = [0, 0, 1];
30  myyellow  = [1, 1, 0];
31  mycyan    = [0, 1, 1];
32  patchcolors = [...;
33      myred;... % front
34      mygreen;... % back
35      myblue;... % right
36      myyellow;... % left
37      mycyan;... % top
38      mycyan;... % bottom
39  ];

```

The vertices are shown in Figure a.5 and are defined in Lines 3–16. The faces are defined by listing the indices of the points that define the face. For example, the front face, defined in Line 19, consists of points 1 – 2 – 6 – 5. Faces can be defined by N -points, where the matrix that defines the faces has N columns, and the number of rows is the number of faces. The color for each face is defined in Lines 32–39. Matlab code that draws the spacecraft body is listed below.

```

1  function handle =...
2      drawSpacecraftBody(pn,pe,pd,phi,theta,psi, handle)
3  % define points on spacecraft
4  [V, F, patchcolors] = spacecraftVFC;
5  V = rotate(V', phi, theta, psi)'; % rotate spacecraft
6  V = translate(V', pn, pe, pd)'; % translate spacecraft
7  R = [...;
8      0, 1, 0;...
9      1, 0, 0;...
10     0, 0, -1;...
11  ];
12  % transform vertices from NED to XYZ (for rendering)
13  V = V*R;
14  if isempty(handle),
15      handle = patch('Vertices', V, 'Faces', F, ...
16                      'FaceVertexCData', patchcolors, ...
17                      'FaceColor', 'flat');
18  else
19      set(handle, 'Vertices', V, 'Faces', F);
20  end

```

The transposes in Lines 3 and 4 are used because the physical positions in the

APPENDIX A. CREATING ANIMATIONS IN SIMULINK

vertices matrix V are along the rows instead of the columns. A rendering of the spacecraft using vertices and faces is given in Figure a.8.

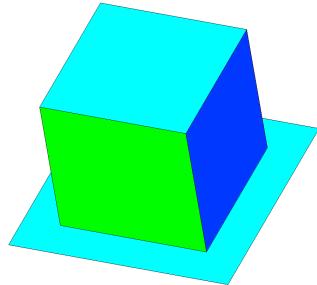


Figure a.8: Rendering of the spacecraft using vertices and faces.

Additional examples using the vertex-face format can be found at the book website.

Appendix b

Modeling in Simulink Using S-functions

This appendix assumes basic familiarity with the Matlab/Simulink environment. For additional information, please consult the Matlab/Simulink documentation. Simulink is essentially a sophisticated tool for solving interconnected hybrid ordinary differential and difference equations. Each block in Simulink is assumed to have the structure

$$\dot{x}_c = f(t, x_c, x_d, u); \quad x_c(0) = x_{c0} \quad (b.1)$$

$$x_d[k+1] = g(t, x_c, x_d, u); \quad x_d[0] = x_{d0} \quad (b.2)$$

$$y = h(t, x_c, x_d, u), \quad (b.3)$$

where $x_c \in \mathbb{R}^{n_c}$ is a continuous state with initial condition x_{c0} , $x_d \in \mathbb{R}^{n_d}$ is a discrete state with initial condition x_{d0} , $u \in \mathbb{R}^m$ is the input to the block, $y \in \mathbb{R}^p$ is the output of the block, and t is the elapsed simulation time. An *s-function* is a Simulink tool for explicitly defining the functions f , g , and h and the initial conditions x_{c0} and x_{d0} . As explained in the Matlab/Simulink documentation, there are a number of methods for specifying an s-function. In this Appendix, we will overview the method of using a level-1 m-file s-function.

Appendix b.1 Second-order Differential Equation

In this section we will show how to implement a system specified by the standard second-order transfer function

$$Y(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} U(s) \quad (b.4)$$

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

using a level-1 m-file s-function. The first step is to represent Equation (b.4) in state space form. Using control canonical form [4], we have

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -2\zeta\omega_n & -\omega_n^2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u \quad (\text{b.5})$$

$$y = \begin{pmatrix} 0 & \omega_n^2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \quad (\text{b.6})$$

The code listing for an m-file s-function that implements the system described by Equations (b.5) and (b.6) is shown below.

```

1   function [sys,x0,str,ts]...
2       = second_order_m(t,x,u,flag,zeta,wn)
3       switch flag,
4           case 0, % initialize block
5               [sys,x0,str,ts]=mdlInitializeSizes;
6               case 1, % define xdot = f(t,x,u)
7                   sys=mdlDerivatives(t,x,u,zeta,wn);
8               case 3, % define xup = g(t,x,u)
9                   sys=mdlOutputs(t,x,u,wn);
10              otherwise,
11                  sys = [];
12              end
13
14 %=====
15 function [sys,x0,str,ts]=mdlInitializeSizes
16     sizes = simsizes;
17     sizes.NumContStates = 2;
18     sizes.NumDiscStates = 0;
19     sizes.NumOutputs = 1;
20     sizes.NumInputs = 1;
21     sizes.DirFeedthrough = 0;
22     sizes.NumSampleTimes = 1;
23     sys = simsizes(sizes);
24
25     x0 = [0; 0]; % define initial conditions
26     str = []; % str is always an empty matrix
27     % initialize the array of sample times
28     ts = [0 0]; % continuous sample time
29
30 %=====
31 function xdot=mdlDerivatives(t,x,u,zeta,wn)
32     xdot(1) = -2*zeta*wn*x(1) - wn^2*x(2) + u;
33     xdot(2) = x(1);
34
35 %=====
36 function y=mdlOutputs(t,x,u,wn)
37     y = wn^2*x(2);

```

Lines 1-2 defines the main m-file function. The inputs to this function are always the elapsed time t ; the state x , which is a concatenation of the continuous state and discrete state; the input u ; and a flag, followed by user defined input parameters, which in this case are ζ and ω_n . The Simulink engine calls the s-function and passes the parameters t , x , u , and flag. When

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

`flag==0`, the Simulink engine expects the s-function to return the structure `sys`, which defines the block; initial conditions `x0`; an empty string `str`; and an array `ts` that defines the sample times of the block. When `flag==1`, the Simulink engine expects the s-function to return the function $f(t, x, u)$; when `flag==2`, the Simulink engine expects the s-function to return $g(t, x, u)$; and when `flag==3` the Simulink engine expects the s-function to return $h(t, x, u)$. The switch statement that calls the proper functions based on the value of `flag` is shown in Lines 3–12. The block setup and the definition of the initial conditions are shown in Lines 14–28. The number of continuous states, discrete states, outputs, and inputs are defined in Lines 17–20, respectively. The direct feedthrough term on Line 21 is set to one if the output depends explicitly on the input u . For example, if $D \neq 0$ in the linear state-space output equation $y = Cx + Du$. The initial conditions are defined on Line 25. The sample times are defined on Line 28. The format for this line is `ts = [period offset]`, where `period` defines the sample period and is 0 for continuous time or -1 for inherited, and where `offset` is the sample time offset, which is typically 0. The function $f(t, x, u)$ is defined in Lines 31–33, and the output function $h(t, x, u)$ is defined in Lines 36–37. A Simulink file that calls this m-file s-function is at <http://controlbook.byu.edu>.

Appendix b.2 Design Study A. Single Link Robot Arm



Homework Problem A.b

Modify the simulink model created in homework A.a by creating an s-function that implements the equations of motion. The input to the s-function should be a slider for torque. The output should go to the animation developed in homework A.a.

Solution

The s-function is listed below.

```

1  function [sys,x0,str,ts,simStateCompliance] = arm_dynamics(t,x,u,flag,P)
2  switch flag
3
4      %%%%%%
5      % Initialization %
6      %%%%%%
7      case 0
8          [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(P);
9
10     %%%%%%
11     % Derivatives %

```

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

```

12    %%%%%%
13    case 1
14        sys=mdlDerivatives(t,x,u, P);
15
16    %%%%%%
17    % Update %
18    %%%%%%
19    case 2
20        sys=mdlUpdate(t,x,u);
21
22    %%%%%%
23    % Outputs %
24    %%%%%%
25    case 3
26        sys=mdlOutputs(t,x,u);
27
28    %%%%%%
29    % GetTimeOfNextVarHit %
30    %%%%%%
31    case 4
32        sys=mdlGetTimeOfNextVarHit(t,x,u);
33
34    %%%%%%
35    % Terminate %
36    %%%%%%
37    case 9
38        sys=mdlTerminate(t,x,u);
39
40    %%%%%%
41    % Unexpected flags %
42    %%%%%%
43    otherwise
44        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
45
46    end
47
48    % end sfuntmpl
49
50    %
51    %=====
52    % mdlInitializeSizes
53    %=====
54    function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(P)
55
56    sizes = simsizes;
57
58    sizes.NumContStates = 2;
59    sizes.NumDiscStates = 0; % system parameters
60    sizes.NumOutputs = 3;
61    sizes.NumInputs = 1;
62    sizes.DirFeedthrough = 0;
63    sizes.NumSampleTimes = 1; % at least one sample time is needed
64
65    sys = simsizes(sizes);
66
67    %
68    % initialize the initial conditions

```

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

```

69 %
70 x0 = [P.theta0; P.thetadot0];
71
72 %
73 %
74 % str is always an empty matrix
75 %
76 str = [];
77
78 %
79 % initialize the array of sample times
80 %
81 ts = [0 0];
82
83 simStateCompliance = 'UnknownSimState';
84 % end mdlInitializeSizes
85
86 %
87 %=====
88 % mdlDerivatives
89 % Return the derivatives for the continuous states.
90 %=====
91 %
92 function sys=mdlDerivatives(t,x,u,P)
93 theta = x(1);
94 thetadot = x(2);
95 tau = u(1);
96
97 % system parameters randomly generated to make the
98 % system uncertain
99 persistent m
100 persistent ell
101 persistent b
102 persistent g
103 if t==0
104 alpha = 0.2; % uncertainty parameter
105 m = P.m * (1+2*alpha*rand-alpha); % kg
106 ell = P.ell * (1+2*alpha*rand-alpha); % m
107 b = P.b * (1+2*alpha*rand-alpha); % N m
108 g = P.g; % m/s^2
109 end
110
111 thetaddot = (3/m/ell^2)*(tau - b*thetadot...
112 - m*g*ell/2*cos(theta));
113
114 sys = [thetadot; thetaddot];
115
116 % end mdlDerivatives
117
118 %
119 %=====
120 % mdlUpdate
121 %=====
122 %
123 function sys=mdlUpdate(t,x,u)
124 sys = [];
125

```

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

```
126 % end mdlUpdate
127 %
128 %=====
129 % mdlOutputs
130 % Return the block outputs.
131 %=====
132 %
133 %
134 function sys=mdlOutputs(t,x,u)
135     theta = x(1);
136
137     sys = [theta; x];
138
139 % end mdlOutputs
140
141 %
142 %=====
143 % mdlGetTimeOfNextVarHit
144 %=====
145 function sys=mdlGetTimeOfNextVarHit(t,x,u)
146
147 sampleTime = 1; % Example, set the next hit to be one second later.
148 sys = t + sampleTime;
149
150 % end mdlGetTimeOfNextVarHit
151
152 %
153 %=====
154 % mdlTerminate
155 % Perform any end of simulation tasks.
156 %=====
157 %
158 function sys=mdlTerminate(t,x,u)
159
160 sys = [];
161
162 % end mdlTerminate
```

See <http://controlbook.byu.edu> for the complete solution.

Appendix b.3 Design Study B. Inverted Pendulum



Homework Problem B.b

Modify the simulink model created in homework B.a by creating an s-function that implements the equations of motion. The input to the s-function should be a slider for force. The output should go to the animation developed in homework B.a.

Solution

The s-function is listed below.

```

1  function [sys,x0,str,ts,simStateCompliance] = pendulumDynamics(t,x,u,flag,P)
2  switch flag,
3
4      %%%%%%%%%%%%%%
5      % Initialization %
6      %%%%%%%%%%%%%%
7      case 0,
8          [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(P);
9
10     %%%%%%%%%%%%%%
11     % Derivatives %
12     %%%%%%%%%%%%%%
13     case 1,
14         sys=mdlDerivatives(t,x,u,P);
15
16     %%%%%%%%%%%%%%
17     % Update %
18     %%%%%%%%%%%%%%
19     case 2,
20         sys=mdlUpdate(t,x,u);
21
22     %%%%%%%%%%%%%%
23     % Outputs %
24     %%%%%%%%%%%%%%
25     case 3,
26         sys=mdlOutputs(t,x,u);
27
28     %%%%%%%%%%%%%%
29     % GetTimeOfNextVarHit %
30     %%%%%%%%%%%%%%
31     case 4,
32         sys=mdlGetTimeOfNextVarHit(t,x,u);
33
34     %%%%%%%%%%%%%%
35     % Terminate %
36     %%%%%%%%%%%%%%
37     case 9,
38         sys=mdlTerminate(t,x,u);
39
40     %%%%%%%%%%%%%%
41     % Unexpected flags %
42     %%%%%%%%%%%%%%
43     otherwise
44         DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
45
46 end
47
48 % end sfuntmpl
49
50 %
51 =====
52 % mdlInitializeSizes

```

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

```

53 % Return the sizes, initial conditions, and sample times for the S-function.
54 %=====
55 %
56 function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(P)
57 %
58 %
59 % call simsizes for a sizes structure, fill it in and convert it to a
60 % sizes array.
61 %
62 % Note that in this example, the values are hard coded. This is not a
63 % recommended practice as the characteristics of the block are typically
64 % defined by the S-function parameters.
65 %
66 sizes = simsizes;
67 %
68 sizes.NumContStates = 4;
69 sizes.NumDiscStates = 0;
70 sizes.NumOutputs = 2+4;
71 sizes.NumInputs = 1;
72 sizes.DirFeedthrough = 0;
73 sizes.NumSampleTimes = 1; % at least one sample time is needed
74 %
75 sys = simsizes(sizes);
76 %
77 %
78 % initialize the initial conditions
79 %
80 x0 = [P.z0; P.theta0; P.zdot0; P.thetadot0];
81 %
82 %
83 % str is always an empty matrix
84 %
85 str = [];
86 %
87 %
88 % initialize the array of sample times
89 %
90 ts = [0 0];
91 %
92 % Specify the block simStateCompliance. The allowed values are:
93 %   'UnknownSimState', < The default setting; warn and assume DefaultSimState
94 %   'DefaultSimState', < Same sim state as a built-in block
95 %   'HasNoSimState', < No sim state
96 %   'DisallowSimState' < Error out when saving or restoring the model sim state
97 simStateCompliance = 'UnknownSimState';
98 %
99 % end mdlInitializeSizes
100 %
101 %
102 %=====
103 % mdlDerivatives
104 % Return the derivatives for the continuous states.
105 %=====
106 %
107 function sys=mdlDerivatives(t,x,u,P)
108     z = x(1);
109     theta = x(2);

```

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

```

110 zdot      = x(3);
111 thetadot = x(4);
112 F        = u(1);
113
114 %%%%%%%%%%%%%%
115 % The parameters for any physical system are never known exactly.
| Feedback
116 % systems need to be robust to this uncertainty. In the simulation
117 % we model uncertainty by changing the physical parameters by a uniform random variable
118 % that represents alpha*100 % of the parameter, i.e., alpha = 0.2, means that the parameter
119 % may change by up to 20%. A different parameter value is chosen every time the simulation
120 % is run.
121 persistent m1
122 persistent m2
123 persistent ell
124 persistent b
125 persistent g
126 if t==0
127     alpha = 0.2; % uncertainty parameter
128     m1 = P.m1 * (1+2*alpha*rand-alpha); % kg
129     m2 = P.m2 * (1+2*alpha*rand-alpha); % kg
130     ell = P.ell * (1+2*alpha*rand-alpha); % m
131     b = P.b * (1+2*alpha*rand-alpha); % N m
132     g = P.g; % the gravity vector is well known and so we don't change it.
133 end
134
135 % define system dynamics xdot=f(x,u)
136 M = [...
137     m1+m2,           m1*ell/2*cos(theta);...
138     m1*ell/2*cos(theta), m1*ell^2/4;...
139 ];
140 C = [...
141     m1*ell/2*thetadot^2*sin(theta) + F - b*zdot;...
142     m1*g*ell/2*sin(theta);...
143 ];
144 tmp = M\c;
145 zddot    = tmp(1);
146 thetaddot = tmp(2);
147
148 sys = [zdot; thetadot; zddot; thetaddot];
149
150 % end mdlDerivatives
151
152 %
153 =====
154 % mdlUpdate
155 % Handle discrete state updates, sample time hits, and major time step
156 % requirements.
157 =====
158 %
159 function sys=mdlUpdate(t,x,u)
160
161 sys = [];
162
163 % end mdlUpdate
164
165 %

```

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

```

166 %=====
167 % mdlOutputs
168 % Return the block outputs.
169 %=====
170 %
171 function sys=mdlOutputs(t,x,u)
172     z         = x(1);
173     theta    = x(2);
174     % add Gaussian noise to the outputs
175     z_m = z;% + 0.01*randn;
176     theta_m = theta;% + 0.001*randn;
177     sys = [z_m; theta_m; x];
178
179 % end mdlOutputs
180
181 %
182 %=====
183 % mdlGetTimeOfNextVarHit
184 % Return the time of the next hit for this block. Note that the result is
185 % absolute time. Note that this function is only used when you specify a
186 % variable discrete-time sample time [-2 0] in the sample time array in
187 % mdlInitializeSizes.
188 %=====
189 %
190 function sys=mdlGetTimeOfNextVarHit(t,x,u)
191
192 sampleTime = 1; % Example, set the next hit to be one second later.
193 sys = t + sampleTime;
194
195 % end mdlGetTimeOfNextVarHit
196
197 %
198 %=====
199 % mdlTerminate
200 % Perform any end of simulation tasks.
201 %=====
202 %
203 function sys=mdlTerminate(t,x,u)
204
205 sys = [];
206
207 % end mdlTerminate

```

See <http://controlbook.byu.edu> for the complete solution.

Appendix b.4 Design Study C. Satellite Attitude Control



Homework Problem C.b

Modify the simulink model created in homework C.a by creating an s-function that implements the equations of motion. The input to the s-function should

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

be a slider for torque. The output should go to the animation developed in homework [C.a](#).

Solution

The s-function is listed below.

```
1 function [sys,x0,str,ts,simStateCompliance] = satellite_dynamics(t,x,u,flag,P)
2 switch flag
3
4 %%%%%%
5 % Initialization %
6 %%%%%%
7 case 0
8 [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(P);
9
10 %%%%%%
11 % Derivatives %
12 %%%%%%
13 case 1
14 sys=mdlDerivatives(t,x,u);
15
16 %%%%%%
17 % Update %
18 %%%%%%
19 case 2
20 sys=mdlUpdate(t,x,u);
21
22 %%%%%%
23 % Outputs %
24 %%%%%%
25 case 3
26 sys=mdlOutputs(t,x,u);
27
28 %%%%%%
29 % GetTimeOfNextVarHit %
30 %%%%%%
31 case 4
32 sys=mdlGetTimeOfNextVarHit(t,x,u);
33
34 %%%%%%
35 % Terminate %
36 %%%%%%
37 case 9
38 sys=mdlTerminate(t,x,u);
39
40 %%%%%%
41 % Unexpected flags %
42 %%%%%%
43 otherwise
44 DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
45
46 end
47
48 % end sfuntmpl
```

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

```

49
50 %
51 %=====
52 % mdlInitializeSizes
53 %=====
54 function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(P)
55
56 sizes = simsizes;
57
58 sizes.NumContStates = 4;
59 sizes.NumDiscStates = 0;
60 sizes.NumOutputs = 6;
61 sizes.NumInputs = 1;
62 sizes.DirFeedthrough = 0;
63 sizes.NumSampleTimes = 1; % at least one sample time is needed
64
65 sys = simsizes(sizes);
66
67 %
68 % initialize the initial conditions
69 %
70 x0 = [P.theta0; P.phi0; P.thetadot0; P.phidot0];
71
72 %
73 % str is always an empty matrix
74 %
75 str = [];
76
77 %
78 % initialize the array of sample times
79 %
80 ts = [0 0];
81
82 simStateCompliance = 'UnknownSimState';
83
84 % end mdlInitializeSizes
85
86 %
87 %=====
88 % mdlDerivatives
89 % Return the derivatives for the continuous states.
90 %=====
91 function sys=mdlDerivatives(t,x,u)
92 theta = x(1);
93 phi = x(2);
94 thetadot = x(3);
95 phidot = x(4);
96 tau = u(1);
97
98 % system parameters randomly generated to make the
99 % system uncertain
100 persistent Js
101 persistent Jp
102 persistent k
103 persistent b
104 if t==0
105     alpha = 0.2; % uncertainty parameter

```

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

```

106     Js = 5; % kg m^2
107     Jp = 1; % kg m^2
108     k = 0.15; % N m
109     b = 0.05; % N m s
110 end
111
112 M = [...
113     Js, 0, 0, Jp;...
114 ];
115 c = [...
116     tau - b*(thetadot-phidot)-k*(theta-phi);...
117     -b*(phidot-thetadot)-k*(phi-theta);...
118 ];
119
120 tmp = inv(M)*c;
121 thetaddot = tmp(1);
122 phiddot = tmp(2);
123
124 sys = [thetadot; phidot; thetaddot; phiddot];
125
126 % end mdlDerivatives
127
128 %
129 %=====
130 % mdlUpdate
131 %=====
132 function sys=mdlUpdate(t,x,u)
133
134 sys = [];
135
136 % end mdlUpdate
137
138 %
139 %=====
140 % mdlOutputs
141 % Return the block outputs.
142 %=====
143 function sys=mdlOutputs(t,x,u)
144     theta = x(1);
145     phi = x(2);
146
147 sys = [theta; phi; x];
148
149 % end mdlOutputs
150
151 %
152 %=====
153 % mdlGetTimeOfNextVarHit
154 %=====
155 function sys=mdlGetTimeOfNextVarHit(t,x,u)
156
157 sampleTime = 1; % Example, set the next hit to be one second later.
158 sys = t + sampleTime;
159
160 % end mdlGetTimeOfNextVarHit
161
162 %

```

APPENDIX B. MODELING IN SIMULINK USING S-FUNCTIONS

```
163 %=====
164 % mdlTerminate
165 % Perform any end of simulation tasks.
166 %=====
167 function sys=mdlTerminate(t,x,u)
168
169 sys = [];
170 % end mdlTerminate
```

See <http://controlbook.byu.edu> for the complete solution.

Appendix c

Review of Ordinary Differential Equations

This appendix provides a refresher on how to solve linear ordinary differential equations with constant coefficients. In particular, we focus on the step response of such system.

Appendix c.1 First order ODE's

This section provides a review of how to use an integrating factor to find the solution to a scalar first order differential equations.

Consider the first order linear constant coefficient ordinary differential equation

$$\frac{dy}{dt}(t) = ay(t) + bu(t), \quad (\text{c.1})$$

with initial condition $y(0) = y_0$, where a and b are constants. Equation (c.1) can be rearranged as

$$\frac{dy}{dt}(t) - ay(t) = bu(t). \quad (\text{c.2})$$

Multiplying both sides of Equation (c.2) by the integrating factor e^{-at} gives

$$e^{-at} \left(\frac{dy}{dt}(t) - ay(t) \right) = e^{-at}bu(t). \quad (\text{c.3})$$

Noting that the left hand side of Equation (c.3) is the derivative of $e^{-at}y(t)$ gives

$$\frac{d}{dt} (e^{-at}y(t)) = e^{-at}bu(t). \quad (\text{c.4})$$

Integrating both sides of Equation (c.4) from 0 to t yields

$$\int_0^t \frac{d}{d\tau} (e^{-a\tau}y(\tau)) d\tau = \int_0^t e^{-a\tau}bu(\tau)d\tau,$$

APPENDIX C. REVIEW OF ORDINARY DIFFERENTIAL EQUATIONS

which implies that

$$e^{-at}y(t) - e^{-a \cdot 0}y(0) = \int_0^t e^{-a\tau}bu(\tau)d\tau.$$

Solving for $y(t)$ gives

$$y(t) = \underbrace{e^{at}y_0}_{\text{homogeneous response}} + \underbrace{\int_0^t e^{a(t-\tau)}bu(\tau)d\tau}_{\text{forced response}}, \quad (\text{c.5})$$

where the first term on the right is the homogenous response, or the response to initial conditions, and the second term on the right is the forced response, or the response to the input $u(t)$, which should be noted is the convolution of $u(t)$ with the impulse response e^{at} .

For the discussion to follow, it is important to note that if a is a complex number, then Equation (c.5) still holds, but that $y(t)$ will be a complex function of time. To see this, assume that $a = \sigma + j\omega$, then

$$\begin{aligned} e^{at} &= e^{(\sigma+j\omega)t} \\ &= e^{\sigma t}e^{j\omega t} \\ &= e^{\sigma t}(\cos(\omega t) + j \sin(\omega t)) \\ &= e^{\sigma t} \cos(\omega t) + j e^{\sigma t} \sin(\omega t), \end{aligned}$$

substituting into Equation (c.5) yields the complex function

$$\begin{aligned} y(t) &= \left(e^{\sigma t} \cos(\omega t) y_0 + \int_0^t e^{\sigma(t-\tau)} \cos(\omega\tau) bu(\tau)d\tau \right) \\ &\quad + j \left(e^{\sigma t} \sin(\omega t) y_0 + \int_0^t e^{\sigma(t-\tau)} \sin(\omega(t-\tau)) bu(\tau)d\tau \right). \end{aligned}$$

Note that if the real part of a is negative, or in other words if a is in the left half of the complex plane, then the homogeneous response decays to zero and the forced response is bounded if $u(t)$ is bounded.

Appendix c.2 Inverse Laplace Transform

Taking the Laplace transform of Equation (c.1) gives

$$(sY(s) - y_0) = aY(s) + bU(s).$$

Solving for $Y(s)$ yields

$$Y(s) = \left(\frac{1}{s-a} \right) y_0 + \left(\frac{b}{s-a} \right) U(s).$$

APPENDIX C. REVIEW OF ORDINARY DIFFERENTIAL EQUATIONS

Taking the inverse Laplace transform gives

$$y(t) = \mathcal{L}^{-1} \left(\frac{1}{s-a} \right) y_0 + \mathcal{L}^{-1} \left(\frac{b}{s-a} \right) * \mathcal{L}^{-1}(U(s)), \quad (\text{c.6})$$

where $*$ denotes the convolution operator. Comparing Equation (c.5) with Equation (c.6) implies that

$$\mathcal{L}^{-1} \left(\frac{b}{s-a} \right) = b e^{at}, \quad t \geq 0. \quad (\text{c.7})$$

In the limit, as $a \rightarrow 0$ we get

$$\mathcal{L}^{-1} \left(\frac{b}{s} \right) = b, \quad t \geq 0. \quad (\text{c.8})$$

Again we note that Equations (c.7) and (c.8) are true even when a and b are complex numbers.

Appendix c.3 Partial Fraction Expansion

The inverse Laplace transform for first order systems can be used to find the response to step inputs, for first order and for higher order systems, through the use of partial fraction expansion. For example, consider the task of finding the step response for the first order system

$$\dot{y} = -ay + bu, \quad (\text{c.9})$$

where a and b are real positive constants, where u is a step input of size A , i.e.,

$$u(t) = \begin{cases} A & t \geq 0 \\ 0 & \text{otherwise} \end{cases}.$$

The Laplace transform of $u(t)$ is $U(s) = \frac{A}{s}$. Similarly, taking the Laplace transform of Equation (c.9) and solving for $Y(s)$ gives

$$Y(s) = \left(\frac{b}{s+a} \right) U(s) = \frac{bA}{s(s+a)}. \quad (\text{c.10})$$

The partial fraction expansion of $Y(s)$ expresses $Y(s)$ as a linear combination of each factor in the denominator, i.e.,

$$Y(s) = \frac{c_1}{s} + \frac{c_2}{s+a}. \quad (\text{c.11})$$

To find the coefficients c_1 and c_2 , set Equation (c.11) equal to Equation (c.10) to obtain

$$\frac{c_1}{s} + \frac{c_2}{s+a} = \frac{bA}{s(s+a)}. \quad (\text{c.12})$$

APPENDIX C. REVIEW OF ORDINARY DIFFERENTIAL EQUATIONS

To find c_1 multiply both sides of Equation (c.12) by s to obtain

$$c_1 + \frac{sc_2}{s+a} = \frac{bA}{s+a}. \quad (\text{c.13})$$

To remove the term involving c_2 in Equation (c.13), set $s = 0$ to obtain

$$c_1 = \frac{bA}{a}.$$

To find c_2 multiply both sides of Equation (c.12) by $s+a$ to obtain

$$\frac{c_1(s+a)}{s} + c_2 = \frac{bA}{s}. \quad (\text{c.14})$$

To remove the term involving c_1 in Equation (c.14), set $s = -a$ to obtain

$$c_2 = -\frac{bA}{a}.$$

Therefore,

$$Y(s) = \frac{\frac{bA}{a}}{s} - \frac{\frac{bA}{a}}{s+a}. \quad (\text{c.15})$$

Since the inverse Laplace transform is a linear operator, we get

$$\begin{aligned} y(t) &= \mathcal{L}^{-1}\left(\frac{\frac{bA}{a}}{s}\right) - \mathcal{L}^{-1}\left(\frac{\frac{bA}{a}}{s+a}\right) \\ &= \frac{bA}{a} - \frac{bA}{a}e^{-at}, \quad t \geq 0 \\ &= \frac{bA}{a}(1 - e^{-at}), \quad t \geq 0. \end{aligned}$$

The same technique works for finding the step response for higher order systems. As an example, find the unit step response of the system described by

$$y^{(4)} + 10y^{(3)} + 35\ddot{y} + 50\dot{y} + 24y = u.$$

Taking the Laplace transform and solving for $Y(s)$, and using the fact that $U(s) = \frac{1}{s}$ gives

$$\begin{aligned} Y(s) &= \left(\frac{1}{s^4 + 10s^3 + 35s^2 + 50s + 24}\right)U(s) \\ &= \left(\frac{1}{s^4 + 10s^3 + 35s^2 + 50s + 24}\right)\frac{1}{s} \\ &= \frac{1}{s} \left(\frac{1}{s+1}\right) \left(\frac{1}{s+2}\right) \left(\frac{1}{s+3}\right) \left(\frac{1}{s+4}\right). \end{aligned} \quad (\text{c.16})$$

Using partial fraction expansion we have that

$$Y(s) = \frac{c_1}{s} + \frac{c_2}{s+1} + \frac{c_3}{s+2} + \frac{c_4}{s+3} + \frac{c_5}{s+4}. \quad (\text{c.17})$$

APPENDIX C. REVIEW OF ORDINARY DIFFERENTIAL EQUATIONS

Setting Equation (c.17) equal to Equation (c.16) gives

$$\frac{c_1}{s} + \frac{c_2}{s+1} + \frac{c_3}{s+2} + \frac{c_4}{s+3} + \frac{c_5}{s+4} = \frac{1}{s} \left(\frac{1}{s+1} \right) \left(\frac{1}{s+2} \right) \left(\frac{1}{s+3} \right) \left(\frac{1}{s+4} \right). \quad (\text{c.18})$$

The coefficient c_1 is found by multiplying both sides of Equation (c.18) by s and then setting $s = 0$ to remove the coefficients c_2 – c_5 to obtain $c_1 = 1/24$. Following a similar procedure for each coefficient yields $c_2 = -1/6$, $c_3 = 1/4$, $c_4 = -1/6$, and $c_5 = 1/24$. The step response is therefore given by

$$\begin{aligned} y(t) &= \mathcal{L}^{-1} \left(\frac{1/24}{s} \right) + \mathcal{L}^{-1} \left(\frac{-1/6}{s+1} \right) + \mathcal{L}^{-1} \left(\frac{1/4}{s+2} \right) + \mathcal{L}^{-1} \left(\frac{-1/6}{s+3} \right) + \mathcal{L}^{-1} \left(\frac{1/24}{s+4} \right) \\ &= \frac{1}{24} - \frac{1}{6}e^{-t} + \frac{1}{4}e^{-2t} - \frac{1}{6}e^{-3t} + \frac{1}{24}e^{-4t}, \quad t \geq 0 \\ &= \frac{1}{24} (1 - 4e^{-t} + 6e^{-2t} - 4e^{-3t} + e^{-4t}), \quad t \geq 0. \end{aligned}$$

In Matlab, the coefficients can be found by using the `residue` command. For the problem above we could use the matlab commands

```

1 num = 1; % numerator polynomial
2 den = poly([0,-1,-2,-3,-4]); % denominator polynomial
3 [R,P,K] = residue(num,den); % compute partial fraction expansion

```

The vector \mathbf{R} will contain the coefficients c_1 – c_5 , the vector \mathbf{P} contains the associated roots, and \mathbf{K} will be the remainder polynomial if the numerator is a higher order polynomial than the denominator.

The method of partial fraction expansion is also valid if the poles of the system are complex. For example, consider finding the response of the second order differential equation

$$\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y = \omega_n^2 u,$$

to a step of size A . Taking the Laplace transform and letting $U(s) = A/s$ gives

$$\begin{aligned} Y(s) &= \left(\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + 2\omega_n^2} \right) \frac{A}{s} \\ &= \frac{A\omega_n^2}{s} \left(\frac{1}{s + \zeta\omega_n + j\omega_n\sqrt{1 - \zeta^2}} \right) \left(\frac{1}{s + \zeta\omega_n - j\omega_n\sqrt{1 - \zeta^2}} \right). \quad (\text{c.19}) \end{aligned}$$

Expanding $Y(s)$ as a sum of partial fractions yields

$$Y(s) = \frac{c_1}{s} + \left(\frac{c_2}{s + \zeta\omega_n + j\omega_n\sqrt{1 - \zeta^2}} \right) + \left(\frac{c_3}{s + \zeta\omega_n - j\omega_n\sqrt{1 - \zeta^2}} \right). \quad (\text{c.20})$$

APPENDIX C. REVIEW OF ORDINARY DIFFERENTIAL EQUATIONS

Setting Equation (c.20) equal to Equation (c.19) gives

$$\begin{aligned} \frac{c_1}{s} + \left(\frac{c_2}{s + \zeta\omega_n + j\omega_n\sqrt{1 - \zeta^2}} \right) + \left(\frac{c_3}{s + \zeta\omega_n - j\omega_n\sqrt{1 - \zeta^2}} \right) \\ = \frac{A\omega_n^2}{s} \left(\frac{1}{s + \zeta\omega_n + j\omega_n\sqrt{1 - \zeta^2}} \right) \left(\frac{1}{s + \zeta\omega_n - j\omega_n\sqrt{1 - \zeta^2}} \right). \end{aligned}$$

Solving for c_1 , c_2 , and c_3 gives

$$\begin{aligned} c_1 &= A \\ c_2 &= -\frac{A}{2} \left(1 + j \frac{\zeta}{\sqrt{1 + \zeta^2}} \right) \\ c_3 &= -\frac{A}{2} \left(1 - j \frac{\zeta}{\sqrt{1 - \zeta^2}} \right). \end{aligned}$$

Therefore,

$$Y(s) = \frac{A}{s} + \frac{-\frac{A}{2} \left(1 + j \frac{\zeta}{\sqrt{1 + \zeta^2}} \right)}{s + \zeta\omega_n + j\omega_n\sqrt{1 - \zeta^2}} + \frac{-\frac{A}{2} \left(1 - j \frac{\zeta}{\sqrt{1 - \zeta^2}} \right)}{s + \zeta\omega_n - j\omega_n\sqrt{1 - \zeta^2}}.$$

Taking the inverse Laplace transform gives

$$\begin{aligned} y(t) &= A - \frac{A}{2} \left(1 + j \frac{\zeta}{\sqrt{1 + \zeta^2}} \right) e^{-\zeta\omega_n t - j\omega_n\sqrt{1 - \zeta^2}t} \\ &\quad - \frac{A}{2} \left(1 - j \frac{\zeta}{\sqrt{1 - \zeta^2}} \right) e^{-\zeta\omega_n t + j\omega_n\sqrt{1 - \zeta^2}t}, \quad t \geq 0 \\ &= A \left(1 - \frac{1}{2} \left| 1 + j \frac{\zeta}{\sqrt{1 + \zeta^2}} \right| e^{-\zeta\omega_n t} \right. \\ &\quad \cdot \left. \left(e^{-j\omega_n\sqrt{1 - \zeta^2}t} e^{j\angle\left(1 + j \frac{\zeta}{\sqrt{1 + \zeta^2}}\right)} + e^{+j\omega_n\sqrt{1 - \zeta^2}t} e^{j\angle\left(1 - j \frac{\zeta}{\sqrt{1 - \zeta^2}}\right)} \right) \right), \quad t \geq 0 \\ &= A \left(1 - \frac{1}{\sqrt{1 - \zeta^2}} e^{-\zeta\omega_n t} \right. \\ &\quad \cdot \left. \left(\frac{e^{-j(\omega_n\sqrt{1 - \zeta^2}t - \tan^{-1}\left(\frac{\zeta}{\sqrt{1 - \zeta^2}}\right))} + e^{j(\omega_n\sqrt{1 - \zeta^2}t - \tan^{-1}\left(\frac{\zeta}{\sqrt{1 - \zeta^2}}\right))}}{2} \right) \right), \quad t \geq 0 \\ &= A \left(1 - \frac{1}{\sqrt{1 - \zeta^2}} e^{-\zeta\omega_n t} \cos\left(\omega_n\sqrt{1 - \zeta^2}t - \tan^{-1}\left(\frac{\zeta}{\sqrt{1 - \zeta^2}}\right)\right) \right), \quad t \geq 0. \end{aligned}$$

APPENDIX C. REVIEW OF ORDINARY DIFFERENTIAL EQUATIONS

In general, complex conjugate poles will always lead to complex conjugate coefficients that end up canceling to produce a decaying exponential multiplied by a cosine term with a phase shift. If the coefficients of the original ordinary differential equation are real, then the response will always be real.

Appendix d

Numerical Solutions to Differential Equations

This appendix provides a brief overview of techniques for numerically solving ordinary differential equations, when the initial value is specified. In particular, we are interested in solving the differential equation

$$\dot{x} = \frac{dx}{dt}(t) = f(x(t), u(t)), \quad x(t_0), \quad (\text{d.1})$$

over the time interval $[t_0, t_f]$, where $x(t) \in \mathbf{R}^n$, and $u(t) \in \mathbf{R}^m$ is assumed to be known, and where we assume that $f(\cdot, \cdot)$ is sufficiently smooth and that unique solutions to the differential equation exist for every x_0 .

Integrating both sides of Equation (d.1) from t_0 to $t < t_f$ gives

$$x(t) = x(t_0) + \int_{t_0}^t f(x(\tau), u(\tau)) d\tau. \quad (\text{d.2})$$

The difficulty with solving Equation (d.2) is that $x(t)$ appears on both sides of the equation, and cannot therefore be solved explicitly for $x(t)$. All numerical solution techniques attempt to approximate the solution to Equation (d.2). Most techniques break the solution into small time increments, usually equal to the sample rate of the computer, which we denote by T_s . Accordingly we write Equation (d.2) as

$$\begin{aligned} x(t) &= x(t_0) + \int_{t_0}^{t-T_s} f(x(\tau), u(\tau)) d\tau + \int_{t-T_s}^t f(x(\tau), u(\tau)) d\tau \\ &= x(t - T_s) + \int_{t-T_s}^t f(x(\tau), u(\tau)) d\tau. \end{aligned} \quad (\text{d.3})$$

The most simple form of numerical approximation for the integral in Equation (d.3) is to use the left endpoint method as shown in Figure ??, where

$$\int_{t-T_s}^t f(x(\tau), u(\tau)) d\tau \approx T_s f(x(t - T_s), u(t - T_s)).$$

APPENDIX D. NUMERICAL SOLUTIONS TO DIFFERENTIAL EQUATIONS

Using the notation $x_k \triangleq x(t_0 + kT_s)$ we get the numerical approximation to the differential equation in Equation (d.1) as

$$x_k = x_{k-1} + T_s f(x_k, u_k), \quad x_0 = x(t_0). \quad (\text{d.4})$$

Equation (d.4) is called the Euler method, or a Runge-Kutta first order method (RK1).

While the RK1 method is often effective for numerically solving ODEs, it typically requires a small sample size T_s . The advantage of the RK1 method is that it only requires one evaluation of $f(\cdot, \cdot)$.

To improve the numerical accuracy of the solution, a second order method can be derived by approximating the integral in Equation (d.3) using the trapezoidal rule shown in Figure ??, where

$$\int_a^b f(\xi) d\xi \approx (b-a) \left(\frac{f(a) + f(b)}{2} \right).$$

Accordingly,

$$\int_{t-T_s}^t f(x(\tau), u(\tau)) d\tau \approx \frac{T_s}{2} (f(x(t-T_s), u(t-T_s)) + f(x(t), u(t))).$$

Since $x(t)$ is unknown, we use the RK1 method to approximate it as

$$x(t) \approx x(t-T_s) + T_s f(x(t-T_s), u(t-T_s)).$$

In addition, we typically assume that $u(t)$ is constant over the interval $[t-T_s, t]$ to get Therefore

$$\int_{t-T_s}^t f(x(\tau), u(\tau)) d\tau \approx \frac{T_s}{2} (f(x(t-T_s), u(t-T_s)) + f(x(t-T_s) + T_s f(x(t-T_s), u(t-T_s)), u(t-T_s))).$$

Accordingly, the numerical integration routine can be written as

$$\begin{aligned} x_0 &= x(t_0) \\ F_1 &= f(x_{k-1}, u_{k-1}) \\ F_2 &= f(x_{k-1} + T_s F_1, u_{k-1}) \\ x_k &= x_{k-1} + \frac{T_s}{2} (F_1 + F_2). \end{aligned} \quad (\text{d.5})$$

Equations (d.5) is the Runge-Kutta second order method or RK2. It requires two function calls to $f(\cdot, \cdot)$ for every time sample.

The most common numerical method for solving ODEs is the Runge-Kutta forth order method RK4. The RK4 method is derived using Simpson's Rule

$$\int_a^b f(\xi) d\xi \approx \left(\frac{b-a}{6} \right) \left(f(a) + 4f \left(\frac{a+b}{2} \right) + f(b) \right),$$

APPENDIX D. NUMERICAL SOLUTIONS TO DIFFERENTIAL EQUATIONS

which is found by approximating the area under the integral using a parabola. The standard strategy is to write the approximation as

$$\int_a^b f(\xi) \xi \approx \left(\frac{b-a}{6} \right) \left(f(a) + 2f\left(\frac{a+b}{2}\right) + 2f\left(\frac{a+b}{2}\right) + f(b) \right),$$

and then to use

$$F_1 = f(a) \tag{d.6}$$

$$F_2 = f\left(a + \frac{T_s}{2} F_1\right) \approx f\left(\frac{a+b}{2}\right) \tag{d.7}$$

$$F_3 = f\left(a + \frac{T_s}{2} F_2\right) \approx f\left(\frac{a+b}{2}\right) \tag{d.8}$$

$$F_4 = f(a + T_s F_3) \approx f(b), \tag{d.9}$$

resulting in the RK4 numerical integration rule

$$\begin{aligned} x_0 &= x(t_0) \\ F_1 &= f(x_{k-1}, u_{k-1}) \\ F_2 &= f\left(x_{k-1} + \frac{T_s}{2} F_1, u_{k-1}\right) \\ F_3 &= f\left(x_{k-1} + \frac{T_s}{2} F_2, u_{k-1}\right) \\ F_4 &= f(x_{k-1} + T_s F_3, u_{k-1}) \\ x_k &= x_{k-1} + \frac{T_s}{6} (F_1 + 2F_2 + 2F_3 + F_4). \end{aligned} \tag{d.10}$$

It can be shown that the RK4 method matches the Taylor series approximation of the integral in Equation (d.3) up to the fourth order term. Higher order methods can be derived, but generally do not result in significantly better numerical approximations to the ODE. The step size T_s must still be chosen to be sufficiently small to result in good solutions. It is also possible to develop adaptive step size solvers. For example the ODE45 algorithm in Matlab, solves the ODE using both an RK4 and an RK5 algorithm. The two solutions are then compared and if they are sufficiently different, then the step size is reduced. If the two solutions are close, then the step size can be increased.

Notes and References

Appendix e

Root Locus

The *root locus* has played an important pedagogical tool in control systems since the 1940s. It was important in the design of feedback amplifiers for communication circuits where the classic design involved the tuning of a small number of parameters. In this appendix we introduce the root locus and show how it can be used to understand the effect of adding an integrator to a second order system with PD control. However, we should note that root locus can be used in many other settings. We are not including this appendix as a main chapter in the book because in our experience Root Locus is rarely used for actual design.

Appendix e.1 Theory

We begin by noting that for the closed loop system shown in Figure e.1, the transfer function from $Y_r(s)$ to $Y(s)$ can be computed as

$$\begin{aligned} Y(s) &= P(s)C(s)(Y_r(s) - Y(s)) \\ \implies Y(s) &= \frac{P(s)C(s)}{1 + P(s)C(s)}Y_r(s). \end{aligned}$$

Therefore, the characteristic equation of the closed loop system is given by

$$\Delta_{cl} = 1 + P(s)C(s) = 0.$$

We have seen in Chapter 7 that by using PD control for a second order system, it is possible to precisely select the locations of the closed loop poles. When an integrator is added for PID control, it is also possible to precisely select the location of the three closed-loop poles. In this chapter we will introduce a simple and convenient tool for visualizing how the closed loop poles change as a function of a single parameter. As motivation, consider a first order plant under proportional control, as shown in Figure e.2. The closed-loop transfer function

APPENDIX E. ROOT LOCUS

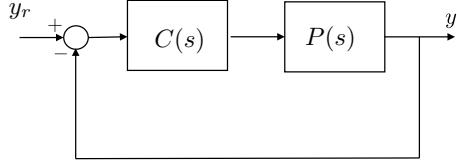


Figure e.1: General feedback system with plant $P(s)$ and controller $C(s)$.

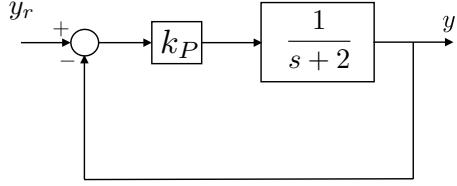


Figure e.2: A first order plant under proportional control.

is given by

$$Y(s) = \frac{k_P}{s + (2 + k_P)} Y_r(s)$$

and the closed loop characteristic polynomial is given by

$$\Delta_{cl}(s) = s + (2 + k_P).$$

Note that the location of the closed loop pole is

$$p_{cl} = -(2 + k_P).$$

We can create a plot of the location of the poles in the complex plane as shown in Figure e.3. When $k_P = 0$, the closed loop pole is located at the open loop pole -2 , but as k_P is increased greater than zero, the closed loop pole moves from -2 into the left half plane in the direction of the arrow shown in Figure e.3. The plot of the closed loop pole as a function of k_P is an example of a *root locus*.

As a second example, consider the second order system under proportional control shown in Figure e.4. In this case the closed loop transfer function is given by

$$Y(s) = \frac{k_P}{s^2 + 2s + k_P} Y_r(s)$$

Note that the open loop poles are located at 0 and -2 . The closed loop characteristic polynomial is given by

$$\Delta_{cl}(s) = s^2 + 2s + k_P \tag{e.1}$$

and the closed loop poles are located at

$$p_{cl} = -1 \pm \sqrt{1 - k_P}.$$

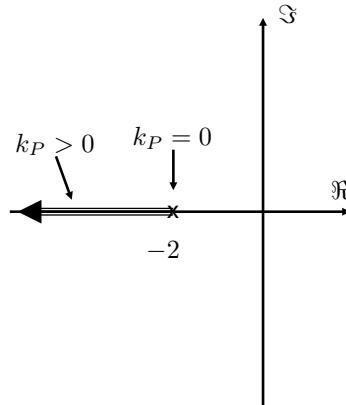


Figure e.3: A plot of the closed loop pole associated with Figure e.2 as a function of $k_P > 0$.

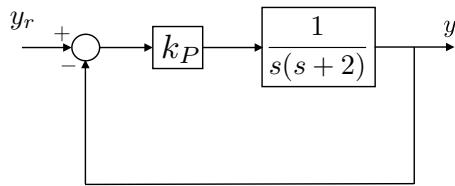


Figure e.4: A second order plant under proportional control.

The position of the closed loop poles as a function of $k_P > 0$ are shown in Figure e.5. When $k_P = 0$, the closed loop poles are equal to the open loop poles at 0 and -1 . When $0 < k_P \leq 1$, the closed loop poles are both real and move toward -1 with increasing k_P . When $k_P = 1$ both poles are located at -1 . When $k_P > 1$ the two poles are complex with real part equal to -1 and increasing complex part as k_P increases. The poles move in the direction of the arrows shown in Figure e.5.

The Matlab command `rlocus` will plot the root locus and can be used interactively to find the gain associated with specific pole locations. To use the `rlocus` command in Matlab, the characteristic equation must be put in *Evan's form*, which is

$$1 + kL(s) = 0,$$

where k is the gain of interest. For example, the characteristic equation associated with the characteristic polynomial in Equation (e.1) is

$$\Delta_{cl}(s) = s^2 + 2s + k_P = 0.$$

The characteristic equation can be put in Evan's form by dividing both sides

APPENDIX E. ROOT LOCUS

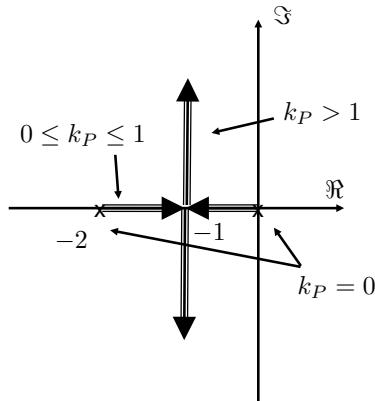


Figure e.5: A plot of the closed loop pole associated with Figure e.4 as a function of $k_P > 0$.

by $s^2 + 2s$ and rearranging to obtain

$$1 + k_P \left(\frac{1}{s^2 + 2s} \right) = 0,$$

where $L(s) = \frac{1}{s^2 + 2s}$. The Matlab command that draws the root locus for this equation is

```
1 >> L = tf([1][1,2,0]);
2 >> figure(1), clf, rlocus(L)
```

where the first line defines the transfer function $L(s) = \frac{1}{s^2 + 2s}$ and the second line invokes the root locus command using L .

The root locus tool is particularly useful for selecting the integrator gains after PD gains have been selected. For example, suppose that a PD controller has been designed for the second order plant shown in Figure e.6 so that the closed loop poles are located at $-2 \pm j2$, where $k_P = 8$ and $k_D = 2$. Now

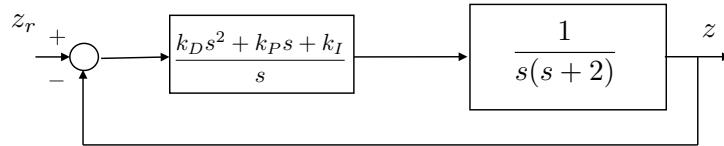


Figure e.6: Closed loop system where the proportional gains have been selected as $k_P = 8$ and $k_D = 2$.

suppose that an integrator is to be added to reject disturbances and eliminate steady state error, but that the integrator gain k_I is to be selected so that the closed loop poles deviate from $-2 \pm j2$ by less than 10%. The characteristic

APPENDIX E. ROOT LOCUS

equation for the closed loop system including the integrator is given by

$$\begin{aligned} 1 + \left(\frac{1}{s^2 + 2s} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right) &= 0 \\ \implies s^3 + (2 + k_D)s^2 + k_P s + k_I &= 0 \\ \implies 1 + k_I \left(\frac{1}{s^3 + (2 + k_D)s^2 + k_P s} \right) &= 0 \\ \implies 1 + k_I \left(\frac{1}{s^3 + 4s^2 + 8s} \right) &= 0 \end{aligned}$$

where the later equation is in Evan's form. Therefore the Matlab command for drawing the root locus is

```
1 >> L = tf([1] [1, 4, 8, 0]);
2 >> figure(1), clf, rlocus(L)
```

and the resulting plot is shown in Figure e.7. Note from Figure e.7 that the

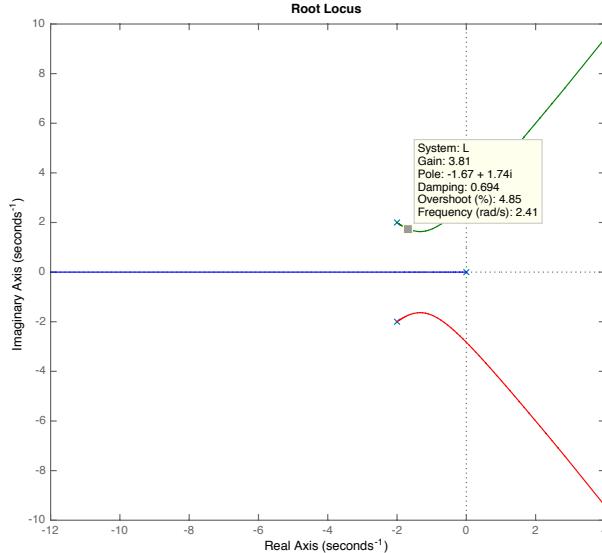


Figure e.7: The result of Matlab's `rlocus` command, and by interactively selecting pole locations close to $-2 \pm j2$.

poles when $k_I = 0$ are at $0, -2 \pm j2$ and are denoted by 'x'. The root locus shows what happens to the closed loop poles as k_I is increased from zero. Note from Figure e.7 that for large values of k_I the closed-loop poles are in the right half plane and the system will be unstable. However, for a gain of $k_I = 3.81$

the closed loop poles are located at $-1.67 \pm j1.74$. This technique can be used to select reasonable values for k_I .

Appendix e.2 Design Study A. Single Link Robot Arm



Homework Problem A.e

For the single link robot arm, use the PD gains derived in HW A.8. Add an integrator to the controller to get PID control. Put the resulting closed loop characteristic equation in Evan's form and use the Matlab `rlocus` command to plot the root locus versus the integrator gain k_I . Select a value for k_I that does not significantly change the other locations of the closed loop poles.

Solution

The closed loop block diagram including an integrator is shown in Figure e.8. The closed loop transfer function is given by

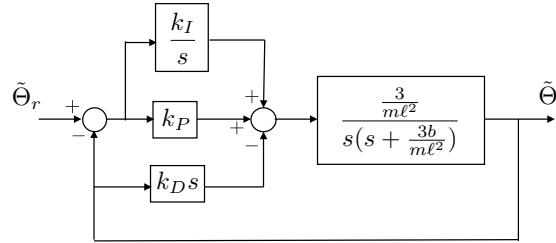


Figure e.8: PID control for the single link robot arm.

$$\tilde{\Theta}(s) = \frac{\frac{3k_P}{m\ell^2}s + \frac{3k_I}{m\ell^2}}{s^3 + \left(\frac{3b+3k_D}{m\ell^2}\right)s^2 + \frac{3k_P}{m\ell^2}s + \frac{3k_I}{m\ell^2}} \tilde{\Theta}^d(s).$$

The characteristic equation is therefore

$$s^3 + \left(\frac{3b+3k_D}{m\ell^2}\right)s^2 + \frac{3k_P}{m\ell^2}s + \frac{3k_I}{m\ell^2} = 0.$$

In Evan's form we have

$$1 + k_I \left(\frac{\frac{3}{m\ell^2}}{s^3 + \left(\frac{3b+3k_D}{m\ell^2}\right)s^2 + \frac{3k_P}{m\ell^2}s} \right) = 0.$$

The appropriate Matlab command is therefore

```

1 >> L = tf([3/m/L^2], [1, (3*b+3*kd)/m/L^2, 3*kp/m/L^2, 0]);
2 >> figure(1), clf, rlocus(L);

```

Appendix e.3 Design Study B. Inverted Pendulum



Homework Problem B.e

Adding an integrator to obtain PID control for the outer loop of the inverted pendulum system, put the characteristic equation in Evan's form and use the Matlab `rlocus` command to plot the root locus versus the integrator gain k_I . Select a value for k_I that does not significantly change the other locations of the closed loop poles.

Solution

The closed loop block diagram for the outer loop of the inverted pendulum including an integrator is shown in Figure e.9. The closed loop transfer function

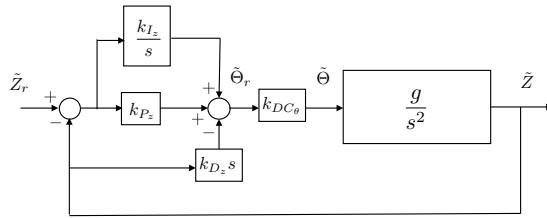


Figure e.9: PID control for the outer loop of the inverted pendulum.

is given by

$$\tilde{Z}(s) = \frac{gk_{DC_\theta}(k_{P_z}s + k_{I_z})}{s^3 + gk_{DC_\theta}k_{D_z}s^2 + gk_{DC_\theta}k_{P_z}s + gk_{DC_\theta}k_{I_z}} \tilde{Z}_r(s).$$

The characteristic equation is therefore

$$s^3 + gk_{DC_\theta}k_{D_z}s^2 + gk_{DC_\theta}k_{P_z}s + gk_{DC_\theta}k_{I_z} = 0.$$

In Evan's form we have

$$1 + k_{I_z} \left(\frac{gk_{DC_\theta}}{s^3 + gk_{DC_\theta}k_{D_z}s^2 - gk_{DC_\theta}k_{P_z}s} \right) = 0.$$

The appropriate Matlab command is therefore

```

1 >> L = tf([g*kDCth],[1,g*kDCth*kd,g*kDCth*kp,0]);
2 >> figure(1), clf, rlocus(L);

```

Appendix e.4 Design Study C. Satellite Attitude Control



Homework Problem C.e

Adding an integrator to obtain PID control for the outer loop of the satellite system, put the characteristic equation in Evan's form and use the Matlab `rlocus` command to plot the root locus verses the integrator gain k_I . Select a value for k_I that does not significantly change the other locations of the closed loop poles.

Solution

The closed loop block diagram for the outer loop including an integrator is shown in Figure e.10. The closed loop transfer function is given by

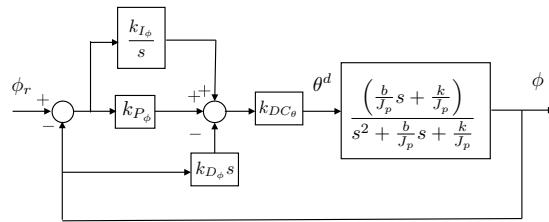


Figure e.10: PID control for the outer loop of the satellite system.

$$\Phi(s) = \frac{b_2 s^2 + b_1 s + b_0}{a_3 s^3 + a_2 s^2 + a_1 s + a_0} \Phi_r(s),$$

APPENDIX E. ROOT LOCUS

where

$$\begin{aligned}
 b_2 &= \frac{bk_{DC_\theta} k_{P_\phi}}{J_p} \\
 b_1 &= \frac{bk_{DC_\theta} k_{I_\phi}}{J_p} + \frac{kk_{DC_\theta} k_{P_\phi}}{J_p} \\
 b_0 &= \frac{kk_{DC_\theta} k_{I_\phi}}{J_p} \\
 a_3 &= 1 + \frac{bk_{DC_\theta} k_{D_\phi}}{J_p} \\
 a_2 &= \frac{b}{J_p} + \frac{kk_{DC_\theta} k_{D_\phi}}{J_p} + \frac{bk_{DC_\theta} k_{P_\phi}}{J_p} \\
 a_1 &= \frac{k}{J_p} + \frac{bk_{DC_\theta} k_{I_\phi}}{J_p} + \frac{kk_{DC_\theta} k_{P_\phi}}{J_p} \\
 a_0 &= \frac{kk_{DC_\theta} k_{I_\phi}}{J_p}.
 \end{aligned}$$

The characteristic equation is therefore

$$\begin{aligned}
 \left(1 + \frac{bk_{DC_\theta} k_{D_\phi}}{J_p}\right) s^3 + \left(\frac{b}{J_p} + \frac{kk_{DC_\theta} k_{D_\phi}}{J_p} + \frac{bk_{DC_\theta} k_{P_\phi}}{J_p}\right) s^2 \\
 + \left(\frac{k}{J_p} + \frac{bk_{DC_\theta} k_{I_\phi}}{J_p} + \frac{kk_{DC_\theta} k_{P_\phi}}{J_p}\right) s + \frac{kk_{DC_\theta} k_{I_\phi}}{J_p} = 0.
 \end{aligned}$$

In Evan's form we have

$$1 + k_{I_\phi} \left(\frac{\frac{bk_{DC_\theta}}{J_p} s + \frac{kk_{DC_\theta}}{J_p}}{\left(1 + \frac{bk_{DC_\theta} k_{D_\phi}}{J_p}\right) s^3 + \left(\frac{b}{J_p} + \frac{kk_{DC_\theta} k_{D_\phi}}{J_p} + \frac{bk_{DC_\theta} k_{P_\phi}}{J_p}\right) s^2 + \left(\frac{k}{J_p} + \frac{kk_{DC_\theta} k_{P_\phi}}{J_p}\right) s} \right) = 0.$$

The appropriate Matlab command is therefore

```

1  >> L = tf([b*kDC/Jp, k*kDC/Jp], ...
2           [1+b*kDC*kD/Jp, b/Jp+k*kDC*kD/Jp+b*kDC*kP/Jp, ...
3            k/Jp+k*kDC*kP/Jp, 0]);
4  >> figure(1), clf, rlocus(L);

```

Notes and References

Appendix f

Review of Linear Algebra

Let A be an $m \times n$ matrix of real numbers, denoted as $A \in \mathbb{R}^{m \times n}$. Then

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}.$$

Matrix addition is defined element wise for matrices of compatible dimensions as

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}}_A + \underbrace{\begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & & & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix}}_B = \underbrace{\begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \vdots & & & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{pmatrix}}_C.$$

To multiply a matrix times a vector, the inner dimensions must match:

$$\underbrace{A}_{m \times n} \underbrace{x}_{n \times 1} = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n a_{1i}x_i \\ \vdots \\ \sum_{i=1}^n a_{mi}x_i \end{pmatrix} = \underbrace{b}_{m \times 1}.$$

For example,

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 \\ 6 \end{pmatrix} = \begin{pmatrix} 17 \\ 39 \end{pmatrix}.$$

APPENDIX F. REVIEW OF LINEAR ALGEBRA

Multiplying two matrices also requires matching dimensions:

$$\underbrace{A}_{m \times n} \underbrace{B}_{n \times p} = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} b_{11} & \dots & b_{1n} \\ \vdots & & \vdots \\ b_{m1} & \dots & b_{mn} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{i=1}^n a_{1i}b_{i1} & \dots & \sum_{i=1}^n a_{1i}b_{ip} \\ \vdots & & \vdots \\ \sum_{i=1}^n a_{mi}b_{i1} & \dots & \sum_{i=1}^n a_{mi}b_{ip} \end{pmatrix} = \underbrace{C}_{m \times p}.$$

Note that if $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, then AB is a valid product, but BA is not a valid product. Even if the dimensions are compatible, i.e., if A and B are square, in general

$$AB \neq BA.$$

Appendix f.1 Transpose

If $A \in \mathbb{R}^{m \times n}$, then the transpose of A , denoted as $A^\top \in \mathbb{R}^{n \times m}$ is obtained by interchanging the rows and the columns. For example

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^\top = (x_1 \quad x_2 \quad x_3),$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}^\top = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}.$$

A square matrix A is said to be *symmetric* if $A^\top = A$. Similarly, A is said to be *skew-symmetric* if $A^\top = -A$. Every square matrix $A \in \mathbb{R}^{n \times n}$ can be decomposed into the sum of a symmetric and a skew-symmetric matrix as

$$\begin{aligned} A &= \frac{1}{2}A + \frac{1}{2}A + \frac{1}{2}A^\top - \frac{1}{2}A^\top \\ &= \left(\frac{1}{2}A + \frac{1}{2}A^\top\right) + \left(\frac{1}{2}A - \frac{1}{2}A^\top\right) \\ &= A_s + A_{ss}, \end{aligned}$$

where $A_s = \frac{1}{2}(A + A^\top)$ is symmetric, and $A_{ss} = \frac{1}{2}(A - A^\top)$ is skew-symmetric.

The following properties are easily verified by direct manipulation:

$$(AB)^\top = B^\top A^\top$$

$$(A + B)^\top = A^\top + B^\top.$$

Appendix f.2 Trace

The trace operator is defined for square matrices. If $A \in \mathbb{R}^{n \times n}$, then

$$\text{tr}(A) = \sum_{i=1}^n a_{ii},$$

or in other words, the sum of the diagonals of A . For example

$$\text{tr} \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = 1 + 4 = 5.$$

Appendix f.3 Determinant

The *determinant* operator is defined for square matrices $A \in \mathbb{R}^{n \times n}$ as

$$\det(A) = \sum_{j=1}^n a_{ij} \gamma_{ij},$$

for any $i = 1, \dots, n$, where the *co-factor*

$$\gamma_{ij} = (-1)^{(i+j)} \det(M_{ij})$$

and where the minor M_{ij} is the $(n - 1) \times (n - 1)$ matrix obtained by removing the i^{th} row and the j^{th} column of A . As an example,

$$\begin{aligned} \det \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 0 \\ 6 & 7 & 8 \end{pmatrix} &= 1 \cdot \det \begin{pmatrix} 5 & 0 \\ 7 & 8 \end{pmatrix} - 2 \cdot \det \begin{pmatrix} 4 & 0 \\ 6 & 8 \end{pmatrix} + 3 \cdot \det \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix} \\ &= 1 \cdot (5 \cdot 8 - 7 \cdot 0) - 2 \cdot (4 \cdot 8 - 6 \cdot 0) + 3 \cdot (4 \cdot 7 - 6 \cdot 5) = -30. \end{aligned}$$

The *adjugate* of A is defined as the transpose of the co-factors of A :

$$\text{adj}(A) = \begin{pmatrix} \gamma_{11} & \dots & \gamma_{n1} \\ \vdots & & \vdots \\ \gamma_{1n} & \dots & \gamma_{nn} \end{pmatrix}.$$

We have the following properties for the determinant:

$$\begin{aligned} \det(A) &= \det(A^\top) \\ \det(AB) &= \det(A)\det(B) \\ \det(A^{-1}) &= \frac{1}{\det(A)}. \end{aligned}$$

There are also well defined rules for finding the determinant of a block matrix. Let

$$E = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

APPENDIX F. REVIEW OF LINEAR ALGEBRA

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$, $C \in \mathbb{R}^{m \times n}$ and $D \in \mathbb{R}^{m \times m}$, then it can be shown that if A is invertible, then

$$\det(E) = \det(A) \det(D - CA^{-1}B).$$

As a special case, if either $B = 0$ or $C = 0$, then

$$\det(E) = \det(A) \det(D).$$

For example,

$$\det \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 0 & 0 & 0 & 8 & 9 \\ 0 & 0 & 0 & 9 & 10 \end{pmatrix} = \det \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{pmatrix} \cdot \det \begin{pmatrix} 8 & 9 \\ 9 & 10 \end{pmatrix}.$$

Appendix f.4 Matrix Inverses

Definition. If $A \in \mathbb{R}^{n \times n}$, and there exists a matrix $B \in \mathbb{R}^{n \times n}$ such that

$$AB = BA = I,$$

then B is said to be the inverse of A and is denoted as $B = A^{-1}$.

The inverse of any square matrix can be expressed in terms of its adjugate and its determinant as

$$A^{-1} = \frac{\text{adj}(A)}{\det(A)}.$$

For a 2×2 matrix we have the simple formula

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{\text{adj} \begin{pmatrix} a & b \\ c & d \end{pmatrix}}{\det \begin{pmatrix} a & b \\ c & d \end{pmatrix}} = \frac{\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}}{ad - bc}.$$

If $\det(A) \neq 0$, then A is called *non-singular*.

If A and B are nonsingular, then

$$(AB)^{-1} = B^{-1}A^{-1}.$$

For block matrices it can be verified by direct substitution that

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1} \end{pmatrix} \quad (\text{f.1})$$

$$= \begin{pmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{pmatrix}, \quad (\text{f.2})$$

APPENDIX F. REVIEW OF LINEAR ALGEBRA

from which we get the famous matrix inversion lemma

$$(A - BD^{-1}C)^{-1} = A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1}. \quad (\text{f.3})$$

When $C = 0$ we get the simple formula

$$\begin{pmatrix} A & B \\ 0 & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}BD^{-1} \\ 0 & D^{-1} \end{pmatrix}.$$

Appendix f.5 Eigenvalues and Eigenvectors

The pair (λ, v) where $\lambda \in \mathbb{C}$ and $0 \neq v \in \mathbb{C}^{n \times 1}$ is said to be an eigenvalue-eigenvector pair of the matrix $A \in \mathbb{R}^{n \times n}$ if

$$Av = \lambda v.$$

The following statements are equivalent

1. (λ, v) are an eigenvalue-eigenvector pair of A ,
2. $Av = \lambda v$,
3. $(\lambda I - A)v = 0$, where I is the $n \times n$ identity matrix,
4. v is in the null space of $(\lambda I - A)$,
5. $\det(\lambda I - A) = 0$.

For example, to find the eigenvalues and eigenvectors of

$$A = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix},$$

first compute

$$\begin{aligned} \det(\lambda I - A) &= \det \begin{pmatrix} \lambda & -1 \\ 2 & \lambda + 3 \end{pmatrix} \\ &= \lambda^2 + 3\lambda + 2 = (\lambda + 2)(\lambda + 1). \end{aligned}$$

Setting $\det(\lambda I - A) = 0$ implies that the eigenvalues are

$$\lambda_1 = -2, \quad \lambda_2 = -1.$$

To find the eigenvector associated with λ_1 we need to find a vector v_1 that is in the null space of $(\lambda_1 I - A)$. Since

$$(\lambda_1 I - A)v_1 = \begin{pmatrix} -2 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} v_{11} \\ v_{12} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

we have that $2v_{11} = -v_{12}$. Therefore $v_1 = (1, -2)^\top$ is in the null space of $(\lambda_1 I - A)$ and is an eigenvector of A associated with eigenvalue λ_1 .

APPENDIX F. REVIEW OF LINEAR ALGEBRA

To find the eigenvector associated with λ_2 we need to find a vector v_2 that is in the null space of $(\lambda_2 I - A)$. Since

$$(\lambda_2 I - A)v_2 = \begin{pmatrix} -1 & -1 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} v_{21} \\ v_{22} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

we have that $v_{21} = -v_{22}$. Therefore $v_2 = (-1, 1)^\top$ is in the null space of $(\lambda_2 I - A)$ and is an eigenvector of A associated with eigenvalue λ_2 .

Since $Av_1 = \lambda_1 v_1$ and $Av_2 = \lambda_2 v_2$ we can write

$$A(v_1 \ v_2) = (v_1 \ v_2) \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.$$

Defining $M = (v_1 \ v_2)$ and $\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$ we have

$$AM = M\Lambda.$$

When M is invertible, we have

$$A = M\Lambda M^{-1}.$$

M will always be invertible when the eigenvalues are distinct. For example, it can be verified that

$$\begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} -2 & 0 \\ 0 & 11 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ -2 & 1 \end{pmatrix}^{-1}.$$

The formula $A = M\Lambda M^{-1}$ is called the eigen-decomposition of A .

The trace and the determinant of A can be written in terms of the eigenvalues of A as

$$\begin{aligned} \text{tr}(A) &= \sum_{i=1}^n \lambda_i(A), \\ \det(A) &= \prod_{i=1}^n \lambda_i(A). \end{aligned}$$

where $\lambda_i(A)$ is the i^{th} eigenvalue of A .

Appendix f.6 Linear Independence and Rank

The vectors $x_1, x_2, \dots, x_p \in \mathbb{R}^{n \times 1}$ are said to be *linearly independent* if

$$c_1 x_1 + c_2 x_2 + \cdots + c_p x_p = 0$$

implies that $c_1 = c_2 = \cdots = c_p = 0$. For example, the vectors $(1, 0)^\top$ and $(0, 1)^\top$ are linearly independent since

$$c_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + c_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

APPENDIX F. REVIEW OF LINEAR ALGEBRA

implies that $c_1 = c_2 = 0$. Alternatively, the vectors $(1, 0)^\top$ and $(3, 0)^\top$ are not linearly independent since

$$c_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + c_2 \begin{pmatrix} 3 \\ 0 \end{pmatrix} = \begin{pmatrix} c_1 + 3c_2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

is true if $c_1 = -3c_2$ for any c_2 .

The *rank* of a matrix is defined to be the number of linearly independent rows or columns. For example

$$\begin{aligned} \text{rank} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} &= 2 \\ \text{rank} \begin{pmatrix} 1 & 3 \\ 0 & 0 \end{pmatrix} &= 1. \end{aligned}$$

Appendix f.7 Special Matrices

The matrix $A \in \mathbb{R}^{n \times n}$ is said to be *diagonal* if

$$A = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix}.$$

For example,

$$A = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

is a diagonal matrix.

The matrix $A \in \mathbb{R}^{n \times n}$ is said to be *upper triangular* if

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix}.$$

For example,

$$A = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 0 & 6 & 7 & 8 \\ 0 & 0 & 9 & 10 \\ 0 & 0 & 0 & 11 \end{pmatrix}$$

is an upper triangular matrix.

APPENDIX F. REVIEW OF LINEAR ALGEBRA

The matrix $A \in \mathbb{R}^{n \times n}$ is said to be in *upper companion form* if

$$A = \begin{pmatrix} -a_1 & -a_2 & \dots & -a_{n-1} & -a_n \\ 1 & 0 & \dots & 0 & 0 \\ \vdots & \ddots & & \vdots & 0 \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix}.$$

For example,

$$A = \begin{pmatrix} -2 & -3 & -4 & -5 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

is in upper companion form.

The matrix $A \in \mathbb{R}^{n \times n}$ is said to be a *Vandermonde matrix* if

$$A = \begin{pmatrix} 1 & a_1 & a_1^2 & \dots & a_1^{n-1} \\ 1 & a_2 & a_2^2 & \dots & a_2^{n-1} \\ 1 & a_3 & a_3^2 & \dots & a_3^{n-1} \\ 1 & a_4 & a_4^2 & \dots & a_4^{n-1} \end{pmatrix}.$$

For example,

$$A = \begin{pmatrix} 1 & 2 & 4 & 9 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \\ 1 & 5 & 25 & 125 \end{pmatrix}$$

is a Vandermonde matrix.

Bibliography

- [1] H. Goldstein, *Classical Mechanics*. Addison-Wesley, 1951.
- [2] V. I. Arnold, *Mathematical Methods of Classical Mechanics*. New York, New York: Springer Verlag, 1989.
- [3] J. Hanc, E. F. Taylor, and S. Tuleja, “Deriving Lagrange’s Equations using Elementary Calculus,” *American Journal of Physics*, vol. 72, pp. 510–513, apr 2014.
- [4] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Addison Wesley, 4rd ed., 2002.
- [5] J. P. Hespanha, *Linear Systems Theory*. Princeton University Press, 2009.
- [6] T. Kailath, *Linear Systems*. Englewood Cliffs, New Jersey: Prentice Hall, 1980.
- [7] C.-T. Chen, *Linear System Theory and Design*. Oxford University Press, 1999.
- [8] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2006.
- [9] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. New York, New York: McGraw Hill, 1987.
- [10] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall, third edit ed., 2005.
- [11] H. K. Khalil, *Nonlinear Control*. Pearson, 2015.
- [12] A. Isidori, *Nonlinear Control Systems*. Communication and Control Engineering, New York, New York: Springer Verlag, 2nd ed., 1989.
- [13] H. Nijmeijer and A. J. van der Schaft, *Nonlinear Dynamical Control Systems*. New York, New York: Springer Verlag, 1990.
- [14] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*. Springer, 1999.

BIBLIOGRAPHY

- [15] Krstić, Kanellakopaulis, and Kokotovic, *Nonlinear and Adaptive Control Design*. Wiley, 1995.
- [16] A. V. Oppenheim and A. S. Willsky, *Signals & Systems*. Prentice Hall, second ed., 1997.
- [17] P. M. DeRusso, R. J. Roy, C. M. Close, and A. A. Desrochers, *State Variables for Engineers*. Wiley, second ed., 1998.
- [18] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems*. Addison Wesley, third edit ed., 1998.
- [19] Dorato, Abdallah, and Cerone, *Linear-Quadratic Control: An Introduction*. Prentice Hall, 1995.
- [20] H. K. Khalil, *Nonlinear Systems*. Upper Saddle River, NJ: Prentice Hall, 3rd ed., 2002.
- [21] D. G. Luenberger, “An Introduction to Observers,” *{IEEE} Transactions on Automatic Control*, vol. 16, pp. 596–602, dec 1971.
- [22] K. J. Astrom and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2009.
- [23] R. C. Dorf and R. H. Bishop, *Modern Control Systems*. Pearson Education, 13th editi ed., 2016.
- [24] F. L. Lewis, *Optimal Estimation: With an Introduction to Stochastic Control Theory*. New York, New York: John Wiley & Sons, 1986.
- [25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2006.
- [26] B. D. O. Anderson and J. B. Moore, *Linear Optimal Control*. Englewood Cliffs, New Jersey: Prentice Hall, 1971.
- [27] *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, 2008.

Index

- Ackermann's formula, 138, 178
- adjugate of a matrix, 60, 404
- analog computer, 174
- analog computer implementation, 132
- animation in Simulink, 353
- augmented state space equations, 196
- augmented system, 155
- ball and beam, 331
- block matrix, 404
- Bode canonical form, 221
- Bode phase gain relationship, 263
- Bode plot, 218, 221, 258
- cascade approximation, 50
- change of variables, 131, 136
- closed loop Bode plot, 260
- closed loop characteristic equation, 130
- closed loop characteristic polynomial, 73
- closed loop performance, 237
- closed loop poles, 73, 130
- closed loop transfer function, 73
- co-factor, 404
- companion form, 134
- complex conjugate poles, 389
- complex numbers, 215
- complex numbers - polar coordinates, 215
- complex numbers - rectangular coordinates, 215
- computed torque, 44
- conjugate of a complex number, 216
- control canonic form, 131, 133, 409
- control canonical form, 283
- control implementation, 283
- controllability matrix, 137, 138, 155
- controllable, 155
- controllable system, 138
- critically damped, 84
- cross over frequency, 257
- damping forces, 25
- damping ratio, 78, 81, 260
- DC gain, 89, 156
- DC-gain, 80, 139
- derivative control, 69
- derivative gain, 70
- design models, 35
- design process, 1
- desired characteristic polynomial, 73
- determinant of a matrix, 404
- diagonal matrix, 408
- digital differentiator, 129
- digital implementation, 184
- dirty derivative, 118
- discrete differentiator, 118
- discrete integrator, 118
- distinct eigenvalues, 407
- disturbance observer, 195
- disturbance rejection, 238, 242, 243, 256, 275

INDEX

- disturbances, 50
- eigen-decomposition, 407
- eigenvalue of a matrix, 406
- eigenvalues, 61, 130
- eigenvector, 406
- equilibrium point, 37
- estimated state, 172
- Euler Lagrange equations, 26
- Euler's relationship, 216
- Euler-Lagrange equation, 10
- Euler-Lagrange equations, 24
- fast subsystem, 49
- feedback linearization, 36, 39, 44
- final value theorem, 99
- first order response, 78
- forced response, 384
- free integrator, 104
- frequency response, 214, 218, 237, 260
- full state feedback, 130, 135
- fundamental theorem of calculus, 99
- gain margin, 259
- generalized coordinates, 9, 25
- generalized forces, 10, 25
- generalized velocity, 9
- handle graphics, 353
- homogeneous response, 384
- innovation, 173
- input disturbance, 105, 195, 237
- input disturbance rejection, 239, 243
- input saturation, 119, 262
- integral control, 69, 129, 154, 271, 273
- integral feedback, 179
- integral gain, 69
- integrating factor, 383
- integrator, 99, 117, 396
- integrator anti-windup, 118
- interaction matrix, 155
- inverse Laplace transform, 78, 81, 385
- inverse of a matrix, 405
- Jacobian linearization, 36, 38, 58
- kinetic energy, 8, 11
- Lagrangian, 26
- Laplace transform, 45, 59, 196, 384
- linear independence, 407
- linearized system, 140, 156
- linearized systems, 180
- loop gain, 258, 271
- loopshape, 256
- loopshaping, 214, 271
- loopshaping control, 3
- low pass filter, 272, 274
- magnitude of a complex number, 215
- magnitude of a transfer function, 217
- matrix, 402
- matrix addition, 402
- matrix adjugate, 404
- matrix determinant, 404
- matrix eigenvalue, 406
- matrix inverse, 405
- matrix minor, 404
- matrix multiplication, 402
- matrix rank, 407
- matrix trace, 404
- matrix transpose, 403
- measured output, 139, 172
- moments of inertia, 16
- natural frequency, 78, 81, 261
- noise, 237
- noise attenuation, 238, 240, 256, 274
- non-singular matrix, 405
- observability matrix, 178
- observable, 197
- observable system, 178
- observation error, 173

INDEX

- observer, 172, 195
- observer canonic form, 175
- observer error, 173
- observer gain, 173, 176, 178
- observer-based control, 2
- observers, 129
- open loop characteristic polynomial, 72
- open loop poles, 72, 130
- ordinary differential equations, 383
- output disturbance, 105, 237
- output disturbance rejection, 238, 242
- output equation, 56
- over damped, 84
- parallel axis theorem, 22
- partial fraction expansion, 86, 385
- PD control, 72
- phase lag filter, 272, 275
- phase lead filter, 272, 276
- phase margin, 258, 277
- phase of a complex number, 215
- phase of a transfer function, 217
- PI control, 69, 271, 273
- PID control, 2, 69, 70, 116, 396
- point mass, 11, 24
- pole locations, 78
- pole placement, 155
- poles, 46, 61
- potential energy, 8, 24
- potential energy of a spring, 25
- prefilter, 284
- proportional control, 69
- proportional gain, 69, 271, 272
- rank of a matrix, 407
- rational transfer function, 86
- realization of a transfer function, 131
- reference command, 69
- reference input, 105
- reference output, 139, 154, 172
- residue, 387
- rise time, 79, 81, 85
- root locus, 154, 393
- s-function, 369
- sample period, 118
- sampled data system, 116
- second order response, 81
- second order systems, 69
- sensor noise, 105
- separation principle, 179, 180
- simulation model, 1
- slow subsystem, 49
- stability, 256
- stability margins, 271
- state evolution equation, 56
- state feedback, 129
- state interaction matrix, 130, 135
- state space equations, 59, 130
- state space model, 56, 172
- state space models, 2, 129
- steady state error, 70, 99
- step response, 78, 86, 261, 383, 385
- successive loop closure, 88, 121, 288
- system type, 103, 106, 245, 273
- Taylor series, 36
- trace of a matrix, 404
- tracking performance, 238, 239, 256, 275
- transfer function, 46, 48, 214, 217
- transfer function model, 131
- transfer function models, 2, 59
- transpose of a matrix, 403
- Tustin approximation, 118
- type 0 system, 245
- type 1 system, 246
- type 2 system, 248
- under damped, 84
- upper companion form, 409
- upper triangular matrix, 408
- Vandermonde matrix, 409
- z-transform, 117
- zeros, 46, 60, 86