

A Design Study Approach to Classical Control

Randal W. Beard Timothy W. McLain
Brigham Young University

Updated: February 20, 2016



Contents

| | | |
|------------|--|------------|
| 1 | Introduction | 1 |
| I | Simulation Models | 10 |
| 2 | Kinetic Energy of Mechanical Systems | 14 |
| 3 | The Euler Lagrange Equations | 27 |
| II | Design Models | 38 |
| 4 | Equilibria and Linearization | 40 |
| 5 | Transfer Function Models | 49 |
| 6 | State Space Models | 62 |
| III | PID Control Design | 75 |
| 7 | Pole Placement for Second Order Systems | 80 |
| 8 | Time Domain Design Specifications | 87 |
| 9 | Successive Loop Closure | 106 |
| 10 | Saturation and limits of performance | 115 |
| 11 | System Type and Integrators | 125 |

| | |
|--|------------|
| CONTENTS | ii |
| 12 Root Locus | 145 |
| 13 Digital Implementation of PID Controllers | 154 |
| IV Observer Based Control Design | 166 |
| 14 Full State Feedback | 168 |
| 15 Integrator with Full State Feedback | 194 |
| 16 Observers | 213 |
| 17 Disturbance Observers | 234 |
| V Loopshaping Control Design | 246 |
| 18 Bode Plots | 248 |
| 19 Tracking, Disturbance Rejection, Noise Attenuation | 273 |
| 20 Stability and Robustness Margins | 296 |
| 21 Compensator Design | 314 |
| VI Design Studies | 339 |
| D. Mass Spring Damper | 340 |
| E. Ball on Beam | 348 |
| F. Gantry Crane | 357 |
| G. Planar VTOL | 367 |
| Appendices | 367 |
| A Creating Animations in Simulink | 381 |

| | |
|--|------------|
| <i>CONTENTS</i> | iii |
| B Modeling in Simulink Using S-functions | 398 |
| C Simple Derivation of the Euler Lagrange Equations | 412 |
| D Linear Algebra Review | 428 |

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 (motors, propellers, etc.) and sensors (accelerometers, gyros, GPS, camera, etc.). 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 design process. In this book we will introduce the two most common design models used for control systems analysis and design: transfer function models and the 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 shows how to linearize the equations of motion and how to create transfer function and state space design models.

In Part ?? of the book we describe time domain design specifications that are used to quantify the performance of the system. These design specifications include the rise time, settling time, and maximum percent overshoot.

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.

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 much 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 methods, that are known to have robustness issues.

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.

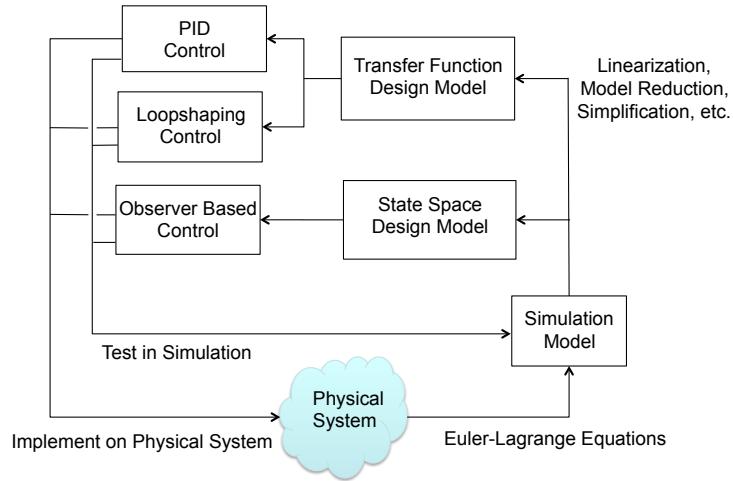


Figure 1.1: The design process. Using principles of physics, the physical system is modeled mathematically, resulting in the simulation model. The simulation model is simplified to create design models that are used for the control design. The control design is then tested and debugged in simulation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

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. The homework problems 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

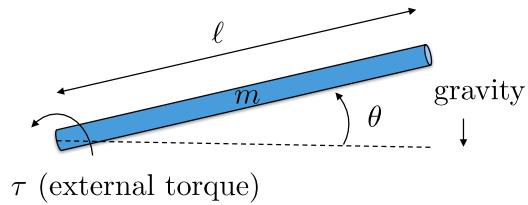


Figure 1.2: Single Link Robot Arm

Figure 1.2 shows the 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 level. The angular speed of the arm is $\dot{\theta}$. There is an applied torque τ at the joint, and assuming a damping torque that opposes the rotation of the joint of magnitude $-b\dot{\theta}$.

Assume the following physical constants: $m = 0.5$ kg, $\ell = 0.3$ m, $g = 9.8$ m/s², $b = 0.01$ Nms.

Homework Problems:

HW A.1 (2.2) Kinetic energy.

HW A.2 (1.5) Simulink animation.

HW A.3 (3.2) Equations of Motion.

HW A.4 (2.2) Simulink S-function.

HW A.5 (4.2) Linearize equations of motion.

HW A.6 (5.2) Transfer function model.

HW A.7 (6.2) State space model.

HW A.8 (7.2) Pole placement using PD control.

HW A.9 (7.2) Simulink implementation of PD control.

HW A.10 (8.2) PD design to time domain specifications.

HW A.12 (10.1) PD design to achieve fastest response subject to saturation.

HW A.13 (11.2) System type.

HW A.14 (12.2) Using root locus to chose the integrator gain.

HW A.15 (13.3) Digital implementation of PID.

HW A.16 (14.3) Full state feedback.

HW A.17 (15.3) Full state feedback with integrator.

HW A.18 (16.3) Observer based control.

HW A.19 (17.2) Disturbance observer.

HW A.20 (18.2) Bode Plot.

HW A.21 (19.2) Performance in Frequency Domain.

HW A.22 (20.2) Stability Margins.

HW A.23 (21.2) Loopshaping Design.

B Design Study: Pendulum on a Cart

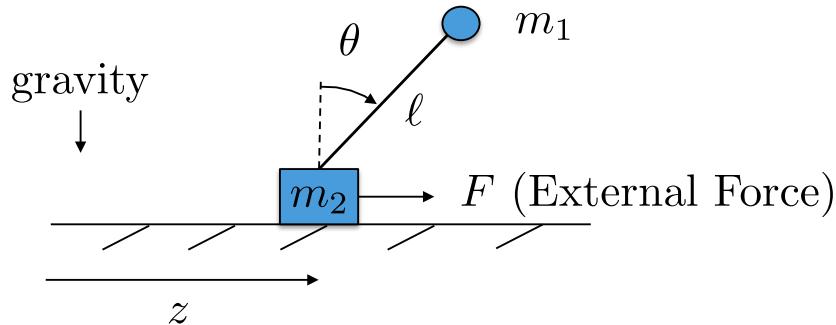


Figure 1.3: Pendulum on a cart.

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 pendulum arm is assumed to be massless, and m_1 is assumed to be a point mass. 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 $-b\dot{z}$.

Assume the following physical constants: $m_1 = 0.25$ kg, $m_2 = 1.0$ kg, $\ell = 0.5$ m, $g = 9.8$ m/s², $b = 0.05$ Ns.

Homework Problems:

HW B.1 (2.3) Kinetic energy.

HW B.2 (1.6) Simulink animation.

HW B.3 (3.3) Equations of Motion.

HW B.4 (2.3) Simulink S-function.

HW B.5 (4.3) Linearize equations of motion.

HW B.6 (5.4) Transfer Function Model.

HW B.7 (6.3) State Space Model.

HW B.11 (9.2) Successive loop closure.

HW B.12 (10.2) PD design to achieve fastest response subject to saturation.

HW B.13 (11.3) System type.

HW B.14 (12.3) Using root locus to chose the integrator gain.

HW B.15 (13.4) Digital implementation of PID.

HW B.16 (14.4) Full state feedback.

HW B.17 (15.4) Full state feedback with integrator.

HW B.18 (16.4) Observer based control.

HW B.19 (17.3) Disturbance observer.

HW B.20 (18.3) Bode Plot.

HW B.21 (19.3) Performance in Frequency Domain.

HW B.22 (20.3) Stability Margins.

HW B.23 (21.3) Loopshaping Design.

C Design Study: Satellite Attitude Control

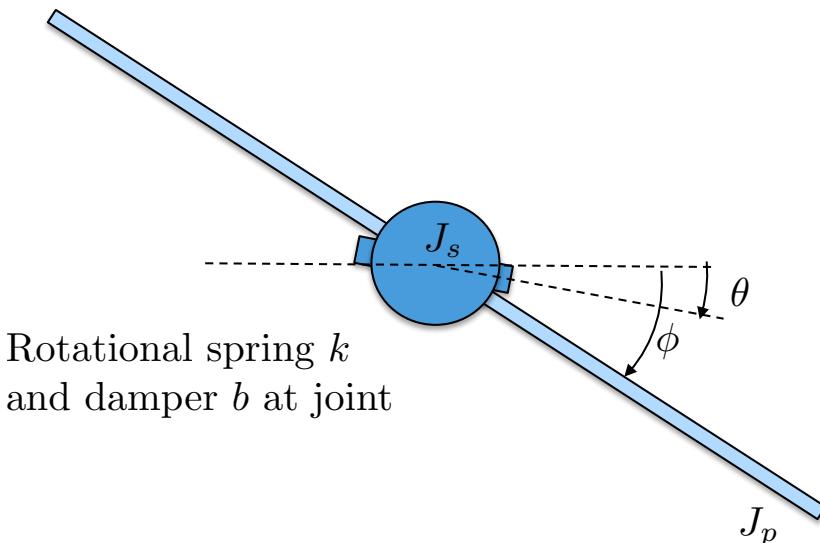


Figure 1.4: Satellite with flexible solar panels.

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 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.

Assume the following physical constants: $J_s = 5 \text{ kg m}^2$, $J_p = 1 \text{ kg m}^2$, $k = 0.15 \text{ N m}$, $b = 0.05 \text{ Nms}$.

Homework Problems:

HW C.1 (2.4) Kinetic energy.

HW C.2 (1.7) Simulink animation.

HW C.3 (3.4) Equations of Motion.

HW C.4 (2.4) Simulink S-function.

HW C.6 (5.5) Transfer Function.

HW C.7 (6.4) State Space Model.

HW C.11 (9.3) Successive loop closure.

HW C.12 (10.3) PD design to achieve fastest response subject to saturation.

HW C.13 (11.4) System type.

HW C.14 (12.4) Using root locus to chose the integrator gain.

HW C.15 (13.5) Digital implementation of PID.

HW C.16 (14.5) Full state feedback.

HW C.17 (15.5) Full state feedback with integrator.

HW C.18 (16.5) Observer based control.

HW C.19 (17.4) Disturbance observer.

HW C.20 (18.4) Bode Plot.

HW C.21 (19.4) Performance in Frequency Domain.

HW C.22 (20.4) Stability Margins.

HW C.23 (21.4) 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 in (generalized) position and velocity. The theory behind Euler-Lagrange theory 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

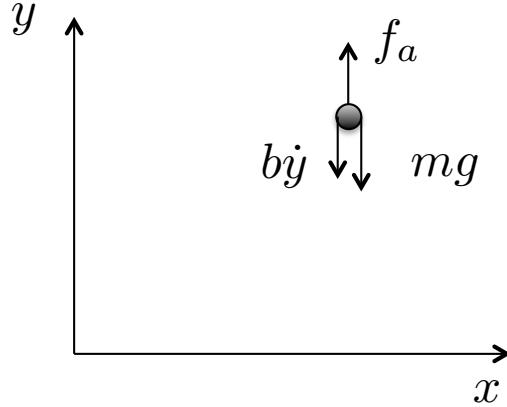


Figure 1.5: Point mass in a gravity field.

forces are in the 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

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

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

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

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

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

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

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

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

$$\begin{aligned} \frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_1} \right) - \frac{\partial L(\mathbf{q})}{\partial q_1} &= -b_1 \dot{q}_1 + \tau_1 \\ \frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_2} \right) - \frac{\partial L(\mathbf{q})}{\partial q_2} &= -b_2 \dot{q}_2 + \tau_2 \\ &\vdots \\ \frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_n} \right) - \frac{\partial L(\mathbf{q})}{\partial q_n} &= -b_n \dot{q}_n + \tau_n, \end{aligned} \quad (1.6)$$

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

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

where B is a diagonal matrix and $\boldsymbol{\tau}_a = (\tau_1, \dots, \tau_n)^\top$ are the generalized forces. The Euler-Lagrange equation will be discussed in more detail in Chapter 3. However, before showing how to use Equation (??) 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

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

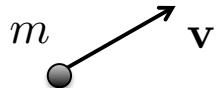


Figure 2.1: Point mass moving at velocity \mathbf{v} .

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

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

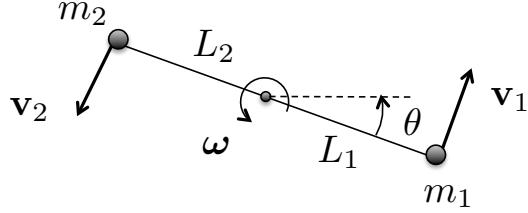


Figure 2.2: Two constrained point masses spinning about a point on adjoining axis.

If the two masses are connected by a mass-less rod, and the motion particles is caused by spinning the rod about a point on the rod, as shown in Figure 2.2, where the angular velocity vector points out of the page and the magnitude is given by $\dot{\theta}$, i.e., $\boldsymbol{\omega} = (0, 0, \dot{\theta})^\top$. The velocity of point i is given by

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

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

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

Therefore

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

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,

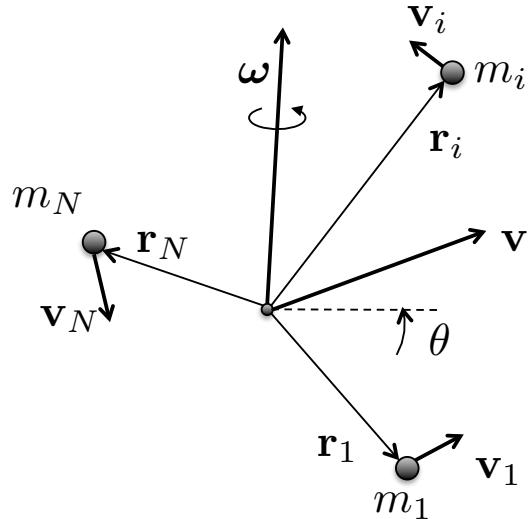


Figure 2.3: Many masses in a rigid body spinning about a common axis and moving at velocity \mathbf{v} .

$i = 1, \dots, N$ in a specified coordinate system are given by \mathbf{r}_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 $\mathbf{v}_i = \boldsymbol{\omega} \times \mathbf{r}_i$ gives

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

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

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

The matrix

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

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

$$J = \int_{\text{body}} ((\mathbf{r}_i^\top \mathbf{r}_i) I_3 - \mathbf{r}_i \mathbf{r}_i^\top) dm_i,$$

where the elemental mass dm_i is located at position \mathbf{r}_i and the integral is over all elemental masses in the system.

Suppose that $\mathbf{r}_i = (x_i, y_i, z_i)$ is expressed with respect to a coordinate system fixed in a rigid body. Then

$$\begin{aligned}
J &= \int_{\text{body}} \left[(x_i \ y_i \ z_i) \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} (x_i \ y_i \ z_i) \right] dm_i \\
&= \int_{\text{body}} \left[(x_i^2 + y_i^2 + z_i^2) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} x_i^2 & x_i y_i & x_i z_i \\ y_i x_i & y_i^2 & y_i z_i \\ z_i x_i & z_i y_i & z_i^2 \end{pmatrix} \right] dm_i \\
&= \int_{\text{body}} \begin{pmatrix} y_i^2 + z_i^2 & -x_i y_i & -x_i z_i \\ -y_i x_i & x_i^2 + z_i^2 & -y_i z_i \\ -z_i x_i & -z_i y_i & x_i^2 + y_i^2 \end{pmatrix} dm_i.
\end{aligned}$$

RWB: Insert example of finding J for a brick.

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

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

In that case the kinetic energy is given by

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

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

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

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

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

2.1.1 Example: Mass Spring System

As an example, consider the mass spring system shown in Figure 2.4 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

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

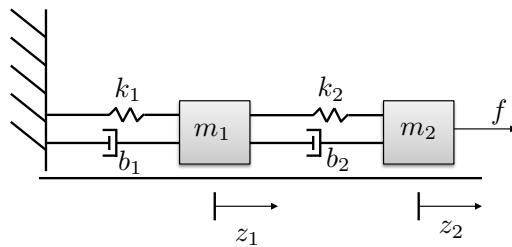


Figure 2.4: Mass spring system with two masses connected by springs and dampers.

2.1.2 Example: Spinning Dumbbell

As another simple example, consider the spinning dumbbell system shown in Figure 2.5, 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} .

To compute the kinetic energy we first find the position of the two masses. If the position of the point of rotation is given by $(z, 0, 0)^\top$, then the position of m_1 and m_2 are given by

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

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

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

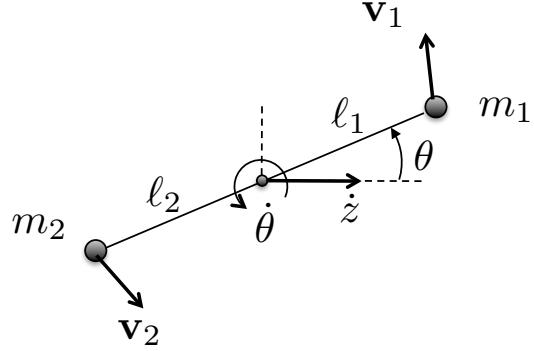


Figure 2.5: Spinning dumbbell. The system is spinning about the pointing out of the page and moving to the right at velocity \dot{z} .

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

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

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

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

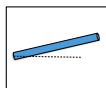
which is in the form

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

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.1

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

Solution

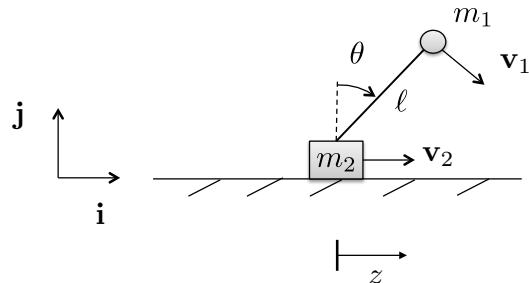


Figure 2.6: Single link robot arm. Compute the velocity of the center of mass and the angular velocity about the center of mass.

The first step is to compute the velocity about the center of mass. If the pivot point is the origin of the coordinate system as shown in Figure 2.6, then the position of the center of mass is

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

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 to compute $J_z = \int_x \int_y (x^2 + y^2) dm$.

As shown in Figure 2.7, we will assume a thin rod to avoid the double integral in calculating J_z . The inertia is computed about the center of mass,

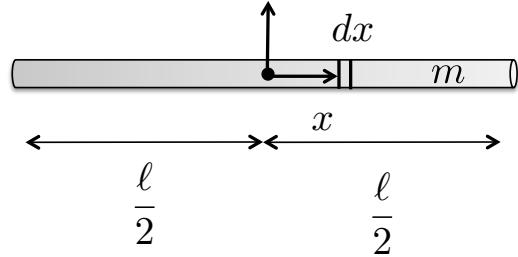


Figure 2.7: Computing the inertia of a thin rod.

where an element of mass located at position x has elemental mass $dm = \frac{m}{\ell} dx$. Therefore, the inertia J_z can be computed as

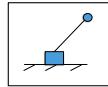
$$J_z = \int_{x=-\frac{\ell}{2}}^{\frac{\ell}{2}} x^2 \frac{m}{\ell} dx = \frac{m}{\ell} \int_{x=-\frac{\ell}{2}}^{\frac{\ell}{2}} x^2 dx = \frac{m}{\ell} \frac{x^3}{3} \Big|_{x=-\frac{\ell}{2}}^{\frac{\ell}{2}} = \frac{m}{3\ell} \left(\frac{\ell^3}{8} - \frac{-\ell^3}{8} \right) = \frac{m\ell^2}{12}.$$

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.1

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

Solution

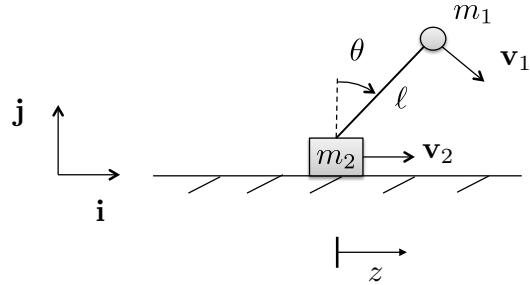


Figure 2.8: Pendulum on a cart. Compute the velocity of the bob and of the cart

The configuration of the pendulum on a cart is shown in Figure 2.8, where horizontal position of the cart with mass m_2 is given by

$$\mathbf{p}_2 = \begin{pmatrix} z \\ 0 \\ 0 \end{pmatrix},$$

and the position of the bob with mass m_1 at the end of the rod is given by

$$\mathbf{p}_1 = \begin{pmatrix} z + \ell \sin \theta \\ \ell \cos \theta \\ 0 \end{pmatrix}.$$

Differentiating to obtain the velocities of m_1 and m_2 we obtain

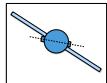
$$\mathbf{v}_1 = \begin{pmatrix} \dot{z} + \ell \dot{\theta} \cos \theta \\ -\ell \dot{\theta} \sin \theta \\ 0 \end{pmatrix}.$$

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

$$\begin{aligned}
 K &= \frac{1}{2}m_1\mathbf{v}_1^\top \mathbf{v}_1 + \frac{1}{2}m_2\mathbf{v}_2^\top \mathbf{v}_2 \\
 &= \frac{1}{2}m_1 \left[(\dot{z} + \ell\dot{\theta}\cos\theta)^2 + (-\ell\dot{\theta}\sin\theta)^2 \right] + \frac{1}{2}m_2\dot{y}^2 \\
 &= \frac{1}{2}m_1 \left[\dot{z}^2 + 2\ell\dot{z}\dot{\theta}\cos\theta + \ell^2\dot{\theta}^2\cos^2\theta + \ell^2\dot{\theta}^2\sin^2\theta \right] + \frac{1}{2}m_2\dot{z}^2 \\
 &= \frac{1}{2}(m_1 + m_2)\dot{z}^2 + \frac{1}{2}m_1\ell^2\dot{\theta}^2 + m_1\ell\dot{z}\dot{\theta}\cos\theta.
 \end{aligned} \tag{2.1}$$

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.1

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

$$K = \frac{1}{2}J_s\dot{\theta}^2 + \frac{1}{2}J_p\dot{\phi}^2. \quad (2.2)$$

Notes and References

Chapter 3

The Euler Lagrange Equations

3.1 Theory

3.1.1 Potential Energy

We will be concerned with two sources of potential energy: namely gravity potential and spring potential. Consider a point mass m in a gravity field

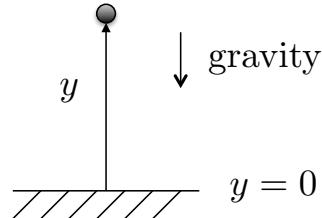


Figure 3.1: The potential energy of a unit mass m in gravity is given by $P = mgy$.

that is at position y . The potential energy is given by

$$P = mgy + P_0,$$

where P_0 is the potential energy when $y = 0$. In general, 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

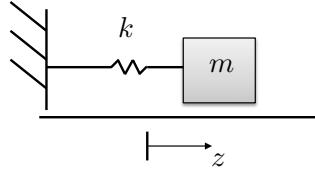


Figure 3.2: The potential energy of spring stretched by distance z is $P = \frac{1}{2}kz^2$.

of 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.

3.1.2 Generalized Forces

The last thing we need to specify before deriving the equations of motion are the so called *generalized forces* acting along each generalized coordinate. The *generalized coordinates* are the minimum set of configuration variables. For example, for the mass-spring-damper system shown in Figure 2.4 the generalized coordinates are $\mathbf{q} = (z_1, z_2)^\top$, for the spinning dumbbell system shown in Figure 2.5, and the single link robot arm shown in Figure 2.6 the generalized coordinate is $\mathbf{q} = \theta$, for the pendulum on a cart system shown in Figure 2.8 the generalized coordinates are $\mathbf{q} = (z, \theta)^\top$, and for the satellite system shown in Figure 3.5 the generalized coordinates are $\mathbf{q} = (\theta, \phi)^\top$.

The generalized forces are the nonconservative forces and torques are the nonconservative forces and torques. For this class we will consider three sources:

- Externally applied forces and torques,
- Friction forces and torques,
- Damping forces and torques.

For example, for the mass-spring-damper system shown in Figure 2.4, the generalized force acting along z_1 is due to the dampers and is given by $\tau_1 = -b_1\dot{z}_1 - b_2(\dot{z}_1 - \dot{z}_2)$, and the generalized force acting along z_2 is due to the

applied force F and the second damper and is given by $\tau_2 = F - b_2(\dot{z}_2 - \dot{z}_1)$. Therefore the generalized force is

$$\boldsymbol{\tau} = \begin{pmatrix} -b_1\dot{z}_1 - b_2(\dot{z}_1 - \dot{z}_2) \\ F - b_2(\dot{z}_2 - \dot{z}_1) \end{pmatrix}.$$

3.1.3 Euler Lagrange Equations

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

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

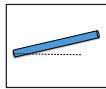
where $K(\mathbf{q}, \dot{\mathbf{q}})$ is the kinetic energy of the system and is, in general, a function of the generalized coordinates and the generalized velocities, and $P(\mathbf{q})$ is the potential energy which is, in general, a function of the generalized coordinates. As shown in Equations (1.6)–(??), the Euler-Lagrange equations are given by

$$\begin{aligned} \frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_1} \right) - \frac{\partial L(\mathbf{q})}{\partial q_1} &= -b_1 \dot{q}_1 + \tau_1 \\ \frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_2} \right) - \frac{\partial L(\mathbf{q})}{\partial q_2} &= -b_2 \dot{q}_2 + \tau_2 \\ &\vdots \\ \frac{d}{dt} \left(\frac{\partial L(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_n} \right) - \frac{\partial L(\mathbf{q})}{\partial q_n} &= -b_n \dot{q}_n + \tau_n, \end{aligned}$$

or in vector form as

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

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.
- (d) Derive the equations of motion using the Euler-Lagrange equations.

Solution

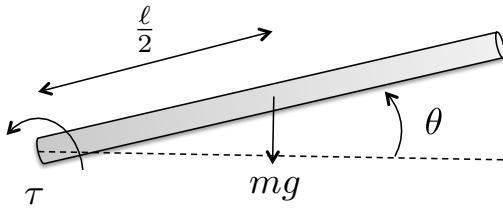


Figure 3.3: Energy calculation for the single link robot arm

Let P_0 be the potential when $\theta = 0$. Then the potential energy is given by

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

The generalized coordinate is

$$q_1 = \theta.$$

The generalized force is

$$\tau_1 = \tau - b\dot{\theta}.$$

From Homework 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

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

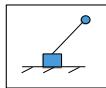
where

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

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.
- (d) Derive the equations of motion using the Euler-Lagrange equations.

Solution

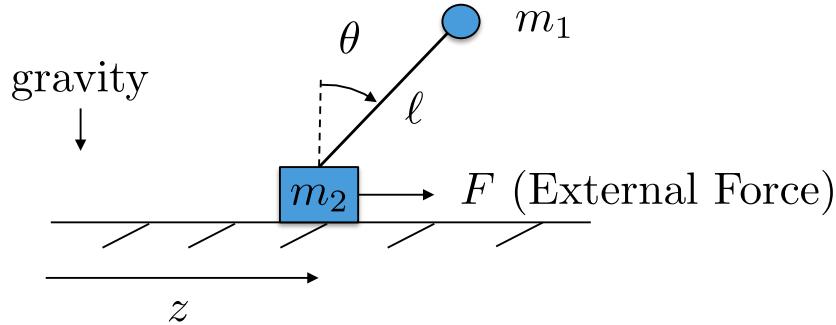


Figure 3.4: Pendulum on a cart.

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

$$P = P_0 - m_1 g \ell (1 - \cos \theta).$$

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 on the cart are $\tau_1 = F$, and the external torque acting on the pendulum is $\tau_2 = 0$. Therefore, the generalized forces are $\boldsymbol{\tau} = (F, 0)^\top$.

Using Equation (2.1), the kinetic energy can be written in terms of the generalized coordinates as

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

and the potential energy can be written as

$$\begin{aligned} P(\mathbf{q}) &= P_0 - m_1g\ell(1 - \cos\theta) \\ &= P_0 - m_1g\ell(1 - \cos q_2). \end{aligned}$$

The Lagrangian is therefore given by

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

Therefore

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

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

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

Therefore the Euler-Lagrange equation

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

gives

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

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

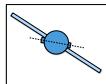
$$\begin{pmatrix} (m_1 + m_2)\ddot{z} + m_1\ell\ddot{\theta}\cos\theta \\ m_1\ell^2\ddot{\theta} + m_1\ell\ddot{z}\cos\theta \end{pmatrix} = \begin{pmatrix} m_1\ell\dot{\theta}^2\sin\theta - b\dot{z} + F \\ m_1g\ell\sin\theta \end{pmatrix}.$$

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

$$\begin{pmatrix} (m_1 + m_2) & m_1\ell \cos \theta \\ m_1\ell \cos \theta & m_1\ell^2 \end{pmatrix} \begin{pmatrix} \ddot{z} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} m_1\ell\dot{\theta}^2 \sin \theta - b\dot{z} + F \\ m_1g\ell \sin \theta \end{pmatrix}. \quad (3.2)$$

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.
- (d) Derive the equations of motion using the Euler-Lagrange equations.

Solution

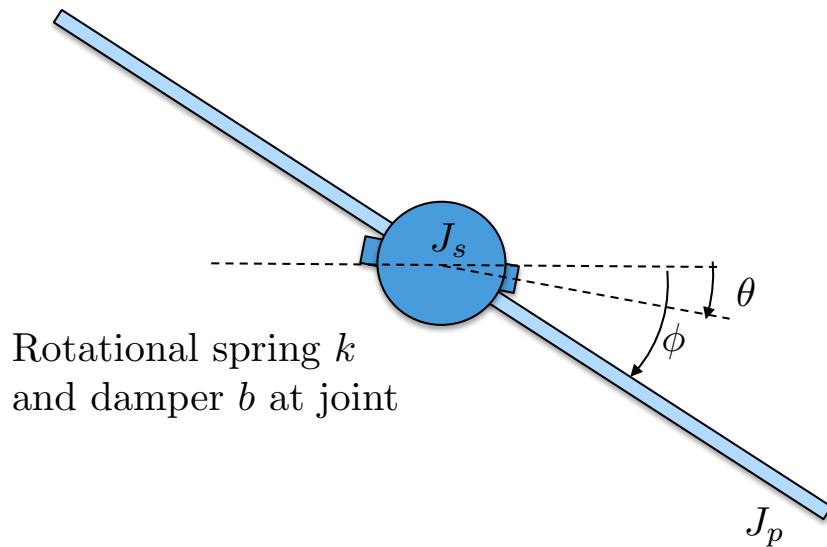


Figure 3.5: Satellite with flexible solar panels.

The generalized coordinates for the satellite attitude control problem are the angles θ and ψ . Therefore, we define the generalized coordinates as $\mathbf{q} = (\theta, \psi)^\top$. We will assume the ability to apply torque on the main body 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 friction term is given by

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

Using Equation (2.2), 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

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

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

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

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

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

Therefore the Euler-Lagrange equation

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

gives

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

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

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

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

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

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

Notes and References

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 (two time derivatives).

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 feedback 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 ?? 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.

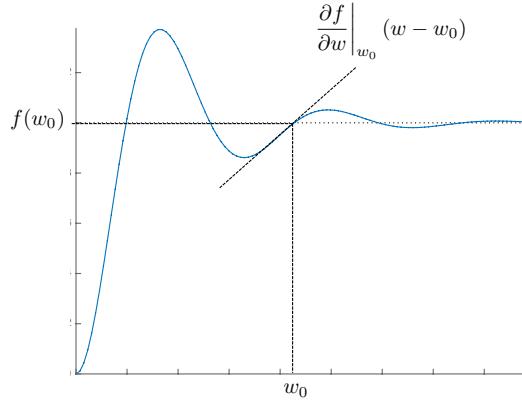
4.1.1 Jacobian Linearization

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

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

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

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

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

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

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 equations of a second order system are given by

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

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

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

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

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

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 $u_e = by_e + g(y_e, 0)$.

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

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

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

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

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

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

Using the equilibrium $u_e = by_e + g(y_e, 0)$ 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.

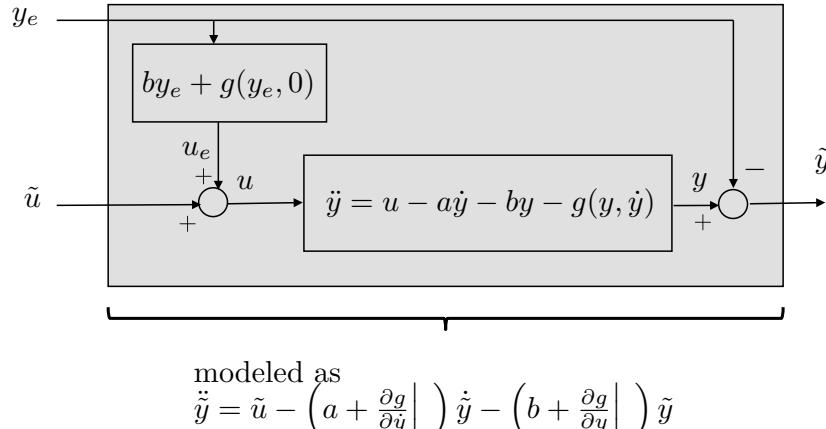


Figure 4.2: Linear model using Jacobian linearization

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

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

then the resulting equations of motion are

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

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.

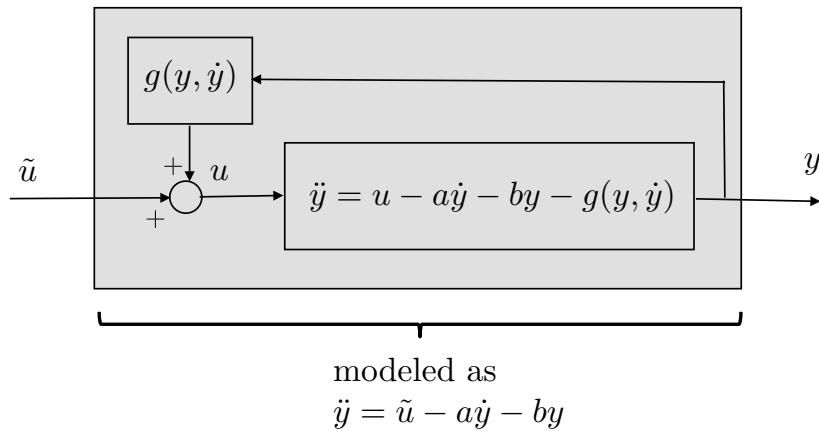
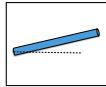


Figure 4.3: Linear model using feedback linearization

4.2 Design Study A. Single Link Robot Arm



Homework Problem A.5

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.

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

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

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 $\tau_e = \frac{mg\ell}{2}\cos\theta_e$.

The equilibria values (θ_e, τ_e) are found by setting $\dot{\theta} = \ddot{\theta} = 0$ in Equation (4.3) to obtain

$$\tau_e = mg\frac{\ell}{2}\cos\theta_e. \quad (4.5)$$

Therefore, any pair (θ_e, τ_e) satisfying Equation (4.5) 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 $\tilde{\theta} \triangleq \theta - \theta_e$, $\dot{\tilde{\theta}} \triangleq \dot{\theta} - \dot{\theta}_e = \dot{\theta}$, $\ddot{\tilde{\theta}} = \ddot{\theta} - \ddot{\theta}_e = \ddot{\theta}$, and $\tilde{\tau} = \tau - \tau_e$, each term in Equation (4.3) can be expanded about the equilibrium as follows:

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

Substituting into Equation (4.3) gives

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

Simplifying this expression using Equation (4.5) gives

$$\frac{m\ell^2}{3} \ddot{\tilde{\theta}} - \frac{mg\ell}{2} (\sin \theta_e) \dot{\tilde{\theta}} = \tilde{\tau} - b \dot{\tilde{\theta}}, \quad (4.6)$$

which are the linearized equations of motion using Jacobian linearization.

Feedback linearization proceeds by using the feedback linearizing control

$$\tau = \frac{mg\ell}{2} \cos \theta + \tilde{\tau} \quad (4.7)$$

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

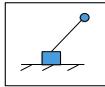
$$\frac{m\ell^2}{3} \ddot{\tilde{\theta}} = \tilde{\tau} - b \dot{\tilde{\theta}}, \quad (4.8)$$

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 (4.7) to the control signal using Jacobian linearization which is

$$\tau = \tau_e + \tilde{\tau} = \frac{mg\ell}{2} \cos \theta_e + \tilde{\tau}. \quad (4.9)$$

The control signal in (4.9) uses the equilibrium angle θ_e whereas the control signal in (4.7) 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.5

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 equation describing the inverted derived in HW B.3 is

$$\begin{aligned} (m_1 + m_2)\ddot{z} + m_1\ell\ddot{\theta}\cos\theta &= m_1\ell\dot{\theta}^2\sin\theta - b\dot{z} + F \\ m_1\ell\ddot{z}\cos\theta + m_1\ell^2\ddot{\theta} &= m_1g\ell\sin\theta. \end{aligned} \quad (4.10)$$

The equilibria values (z_e, θ_e, F_e) are found by setting $\dot{z} = \ddot{z} = \dot{\theta} = \ddot{\theta} = 0$ in Equation (4.10) to obtain

$$F_e = 0 \quad (4.11)$$

$$m_1g\ell\sin\theta_e = 0. \quad (4.12)$$

Therefore, any triple (z_e, θ_e, F_e) satisfying Equation (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.

To linearize around (z_e, θ_e, F_e) note that

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

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 (4.10) in its linearized form as

$$\begin{aligned}(m_1 + m_2)[\ddot{z}_e + \ddot{\tilde{z}}] + m_1\ell[\ddot{\tilde{\theta}}] &= m_1\ell[0] - b[\dot{z}_e + \dot{\tilde{z}}] + [F_e + \tilde{F}] \\ m_1\ell[\ddot{\tilde{z}}] + m_1\ell^2[\ddot{\theta}_e + \ddot{\tilde{\theta}}] &= m_1g\ell[\tilde{\theta}],\end{aligned}$$

which simplifies to

$$\begin{pmatrix} (m_1 + m_2) & m_1\ell \\ m_1\ell & m_1\ell^2 \end{pmatrix} \begin{pmatrix} \ddot{\tilde{z}} \\ \ddot{\tilde{\theta}} \end{pmatrix} = \begin{pmatrix} -b\dot{\tilde{z}} + \tilde{F} \\ m_1g\ell\tilde{\theta} \end{pmatrix}, \quad (4.13)$$

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

RWB: Need references to additional material on feedback linearization. Say something about generalizing the results where the nonlinearities are in the input channel. In particular, feedback linearization can still be used if there is a change of variables that put all nonlinearities into the feedback channel. More generally, similar ideas can be applied using backstepping. Add some references.

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

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

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

$$\begin{aligned}s^n Y(s) + a_{n-1}s^{n-1}Y(s) + \cdots + a_1sY(s) + a_0Y(s) \\ = b_m s^m U(s) + b_{m-1}s^{m-1}U(s) + \cdots + b_1sU(s) + b_0U(s).\end{aligned}\quad (5.1)$$

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

$$(s^n + a_{n-1}s^{n-1} + \cdots + a_1s + a_0) Y(s) = () U(s).$$

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

$$b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0 = 0.$$

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

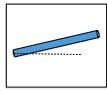
$$Y(s) = \frac{5s + 6}{s^3 + 2s^2 + 3s + 4}U(s).$$

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.6

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 (4.8) as

$$\frac{m\ell^2}{3}\ddot{\theta} = \tilde{\tau} - b\dot{\theta}. \quad (5.2)$$

Taking the Laplace transform of Equation (5.2) 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

$$\Theta(s) = \left(\frac{\frac{3}{m\ell^2}}{s^2 + \frac{3b}{m\ell^2}s} \right) \tilde{\tau}(s), \quad (5.3)$$

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 (4.7). The block diagram associated with Equation (5.3) is shown in Figure 5.1

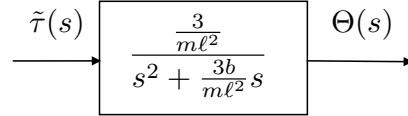


Figure 5.1: A block diagram of the single link robot arm.

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 the system is given by the coupled differential equations

$$\begin{aligned} \ddot{y}_1 + a_{11}\dot{y}_1 + \ddot{y}_2 + a_{12}\dot{y}_2 &= b_{11}u_1 + b_{12}u_2 + b_{13}\dot{u}_3 \\ \ddot{y}_2 + a_{22}y_2 &= b_{21}u_2. \end{aligned}$$

Taking the Laplace transform and arranging in matrix notation gives

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

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

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

we get

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

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

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 interested 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

$$\begin{aligned}(s^2 + 0.1s)Y_1(s) - 2Y_2(s) &= U(s) \\ (s^2 + 20s + 200)Y_2(s) - 4sY_1(s) &= 3U(s).\end{aligned}\quad (5.4)$$

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

$$\begin{aligned}\begin{pmatrix} Y_1(s) \\ Y_2(s) \end{pmatrix} &= \frac{\left(\begin{array}{c|c} s^2 + 20s + 200 & 2 \\ \hline 4s & s^2 + 0.1s \end{array} \right)}{(s^2 + 0.1s)(s^2 + 20s + 200) - 8s} \begin{pmatrix} 1 \\ 3 \end{pmatrix} U(s) \\ &= \begin{pmatrix} \frac{s^2+20s+206}{s^4+20.1s^3+202s^2+12s} \\ \frac{3s^2+4.3s}{s^4+20.1s^3+202s^2+12s} \end{pmatrix} U(s).\end{aligned}\quad (5.5)$$

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 simply model the system as a cascade of fast and slow subsystems. Note that the poles of the transfer functions in Equation (5.5) 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 5.4 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

$$Y_1(s) = \frac{1}{s^2 + 0.1s}U(s) + \frac{2}{s^2 + 0.1s}Y_2(s) \quad (5.6)$$

$$Y_2(s) = \frac{3}{s^2 + 20s + 200}U(s) + \frac{4s}{s^2 + 20s + 200}Y_1(s), \quad (5.7)$$

where Equation (5.6) with poles at 0 and -0.1 approximate the slow subsystem, and Equation (5.7) 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.

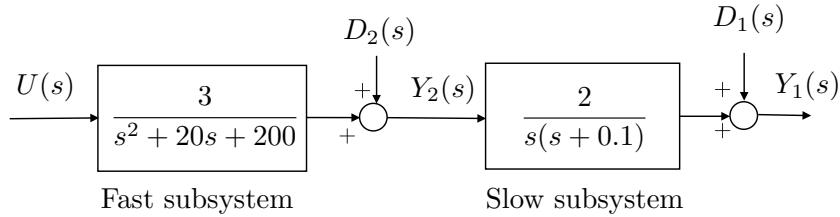
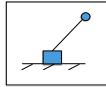


Figure 5.2: System approximated by cascade of fast and slow subsystems.

5.4 Design Study B. Inverted Pendulum



Homework Problem B.6

- (a) Starting with the linearized equations for the inverted pendulum, use the Laplace transform to convert the equations of motion to the s-domain.
- (b) Find the full transfer matrix from the input $\tilde{F}(s)$ to the outputs $\tilde{Z}(s)$ and $\tilde{\Theta}(s)$. 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 B.5, the linearized equations of motion are given by

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

Inverting the matrix on the left and side gives

$$\begin{aligned} \begin{pmatrix} \ddot{\tilde{z}} \\ \ddot{\tilde{\theta}} \end{pmatrix} &= \begin{pmatrix} (m_1 + m_2) & m_1\ell \\ m_1\ell & m_1\ell^2 \end{pmatrix}^{-1} \begin{pmatrix} -b\dot{\tilde{z}} + \tilde{F} \\ m_1g\ell\tilde{\theta} \end{pmatrix} \\ &= \frac{\begin{pmatrix} m_1\ell^2 & -m_1\ell \\ -m_1\ell & (m_1 + m_2) \end{pmatrix}}{m_1m_2\ell^2} \begin{pmatrix} -b\dot{\tilde{z}} + \tilde{F} \\ m_1g\ell\tilde{\theta} \end{pmatrix} \\ &= \begin{pmatrix} -\frac{b}{m_2}\dot{\tilde{z}} + \frac{1}{m_2}\tilde{F} - \frac{m_1g}{m_2}\tilde{\theta} \\ \frac{b}{m_2\ell}\dot{\tilde{z}} - \frac{1}{m_2\ell}\tilde{F} + \frac{(m_1+m_2)g}{m_2\ell}\tilde{\theta} \end{pmatrix}, \end{aligned}$$

or in other words, the coupled differential equations

$$\begin{aligned} \ddot{\tilde{z}} + \frac{b}{m_2}\dot{\tilde{z}} &= \frac{1}{m_2}\tilde{F} - \frac{m_1g}{m_2}\tilde{\theta} \\ \ddot{\tilde{\theta}} - \frac{(m_1+m_2)g}{m_2\ell}\tilde{\theta} &= \frac{b}{m_2\ell}\dot{\tilde{z}} - \frac{1}{m_2\ell}\tilde{F}. \end{aligned}$$

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

$$(s^2 + \frac{b}{m_2}s)\tilde{Z}(s) = \frac{1}{m_2}\tilde{F}(s) - \frac{m_1g}{m_2}\tilde{\Theta}(s) \quad (5.8)$$

$$(s^2 - \frac{(m_1+m_2)g}{m_2\ell})\tilde{\Theta}(s) = \frac{b}{m_2\ell}s\tilde{Z}(s) - \frac{1}{m_2\ell}\tilde{F}(s). \quad (5.9)$$

To find the transfer matrix from \tilde{F} to $(\tilde{Z}, \tilde{\Theta})^\top$, write Equation (5.8) and (5.9) in matrix form as

$$\left(\begin{array}{c|c} s^2 + \frac{b}{m_2}s & \frac{m_1g}{m_2} \\ \hline -\frac{b}{m_2}s & s^2 - \frac{(m_1+m_2)g}{m_2\ell} \end{array} \right) \begin{pmatrix} \tilde{Z}(s) \\ \tilde{\Theta}(s) \end{pmatrix} = \begin{pmatrix} \frac{1}{m_2} \\ -\frac{1}{m_2\ell} \end{pmatrix} \tilde{F}(s),$$

and invert the matrix on the left hand side to obtain

$$\begin{pmatrix} \tilde{Z}(s) \\ \tilde{\Theta}(s) \end{pmatrix} = \begin{pmatrix} \frac{\frac{1}{m_2}s^2 - \frac{g}{m_2\ell}}{s^4 + \frac{b}{m_2}s^3 - \frac{(m_1+m_2)g}{m_2\ell}s^2 - \frac{bg}{m_2\ell}s} \\ \frac{-\frac{1}{m_2\ell}s^2}{s^4 + \frac{b}{m_2}s^3 - \frac{(m_1+m_2)g}{m_2\ell}s^2 - \frac{bg}{m_2\ell}s} \end{pmatrix} \tilde{F}(s).$$

Plugging in the nominal values from Section B we get that there are two fast poles at ± 4.95 , and two slow poles at 0 and -0.04 . 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 (5.8) and (5.9) and divide by the polynomial on the left hand side to obtain

$$\tilde{Z}(s) = \frac{\frac{1}{m_2}}{s(s + \frac{b}{m_2})} \tilde{F}(s) + \frac{-\frac{m_1g}{m_2}}{s(s + \frac{b}{m_2})} \tilde{\Theta}(s) \quad (5.10)$$

$$\tilde{\Theta}(s) = \frac{\frac{b}{m_2\ell}s}{(s^2 - \frac{(m_1+m_2)g}{m_2\ell})} \tilde{Z}(s) + \frac{-\frac{1}{m_2\ell}}{(s^2 - \frac{(m_1+m_2)g}{m_2\ell})} \tilde{F}(s). \quad (5.11)$$

Note that the roots of the polynomial $s(s + \frac{b}{m_2}) = 0$ are at 0 and -0.05 , and the roots of the polynomial $s^2 - \frac{(m_1+m_2)g}{m_2\ell} = 0$ are at ± 4.95 . Therefore, Equation (5.10) approximate the slow dynamics and (5.11) approximate the fast dynamics. Therefore, we let \tilde{F} drive the fast dynamics with output $\tilde{\Theta}$ and we let $\tilde{\Theta}$ drive the slow dynamics with output \tilde{Z} to get

$$\begin{aligned} \tilde{Z}(s) &= \frac{-\frac{m_1g}{m_2}}{s(s + \frac{b}{m_2})} \tilde{\Theta}(s) + D_1(s) \\ \tilde{\Theta}(s) &= \frac{-\frac{1}{m_2\ell}}{(s^2 - \frac{(m_1+m_2)g}{m_2\ell})} \tilde{F}(s) + D_2(s) \end{aligned}$$

where we assume that d_1 and d_2 are unknown disturbances. The block diagram for the approximate system is shown in Figure 5.3

The cascade approximation makes since physically because the force on the cart has an almost immediate effect on the angle of the rod. On the other hand, if the angle of the rod is a nonzero constant then the position of the cart must be moving. Therefore, the desired angle of the rod will dictate a desired velocity for the cart.

The PID and loopshaping techniques that we will discuss in Parts III and V will use transfer function models of the plant. However, the state

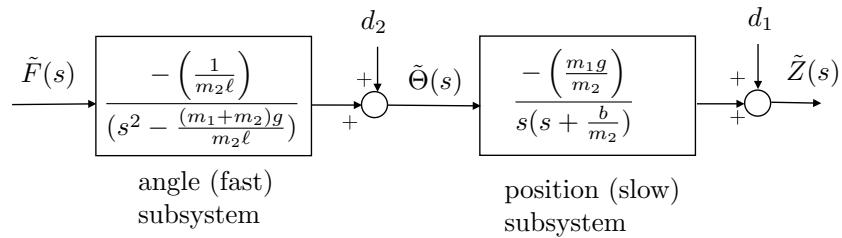
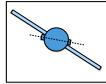


Figure 5.3: The inverted pendulum dynamics are approximated by cascade of fast and slow subsystems. The fast subsystem is the transfer function from the force to the angle, and the slow subsystem is the transfer function from the angle to the position.

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.6

- (a) Starting with the linearized equations for the satellite attitude problem, 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 $\Psi(s)$ 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

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

or in other words, the coupled differential equations

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

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

$$(s^2 + \frac{b}{J_s}s + \frac{k}{J_s})\Theta(s) = (\frac{b}{J_s}s + \frac{k}{J_s})\Psi(s) + \frac{1}{J_s}\tau(s) \quad (5.12)$$

$$(s^2 + \frac{b}{J_p}s + \frac{k}{J_p})\Psi(s) = (\frac{b}{J_p}s + \frac{k}{J_p})\Theta(s). \quad (5.13)$$

To find the transfer matrix from τ to $(\Theta, \Phi)^\top$, write Equation (5.12) and (5.13) 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

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

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 (5.12) and (5.13) and divide by the polynomials on the left hand side to obtain

$$\Theta(s) = \frac{\frac{b}{J_s}s + \frac{k}{J_s}}{s^2 + \frac{b}{J_s}s + \frac{k}{J_s}} \Psi(s) + \frac{\frac{1}{J_s}}{s^2 + \frac{b}{J_s}s + \frac{k}{J_s}} \tau(s) \quad (5.14)$$

$$\Psi(s) = \frac{\frac{b}{J_p}s + \frac{k}{J_p}}{s^2 + \frac{b}{J_p}s + \frac{k}{J_p}} \Theta(s). \quad (5.15)$$

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 (5.14) approximate the slow dynamics and (5.15) 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 Ψ to get

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

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

where we assume that $d(t)$ is an unknown disturbance signal. The block diagram for the approximate system is shown in Figure 5.4

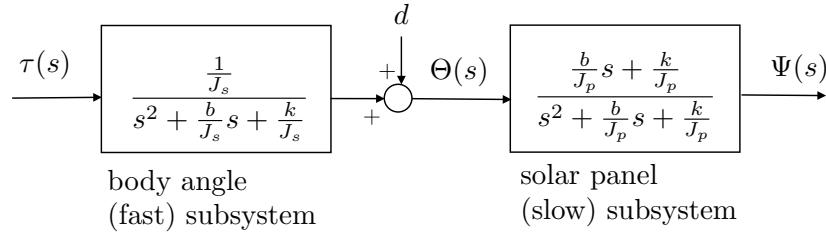


Figure 5.4: The satellite attitude dynamics are approximated by cascade of fast and slow subsystems. The fast subsystem is the transfer function from the torque to the angle θ , and the slow subsystem is the transfer function from the angle θ to the angle ψ .

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 ψ through the spring damper system.

Notes and References

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 first order nonlinear differential equation. The general state space equations for a discrete time system are written as

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

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

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

Any system of 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

$$\begin{aligned} \dot{x} &= \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} \dot{y} \\ \ddot{y} \\ \dddot{y} \end{pmatrix} = \begin{pmatrix} \dot{y} \\ \ddot{y} \\ -y^2\dot{y} - \sin(y\dot{y}) - e^y + u \end{pmatrix} \\ &= \begin{pmatrix} x_2 \\ x_3 \\ -x_1^2x_3 - \sin(x_1x_2) - e^{x_1} + u \end{pmatrix} \triangleq f(x, u), \end{aligned}$$

and the output equation is

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

As another example, consider the coupled differential equation

$$\begin{aligned} \ddot{y}_1 + \dot{y}_1y_2 + y_2 &= u \\ \ddot{y}_2 + \dot{y}_2 \cos(y_1) + \dot{y}_1 &= 0. \end{aligned}$$

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

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

$$y = h(x, u) \tag{6.6}$$

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

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

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

where the Jacobian matrices are defined as

$$\begin{aligned}\frac{\partial f}{\partial x} &\triangleq \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \\ \frac{\partial f}{\partial u} &\triangleq \begin{pmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{pmatrix} \\ \frac{\partial h}{\partial x} &\triangleq \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial h_p}{\partial x_1} & \cdots & \frac{\partial h_p}{\partial x_n} \end{pmatrix} \\ \frac{\partial h}{\partial u} &\triangleq \begin{pmatrix} \frac{\partial h_1}{\partial u_1} & \cdots & \frac{\partial h_1}{\partial u_m} \\ \vdots & & \vdots \\ \frac{\partial h_p}{\partial u_1} & \cdots & \frac{\partial h_p}{\partial u_m} \end{pmatrix}.\end{aligned}$$

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 $A = \frac{\partial f}{\partial x}|_e$ and $B = \frac{\partial f}{\partial u}|_e$, 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), 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

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

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

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

then the transfer function from u to y is given by

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

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 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

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

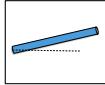
and that the poles of $H(s)$ are given by the roots of the polynomial

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

Recall from linear algebra 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 14 and 16 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.7

For the feedback linearized equations of motion for the single link robot arm given in Equation (4.8), 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 (4.8) and solving for $\ddot{\theta}$ gives

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

Therefore,

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

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

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

Therefore, the linearized state space equations are given by

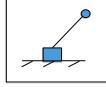
$$\begin{aligned}\dot{\tilde{x}} &= \begin{pmatrix} 0 & 1 \\ \frac{3g \sin \theta_e}{2\ell} & -\frac{3b}{2\ell} \end{pmatrix} \tilde{x} + \begin{pmatrix} 0 \\ \frac{3}{m\ell^2} \end{pmatrix} \tilde{u} \\ \tilde{y} &= (1 \ 0) \tilde{x}.\end{aligned}\tag{6.16}$$

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 (5.3).

6.3 Design Study B. Inverted Pendulum



Homework Problem B.7

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

$$\begin{aligned}\begin{pmatrix} \ddot{\tilde{z}} \\ \ddot{\tilde{\theta}} \end{pmatrix} &= \frac{\begin{pmatrix} m_1\ell^2 & -m_1\ell \\ -m_1\ell & (m_1+m_2) \end{pmatrix}}{m_1^2\ell + m_1m_2\ell^2 - m_1^2\ell^2} \begin{pmatrix} -b\dot{\tilde{z}} + \tilde{f} \\ m_1g\ell\tilde{\theta} \end{pmatrix} \\ &= \begin{pmatrix} -\frac{b}{m_2}\dot{\tilde{z}} + \frac{1}{m_2}\tilde{F} - \frac{m_1g}{m_2}\tilde{\theta} \\ \frac{b}{m_2\ell}\dot{\tilde{z}} - \frac{1}{m_2\ell}\tilde{F} + \frac{(m_1+m_2)g}{m_2\ell}\tilde{\theta} \end{pmatrix}.\end{aligned}$$

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

$$\begin{aligned}\dot{\tilde{x}} &\triangleq \begin{pmatrix} \dot{\tilde{x}}_1 \\ \dot{\tilde{x}}_2 \\ \dot{\tilde{x}}_3 \\ \dot{\tilde{x}}_4 \end{pmatrix} = \begin{pmatrix} \dot{\tilde{z}} \\ \dot{\tilde{\theta}} \\ \dot{\tilde{z}} \\ \dot{\tilde{\theta}} \end{pmatrix} = \begin{pmatrix} \tilde{x}_3 \\ \tilde{x}_4 \\ -\frac{b}{m_2}\tilde{x}_3 + \frac{1}{m_2}\tilde{u} - \frac{m_1g}{m_2}\tilde{x}_2 \\ \frac{b}{m_2\ell}\tilde{x}_3 - \frac{1}{m_2\ell}\tilde{u} + \frac{(m_1+m_2)g}{m_2\ell}\tilde{x}_2 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{m_1g}{m_2} & -\frac{b}{m_2} & 0 \\ 0 & \frac{(m_1+m_2)g}{m_2\ell} & \frac{b}{m_2\ell} & 0 \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{m_2} \\ -\frac{1}{m_2\ell} \end{pmatrix} \tilde{u}.\end{aligned}$$

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

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

Therefore, the linearized state space equations are given by

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

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

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

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

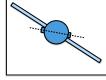
$$\begin{aligned} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} &= \begin{pmatrix} x_3 \\ x_4 \\ \frac{m_1^2 \ell^2 x_4^2 \sin x_2 - b m_1 \ell^2 x_3 + m_1 \ell^2 u - m_1^2 \ell^2 g \sin x_2 \cos x_2}{m_1 m_2 \ell^2 + m_1^2 \ell^2 \sin^2 x_2} \\ \frac{-m_1^2 \ell^2 x_4^2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

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

$$\begin{aligned}
 A &\stackrel{\triangle}{=} \left. \frac{\partial f}{\partial x} \right|_e = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{\partial f_3}{\partial x_2} & \frac{-bm_1\ell^2}{m_1m_2\ell^2+m_1\ell^2\sin^2 x_2} & \frac{2m_1^2\ell x_4 \sin x_2}{m_1m_2\ell^2+m_1\ell^2\sin^2 x_2} \\ 0 & \frac{\partial f_4}{\partial x_2} & \frac{-bm_1\ell \cos x_2}{m_1m_2\ell^2+m_1^2\ell^2\sin^2 x_2} & \frac{-2m_1^2\ell^2 x_4 \sin x_2 \cos x_2}{m_1m_2\ell^2+m_1^2\ell^2\sin^2 x_2} \end{pmatrix} \Bigg|_e = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-m_1g}{m_2} & \frac{-b}{m_2} & 0 \\ 0 & \frac{(m_1+m_2)g}{m_2\ell} & \frac{b}{m_2\ell} & 0 \end{pmatrix} \\
 B &\stackrel{\triangle}{=} \left. \frac{\partial f}{\partial u} \right|_e = \begin{pmatrix} 0 \\ 0 \\ \frac{m_1\ell^2}{m_1m_2\ell^2+m_1^2\ell^2\sin^2 x_2} \\ -\frac{m_1\ell \cos \theta}{m_1m_2\ell^2+m_1^2\ell^2\sin^2 x_2} \end{pmatrix} \Bigg|_e = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{m_2} \\ -\frac{1}{m_2\ell} \end{pmatrix} \\
 C &\stackrel{\triangle}{=} \left. \frac{\partial h}{\partial x} \right|_e = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \\
 D &\stackrel{\triangle}{=} \left. \frac{\partial h}{\partial u} \right|_e = \begin{pmatrix} 0 \\ 0 \end{pmatrix},
 \end{aligned}$$

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

6.4 Design Study C. Satellite Attitude Control



Homework Problem C.7

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

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

Similarly

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

Therefore, the state space model is

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

where

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k}{J_s} & \frac{k}{J_s} & -\frac{b}{J_s} & \frac{b}{J_s} \\ \frac{k}{J_p} & -\frac{k}{J_p} & \frac{b}{J_p} & -\frac{b}{J_p} \end{pmatrix}$$

$$B = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{J_s} \\ 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Notes and References

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 intuitively 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 the following parts on observer based control and loopshaping, we will extend the basic ideas of this part to higher order systems.

Figure 6.1 shows the basic idea for proportional control. The output y of the physical system is subtracted from the commanded output y^c to produce the error e . Proportional control determine 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.

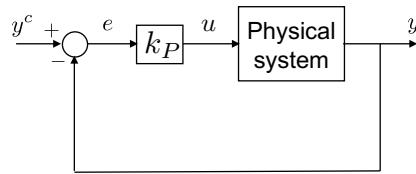


Figure 6.1: Proportional Control. The feedback control is a gain k_p multiplied by the error.

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.

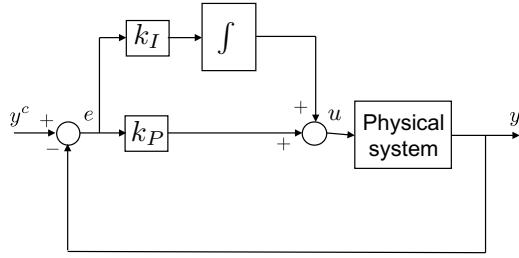


Figure 6.2: Proportional plus Integral (PI) Control. The feedback control is a linear combination of proportional and integral control.

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.

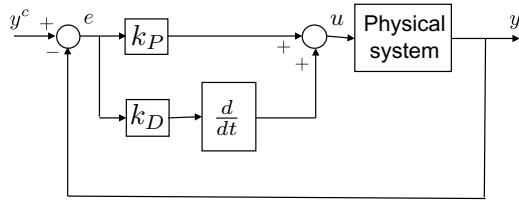


Figure 6.3: Proportional plus Derivative (PD) Control. The feedback control is a linear combination of proportional and derivative control.

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.

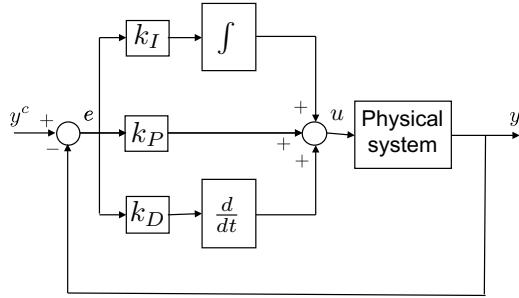


Figure 6.4: Proportional plus Integral plus Derivative (PID) Control. The feedback control is a linear combination of proportional and integral and derivative control.

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 commanded output $y^c(t)$. The proportional control term will tend to push the system so as to reduce the gap between $y(t)$ and $y^c(t)$. As $y(t)$ begins to change, and very 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.

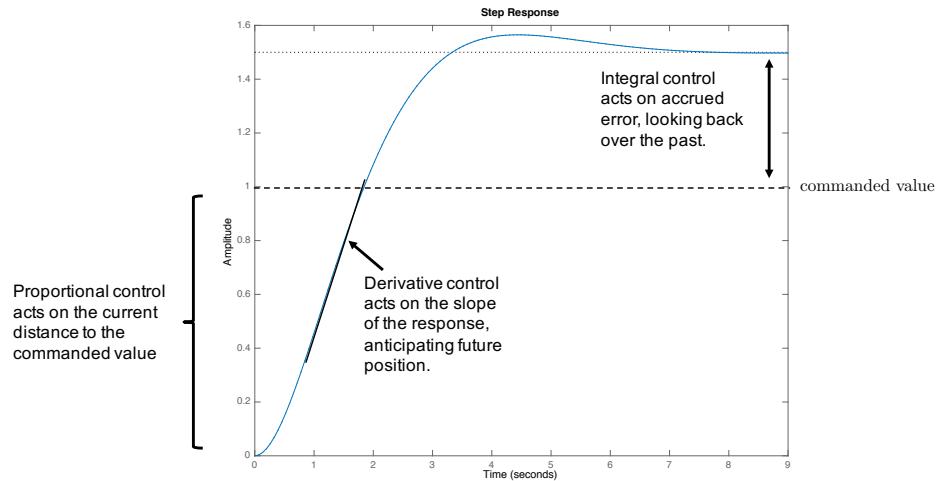


Figure 6.5: Intuitive interpretation of PID control. Proportional control acts on the current error, integral control acts on accrued past error, and derivative control acts on anticipated future error.

This part will be 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 will introduce several design specifications and show how those relate to pole locations for the special case of second order systems. We will also show how pole locations can be selected to achieve those specifications. The performance of physical systems is typically limited by how much force and torque can be applied to the system given realistic constraints on motors and other actuation devices. In Chapter 10 we show how saturation constraints on the system inputs, effectively limits the performance of the system. We also show how to design PD controllers to maximize the performance of second order systems given those constraints. In Chapter 11 we begin our study of integrators and their effect on the closed loop system. We introduce the notion of *system type* in the context of reference tracking and disturbance rejection. In Chapter 12 we introduce the root locus design tool and show how it can be used to select the integrator gain. Many complex mechanical systems that are not second order, can be modeled as a cascade of second order systems. This leads to a design technique called successive loop closure, and in Chapter 9 we describe this method, and its implications. Finally, in Chapter 13 we show how to write computer code to implement PID controllers.

Chapter 7

Pole Placement for Second Order Systems

7.1 Theory

Suppose that we have modeled a physical system using 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 given by the roots of the *open loop characteristic polynomial*

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

which are given by

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

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^c , then the block diagram is shown in Figure 7.1.

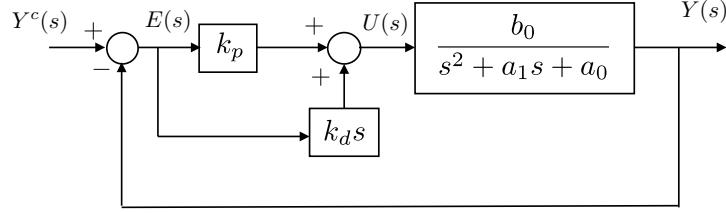


Figure 7.1: PD control of a second order system.

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

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

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

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

Note that for the closed loop system, we can precisely specify the closed loop poles 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

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

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

$$k_P = \frac{\alpha_0 - a_0}{b_0}$$

$$k_D = \frac{\alpha_1 - a_1}{b_0}.$$

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

$$z_{\text{closed loop}} = -\frac{k_P}{k_D}.$$

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.

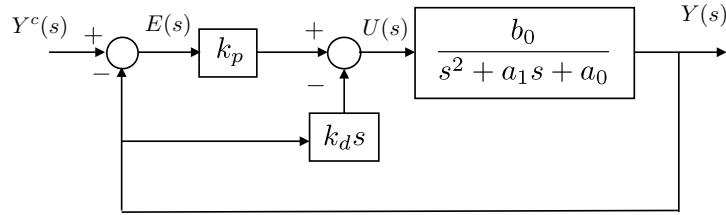


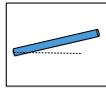
Figure 7.2: PD control of a second order system, where the derivative control only differentiates the output y and not the error e .

In this case the closed loop transfer function is calculated as

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

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 $y^c(t)$ 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 $e(t) = y^c(t) - y(t)$.

7.2 Design Study A. Single Link Robot Arm



Homework Problem A.8

- (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 PD control architecture shown in Figure 7.2, find the closed loop transfer function from θ^c 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 $p_2 = -4$.

Solution

The open loop transfer function found in problem A.6 is

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

Using $\theta_e = 0$ and substituting parameters from Section ?? gives

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

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

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

which are given by

$$p_{\text{open loop}} = 0, -0.667.$$

Using PD control, the closed loop system is therefore shown in Figure 7.3.

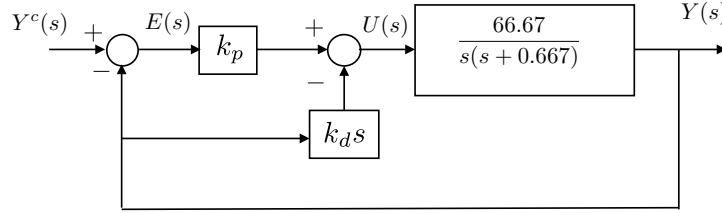


Figure 7.3: PD control of the single link robot arm.

The transfer function of the closed loop system is given by

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

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

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

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

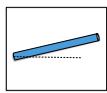
$$0.667 + 66.67k_D = 7$$

$$66.67k_P = 12.$$

Solving for k_P and k_D gives

$$k_P = 0.18$$

$$k_D = 0.095.$$



Homework Problem A.9

Using the gains found in Problem A.8, 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 7.4. Recall that the torque in the transfer function model is

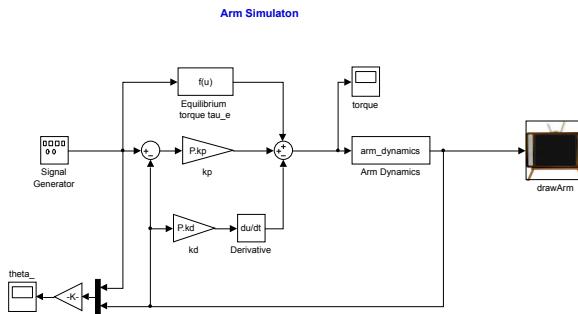


Figure 7.4: Simulink file for Homework A.9

the linearized torque $\tilde{\tau} = \tau - \tau_e$. Therefore, the torque applied to the robot arm is $\tau = \tau_e + \tilde{\tau}$. The equilibrium torque must be added to the torque computed by the PID controller as shown in Figure 7.4.

Solution

For a solution to this problem, see the wiki associated with this book.

Notes and References

Chapter 8

Time Domain Design Specifications

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 derive the relationship between two pole locations and the step response. In Section 8.1.3 we look at how pole locations can be selected to achieve certain time domain specifications, specifically *rise time*, *settling time*, and *maximum percent overshoot*. Finally in Section 8.1.4 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

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

Taking the inverse Laplace transform gives

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

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

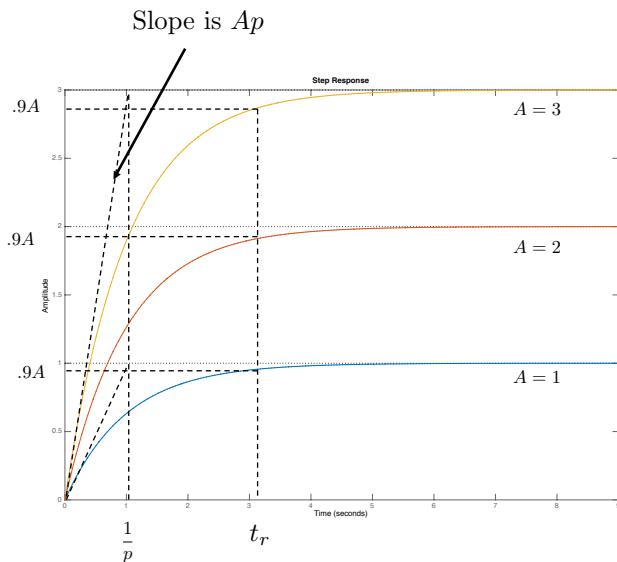


Figure 8.1: Step response of a first order system with pole p . Three step responses are shown, where the step size is $A = 1$, $A = 2$, and $A = 3$, respectively.

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.

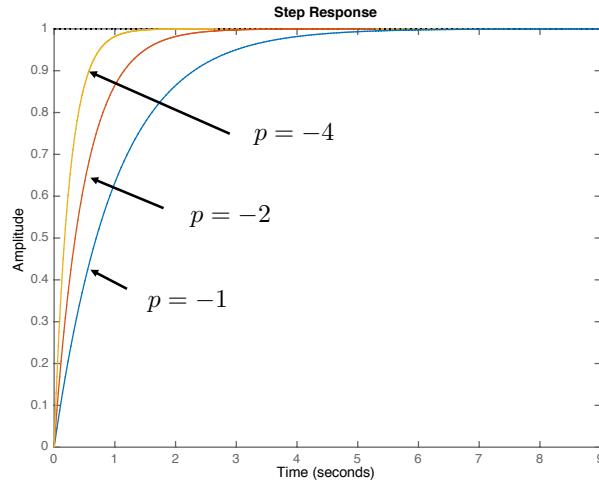


Figure 8.2: Step response of a first order system with poles $p = -1$, $p = -2$, and $p = -4$. The rise time increases as the pole location moves further into the left half of the complex plane.

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:

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

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

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

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 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.

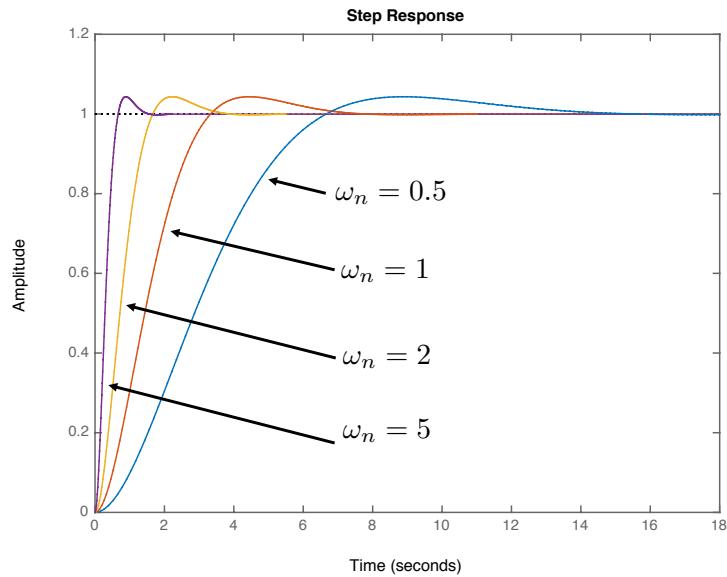


Figure 8.3: Second order response for $\omega_n = 0.5, 1, 2, 5$, and $\zeta = 0.707$.

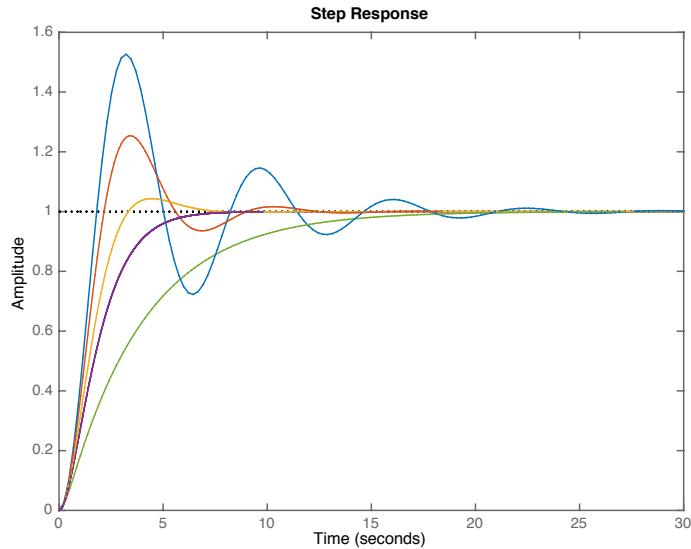


Figure 8.4: Second order response for $\omega_n = 1$, and $\zeta = 0.2, 0.4, 0.6, 0.707, 0.9, 1, 2$.

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.

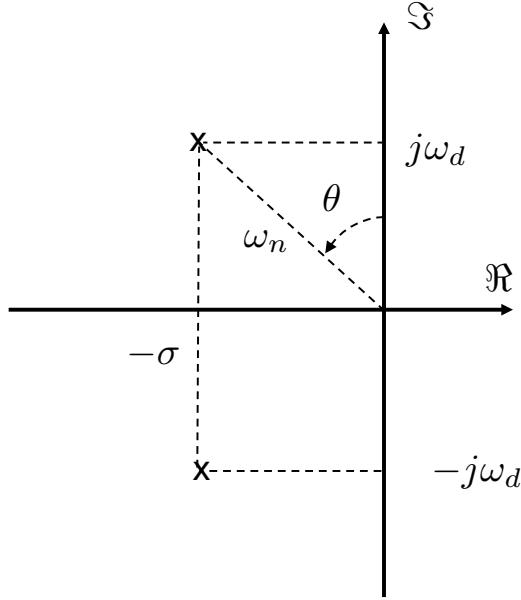


Figure 8.5: Complex conjugate poles in the left half plane.

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) gives

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

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 ??, 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 $0 < \zeta < 1$.

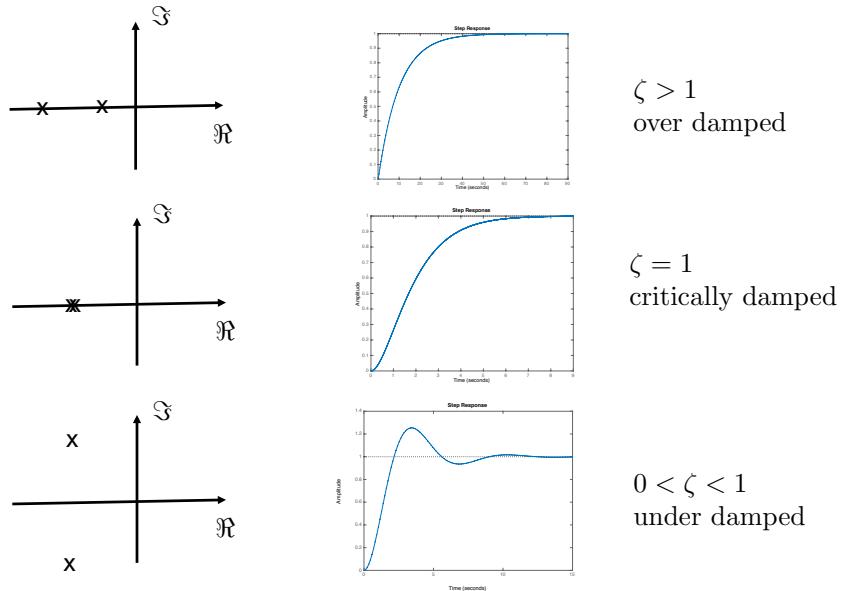


Figure 8.6: Second order response as a function of the damping ratio ζ .

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, when the poles are complex, results in

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

A plot of this signal is shown in Figure 8.7.

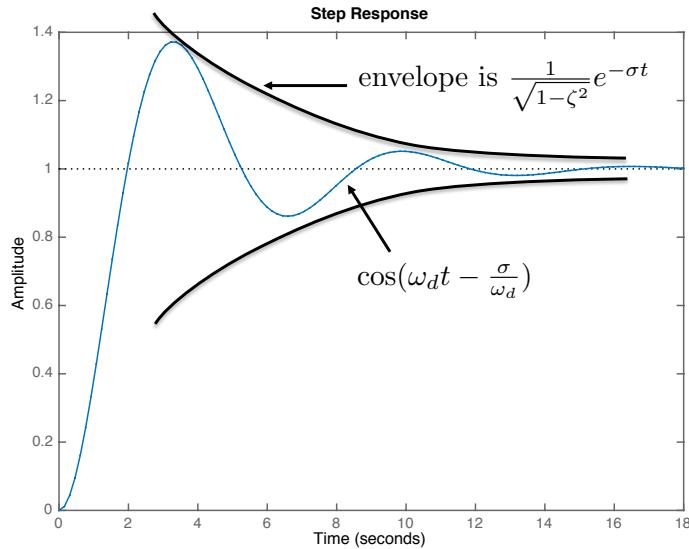


Figure 8.7: Second order response. The steady state value is A . The envelope is $\frac{1}{\sqrt{1-\zeta^2}} e^{-\sigma t}$, and the frequency and phase shift are given by ω_d and $\frac{\sigma}{\omega_d}$ respectively.

8.1.3 Time Domain Specifications

In this section we define several quantities that are often used as design specifications for feedback control systems. These specifications are shown in Figure 8.8.

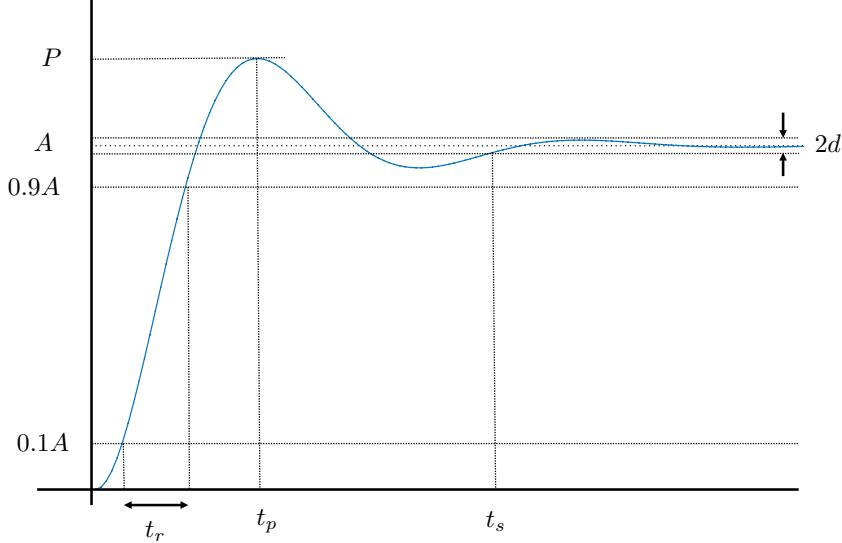


Figure 8.8: Plot defining important time domain specifications like rise time, settling time, and maximum percent overshoot.

As shown in the figure, the *rise time* t_r is defined to be time that it takes the output to transition from 10% to 90% of its final value A . The *settling time* t_s is defined to be the time it takes the output to get within d of its final value A . The threshold d is often defined in terms of a percentage of A , where 1%, 2%, and 5% are common values. The *peak value* P is the maximum value of the step response, and the time at which the peak is achieved is defined to be the *peak time* t_p . The *maximum percent overshoot* M_p is defined as

$$M_p = 100 \left(\frac{P - A}{A} \right),$$

and quantifies how far the response overshoots its desired value.

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

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

Plugging this result back into the derivative to get the peak response gives

$$P = A \left(1 + e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} \right).$$

Therefore, the maximum percent overshoot is

$$M_p = 100e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}, \quad 0 < \zeta < 1. \quad (8.5)$$

To get a simple expression for the rise time we use the approximation

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

When $\zeta = 0.707$ we get

$$t_r \approx \frac{2.2}{\omega_n}. \quad (8.6)$$

Suppose that we would like the settling time t_s to denote that the response settles to within $a\%$ of the final value A , then

$$a = 100 \frac{d}{A} \implies d = \frac{aA}{100}.$$

Therefore, we need to find t_s such that

$$|y(\tau) - A| < \frac{aA}{100}, \quad \text{for all } \tau \geq t_s.$$

Plugging in from Equation (8.4) gives

$$\left| A + A \frac{e^{-\sigma t_s}}{\sqrt{1 - \zeta^2}} \cos \left(\omega_d t - \frac{\sigma}{\omega_d} \right) - A \right| < \frac{aA}{100}.$$

Using only the envelope function to evaluate the last time that the response can possibly enter the threshold gives

$$\frac{e^{-\sigma t_s}}{\sqrt{1 - \zeta^2}} < \frac{a}{100}.$$

Solving for t_s results in

$$t_s = \frac{1}{\sigma} \ln \left(\frac{100}{a \sqrt{1 - \zeta^2}} \right).$$

When $\zeta = 0.707$ we get the approximation

$$t_s \approx \frac{1}{\sigma} \ln \left(\frac{141}{a} \right). \quad (8.7)$$

Suppose that in a problem description we are given specifications for rise time, maximum percent overshoot, and settling time. For example, suppose that we are to choose second order poles so that

- $t_r < a$ seconds,
- $M_p < b\%$,
- $t_s < c$ seconds (within $d\%$).

From Equation (8.6) we get that

$$\begin{aligned} t_r &< b \\ \implies \frac{2.2}{\omega_n} &< b \\ \implies \omega_n &> \frac{2.2}{b}. \end{aligned}$$

Similarly, from Equation (8.5) we have

$$\begin{aligned} M_p &< b\% \\ \implies 100e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} &< b \\ \implies -\frac{\zeta\pi}{\sqrt{1-\zeta^2}} &< \ln\left(\frac{b}{100}\right) \\ \implies \frac{\zeta\pi}{\sqrt{1-\zeta^2}} &> \ln\left(\frac{100}{b}\right) \\ \implies \zeta^2\pi^2 &> (1-\zeta^2)\ln^2\left(\frac{100}{b}\right) \\ \implies \zeta^2 \left[\pi^2 + \ln^2\left(\frac{100}{b}\right) \right] &> \ln^2\left(\frac{100}{b}\right) \\ \implies \zeta &> \sqrt{\frac{\ln^2\left(\frac{100}{b}\right)}{\pi^2 + \ln^2\left(\frac{100}{b}\right)}} \\ \implies \theta &> \sin^{-1} \sqrt{\frac{\ln^2\left(\frac{100}{b}\right)}{\pi^2 + \ln^2\left(\frac{100}{b}\right)}}. \end{aligned}$$

From Equation (8.7) we get

$$\begin{aligned} t_s &< c \\ \implies \frac{1}{\sigma} \ln\left(\frac{141}{d}\right) &< c \\ \implies \sigma &> \frac{1}{c} \ln\left(\frac{141}{d}\right). \end{aligned}$$

Summary

$$\Delta_{\text{polar}}(s) = s^2 + 2\zeta\omega_n s + \omega_n^2$$

$$p_{\text{polar}} = -\zeta\omega_n \pm j\omega_n\sqrt{1 - \zeta^2}$$

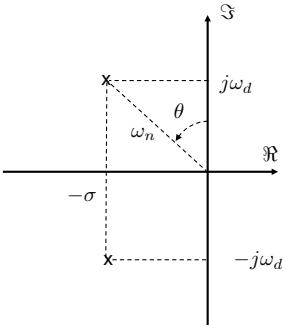
$$\Delta_{\text{rectangular}}(s) = s^2 + 2\sigma s + (\sigma^2 + \omega_d^2)$$

$$p_{\text{rectangular}} = -\sigma \pm j\omega_d$$

$$\zeta = \sin \theta$$

$$\sigma = \zeta\omega_n$$

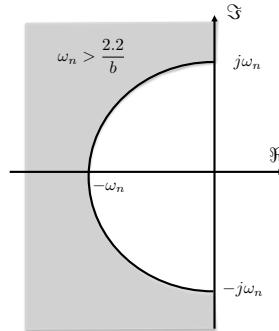
$$\omega_n^2 = \sigma^2 + \omega_d^2$$



Rise time

$$t_r < a \text{ seconds}$$

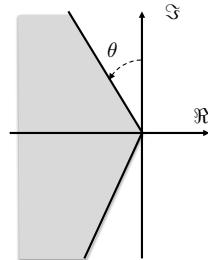
$$\implies \omega_n > \frac{2.2}{a}$$



Maximum Percent Overshoot

$$M_p < b\%$$

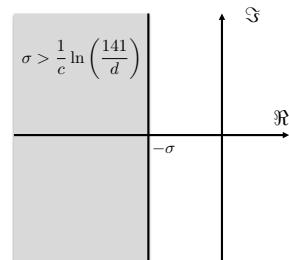
$$\implies \theta > \sin^{-1} \sqrt{\frac{\ln^2(\frac{100}{b})}{\pi^2 + \ln^2(\frac{100}{b})}}$$



Settling Time

$$t_s < c \text{ seconds (within } d\%)$$

$$\implies \sigma > \frac{1}{c} \ln \left(\frac{141}{d} \right)$$



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

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. 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

$$\begin{aligned} y(t) &= \mathcal{L}^{-1} \left\{ \frac{6}{s(s+2)(s+3)} \right\} \\ &= \mathcal{L}^{-1} \left\{ \frac{1}{s} + \frac{-3}{s+2} + \frac{2}{s+3} \right\} \\ &= 1 - 3e^{-2t} + 2e^{-3t}, \quad t \geq 0. \end{aligned} \tag{8.8}$$

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

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

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 (8.8).

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

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

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 8.9 shows the step response for different values of τ , where it can be seen that adding a zero has a strong effect on rise time, settling time, and maximum percent overshoot.

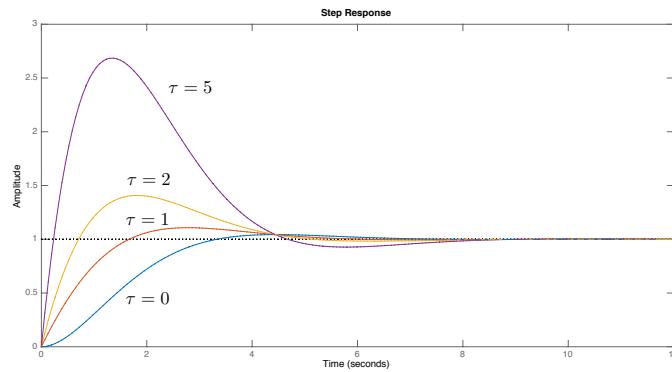
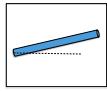


Figure 8.9: Step response of a second order system to a single zero.

8.2 Design Study A. Single Link Robot Arm



Homework Problem A.10

(a) For the single link robot arm, suppose that the design requirements are:

- Rise time: $t_r < 0.8$ seconds,
- Maximum percent overshoot: $M_p < 10\%$,
- Settling time: $t_s < 2$ seconds, within 1%.

Find the region in the s-plane where the poles would result in the design requirements being satisfied.

- (b) Select pole locations that satisfy the design requirements and also minimize the distance to the origin in the complex plane. Minimizing the distance to the origin will result in minimal control action, i.e., $u(t)$ will be small in some sense.
- (c) Using the PD architecture shown in Figure ??, find the proportional gain k_p and the derivative gain k_d that place the poles at the location selected in part (b).

- (d) Implement the PD controller in Simulink when $\theta^d(t)$ is a square wave of magnitude 25 degrees and frequency 0.05 Hz. Verify that the step response satisfies the design specifications.

Solution

From the summary at the end of Section 8.1.3 we have

$$t_r < 0.8 \implies \omega_n > \frac{2.2}{0.8} = 2.75$$

$$M_p < 10\% \implies \theta > \sin^{-1} \sqrt{\frac{\ln^2(\frac{100}{10})}{\pi^2 + \ln^2(\frac{100}{10})}} = 36 \text{ degrees}$$

$$t_s < 2 \implies \sigma > \frac{1}{2} \ln \left(\frac{141}{1} \right) = 2.47.$$

The possible pole locations are shown in Figure 8.10.

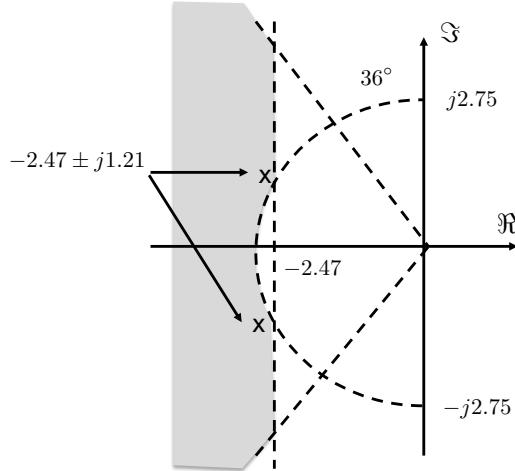


Figure 8.10: Possible pole locations for problem A.10.

To select the pole locations, we pick the poles within the shaded region of Figure 8.10 so that the distance to the origin is minimized. This will be when the real part is equal to -2.47, and the magnitude is equal to 2.75. Therefore, we can solve for the imaginary part m as

$$\sqrt{(2.47)^2 + m^2} = 2.75 \implies m = 1.21.$$

Therefore, the desired closed loop poles are $-2.47 \pm j1.21$, which implies that the desired closed loop characteristic polynomial is

$$\Delta_{cl}^d(s) = (s + 2.47 + j1.21)(s + 2.47 - j1.21) = s^2 + 4.94s + 7.564. \quad (8.9)$$

From Section ?? we have that with PD control, the closed loop characteristic equation is

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

Therefore, equating (8.9) and (8.10) and solving for k_P and k_D we get

$$\begin{aligned} k_P &= 0.1135 \\ k_D &= 0.0641. \end{aligned}$$

The Simulink file associated with this problem is included on the wiki associated with the book.

Notes and References

Chapter 9

Successive Loop Closure

9.1 Theory

When we derived the Laplace transform in Chapter 5 we saw that for many systems, the transfer functions can be arranged cascade form as shown in Figure 9.1 where the input to the system is u and the output is y_3 , but where there are intermediary values that represent the natural physical coupling between internal variable. Many physical systems can be represented by a cascade of systems. When the system is represented as a cascade, a powerful

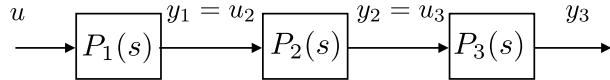


Figure 9.1: Open-loop transfer function modeled as a cascade of three transfer functions.

design method is to use what is called *successive loop closure*. The basic idea behind successive loop closure is to close several simple feedback loops in succession around the open-loop plant dynamics rather than designing a single (presumably more complicated) control system. To illustrate how this approach can be applied, consider the open-loop system shown in Figure 9.1. The open-loop dynamics are given by the product of three transfer functions in series: $P(s) = P_1(s)P_2(s)P_3(s)$. We assume that each of the transfer functions has an output (y_1 , y_2 , y_3) that can be measured and used for feedback. Typically, each of the transfer functions, $P_1(s)$, $P_2(s)$, $P_3(s)$, is of relatively low order – usually first or second order. In this case, we are

interested in controlling the output y_3 . Instead of closing a single feedback loop around y_3 , we will instead close feedback loops around y_1 , y_2 , and y_3 in succession as shown in Figure 9.2.

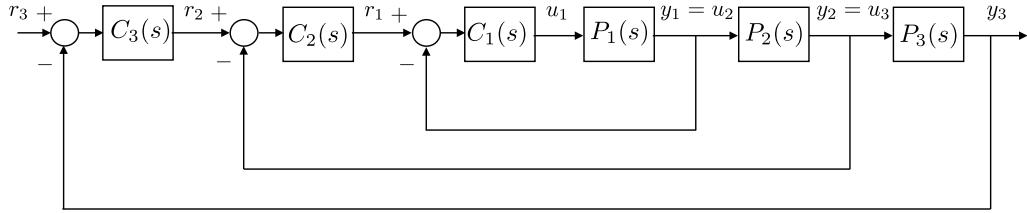


Figure 9.2: Three stage successive loop closure design.

We will design the compensators $C_1(s)$, $C_2(s)$, and $C_3(s)$ 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., $t_{r_2} > 5t_{r_1}$. 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 9.2, the goal is to design a closed-loop system from r_1 to y_1 having a rise time of t_{r_1} and a DC-gain equal to one. If the rise time of the inner loop is significantly faster than the rise time of the next loop, and if the DC-gain is one, then relative to the middle loop the inner loop can be effectively modeled as a gain of one. This is depicted schematically in Figure 9.3. With the inner-loop transfer function modeled as a gain of 1, design of the second loop is simplified because it only includes the plant transfer function $P_2(s)$ and the compensator $C_2(s)$. The critical design step in closing the loops successively is to design the rise time of the next loop so that it is a factor of 5 to 10 slower than the preceding loop. In this case, we require $t_{r_2} > \frac{1}{W_1} t_{r_1}$, where $W_1 > 5$ is a design parameter. If the medium loop also has a DC-gain of one, then relative to the outermost loop, the transfer function from $r_2(s)$ to $y_2(s)$ can be replaced with a gain of 1 for the design of the outermost loop, as shown in Figure 9.4. Again, there is a rise time constraint on the design of the outer loop: $t_{r_3} > \frac{1}{W_2} t_{r_2}$, where $W_2 > 5$ is again a design parameter. Because each of the plant models $P_1(s)$, $P_2(s)$, $P_3(s)$ are first or second order, conventional PID compensators can be

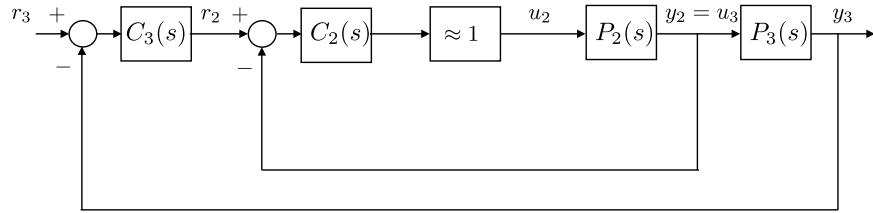


Figure 9.3: Successive loop closure design with inner loop modeled as a unity gain.

employed effectively. The loopshaping techniques discussed in Part V can also be effectively used for successive loop closure.

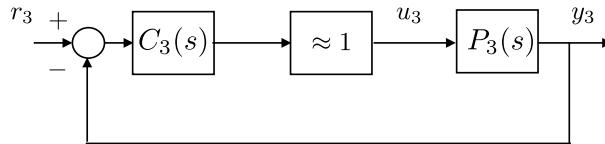
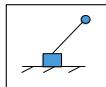


Figure 9.4: Successive loop closure design with two inner loops modeled as a unity gain.

9.2 Design Study B. Inverted Pendulum



Homework Problem B.11

- (a) For inverted pendulum, 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 $t_{r_z} = 50t_{r_\theta}$ 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.

Solution

The block diagram for the inner loop is shown in Figure 9.5.

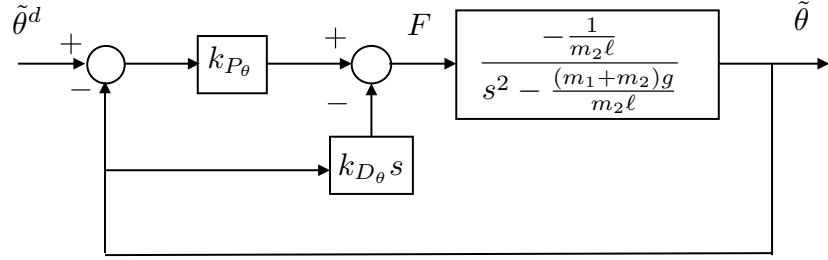


Figure 9.5: Block diagram for inner loop of inverted pendulum control

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

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

Therefore the closed loop characteristic equation is

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

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

$$\omega_{n_\theta} = \frac{2.2}{t_{r_\theta}} = 4.4$$

$$\zeta_\theta = 0.707.$$

Therefore

$$k_{P_\theta} = -(m_1 + m_2)g - m_2\ell\omega_{n_\theta}^2 = -21.93$$

$$k_{D_\theta} = -2\zeta_\theta\omega_{n_\theta}m_2\ell = -3.11.$$

The DC gain of the inner loop is given by

$$k_{DC_\theta} = \frac{k_{P_\theta}}{(m_1 + m_2)g + k_{P_\theta}} = 0.3214.$$

Replacing the inner loop by its DC gain, the block diagram for the outer loop is shown in Figure 9.6.

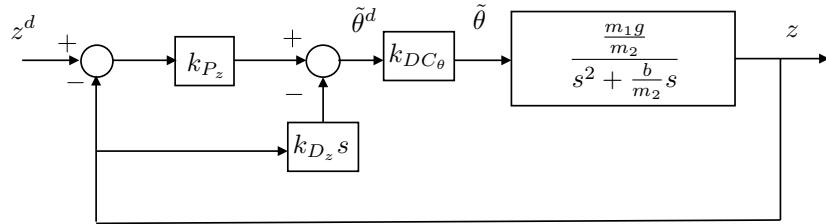


Figure 9.6: Block diagram for outer loop of inverted pendulum control

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

$$Z(s) = \frac{m_1 g k_{DC_\theta} k_{P_z} / m_2}{s^2 + \left(\frac{b}{m_2} + \frac{m_1 g k_{DC_\theta} k_{D_z}}{m_2} \right) s + \left(\frac{m_1 g k_{DC_\theta} k_{P_z}}{m_2} \right)} Z^d(s).$$

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

$$\Delta_{cl}(s) = s^2 + \left(\frac{b}{m_2} + \frac{m_1 g k_{DC_\theta} k_{D_z}}{m_2} \right) s + \left(\frac{m_1 g k_{DC_\theta} k_{P_z}}{m_2} \right).$$

The desired characteristic equation is

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

where

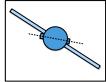
$$\begin{aligned} t_{r_z} &= 50t_{r_\theta} = 25 \\ \omega_{n_z} &= \frac{2.2}{t_{r_z}} = 0.088 \\ \zeta_z &= 0.707. \end{aligned}$$

The PD gains are therefore

$$\begin{aligned} k_{P_z} &= \frac{m_2\omega_{n_z}^2}{m_1 g k_{DC_z}} = 0.0098 \\ k_{D_z} &= \frac{2\zeta_z\omega_{n_z} m_2 - b}{m_1 g k_{DC_\theta}} = 0.0945. \end{aligned}$$

The Simulink files are on the wiki page associated with the book.

9.3 Design Study C. Satellite Attitude Control



Homework Problem C.11

- (a) For simple satellite system, using the principle of successive loop closure, draw a block diagram that uses PD control for the inner loop control and proportional control for the outer loop control. The input to the outer loop controller is the desired angle of the solar panel ϕ^d and the output is the desired angle of the satellite θ^d . The input to the inner loop controller is θ^d 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 proportional gain k_{P_ϕ} so that the rise time of the outer loop is $t_{r_\phi} = 2t_{r_\theta}$.
- (e) Find the DC gain of the output loop.

- (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.

Solution

The block diagram for the inner loop is shown in Figure 9.7.

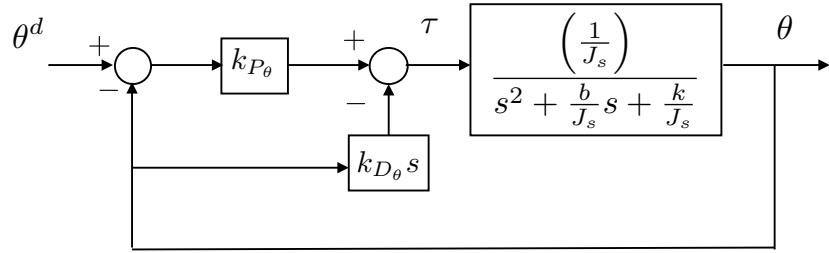


Figure 9.7: Block diagram for inner loop of satellite control

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

$$\omega_{n\theta} = \frac{2.2}{t_{r\theta}} = 1.1$$

$$\zeta_\theta = 0.707.$$

Therefore

$$k_{P\theta} = \omega_{n\theta}^2 J_s - k = 5.9$$

$$k_{D\theta} = 2\zeta_\theta \omega_{n\theta} J_s - b = 7.727.$$

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 9.8.

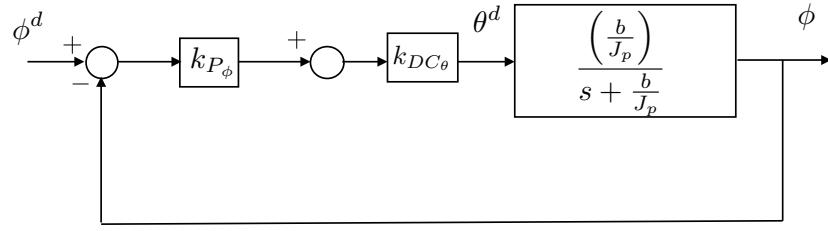


Figure 9.8: Block diagram for outer loop of satellite attitude control

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

$$\Phi(s) = \frac{\left(\frac{bk_{DC_\theta}k_{P_\theta}}{J_p}\right)}{s + \left(\frac{b+bk_{DC_\theta}k_{P_\phi}}{J_p}\right)} \Phi^d(s).$$

The DC gain for the outer loop is equal to

$$k_{DC_\phi} = \frac{bk_{DC_\theta}k_{P_\theta}}{b + bk_{DC_\theta}k_{P_\phi}},$$

which implies that there will be steady state error.

The closed loop characteristic equation is

$$\Delta_{cl}(s) = s + \left(\frac{b + bk_{DC_\theta}k_{P_\phi}}{J_p}\right).$$

The desired characteristic equation is

$$\Delta_{cl}^d(s) = s + p_{cl},$$

where

$$t_{r_\phi} = 2t_{r_\theta} = 4$$

$$p_{cl} = \frac{1}{t_{r_\phi}} = 0.25$$

The proportional gain is therefore

$$k_{P_\phi} = \frac{J_p p_{cl} - b}{b k_{DC_\theta}} = 4.1017.$$

The DC gain of the outer loop is

$$k_{DC_\phi} = 0.8519.$$

The Simulink files are on the wiki page associated with the book.

Notes and References

Chapter 10

Saturation and limits of performance

RWB: Remove this chapter and move some of the material to a section in the previous chapter. Explain it as a method for finding an upper bound on ω_n , but still focus on ω_n/t_r and ζ as tuning parameters.

All physical systems have limits to their possible performance. These limits include

- Input saturation constraints,
- Input slew rate constraints,
- Output constraints (like the rail limits in the ball and beam problem),
- Power or energy constraints.

As an introduction on how these constraints might be addressed, we will consider how input saturation constraints impact the possible rise time achievable by the closed loop system.

All physical systems have input constraints. 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. 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 10.1 shows a saturation block in conjunction with a PD control scheme. Mathematically,

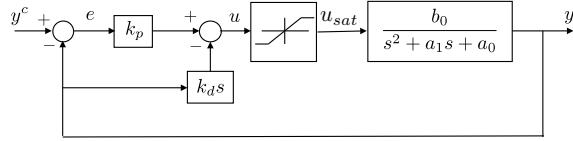


Figure 10.1: Control system example.

the saturation block is given by

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

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 design the feedback gains so that the system remains linear by keeping the input out of saturation.

Given the second-order system shown in Figure 10.1 with proportional feedback on the output error and derivative feedback on the output, when the system is not in saturation, the closed-loop transfer function is

$$\frac{y}{y^c} = \frac{b_0 k_p}{s^2 + (a_1 + b_0 k_d)s + (a_0 + b_0 k_p)}. \quad (10.1)$$

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, where $u^{max} = k_p e^{max}$. 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

$$k_p = \frac{u^{max}}{e^{max}}, \quad (10.2)$$

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.

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

$$\begin{aligned}\omega_n &= \sqrt{a_0 + b_0 k_p} \\ &= \sqrt{a_0 + b_0 \frac{u^{\max}}{e^{\max}}},\end{aligned}$$

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 is

$$t_r = \frac{2.2}{\omega_n} = \frac{2.2}{\sqrt{a_0 + b_0 \frac{u^{\max}}{e^{\max}}}},$$

which is the fastest possible achievable rise time for an error step of e^{\max} .

10.0.1 Saturation Constraint

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 10.2, where it is clear that the value for \tilde{u}_{\max} can be succinctly written as

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

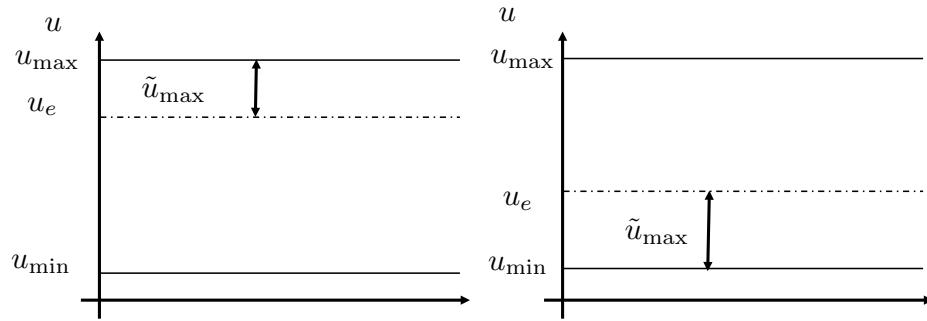
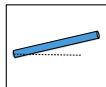


Figure 10.2: The saturation constraint for \tilde{u} depends on the value of the equilibrium input u_e .

10.1 Design Study A. Single Link Robot Arm



Homework Problem A.12

Suppose that the size of the input torque is limited to $\tau_{\max} = 1 \text{ Nm}$.

- (a) Considering the equilibrium torque τ_e , derive a bound $\tilde{\tau}_{\max}$ on the linearized torque $\tilde{\tau}$ such that $|\tilde{\tau}| \leq \tilde{\tau}_{\max}$ when the maximum possible equilibrium torque is used.
- (b) Select the proportional gain k_P so that $\tilde{\tau}$ just saturates when a step of size $A_{th} = 50$ degrees is placed on $\tilde{\theta}_d$.
- (c) Find the corresponding natural frequency and rise time, and find k_D that results in a damping ratio of $\zeta = 0.707$.
- (d) Modify the Simulink diagram developed in Problem A.10, to include a saturation block on the torque τ , and implement PD control with the revised gains.

Solution

From Problem A.5, the equilibrium torque is $\tau_e = \frac{mg\ell}{2} \cos \theta_e$. The output torque of the controller is $\tilde{\tau} = \tau - \tau_e$ which implies that $\tau = \tau_e + \tilde{\tau}$. Therefore,

since the total torque is constrained as

$$|\tau| \leq \tau_{\max},$$

we can constrain the total torque output by the controller as

$$|\tau| = |\tau_e + \tilde{\tau}| \leq |\tau| + |\tilde{\tau}| \leq \tau_{\max}.$$

Therefore the torque constraint will be satisfied if

$$|\tilde{\tau}| \leq \tau_{\max} - |\tau_e| \leq 1 - \frac{mg\ell}{2} \stackrel{\triangle}{=} \tilde{\tau}_{\max}.$$

When a step size of $\theta^d = A_\theta$ is placed on the closed loop system, the size of the torque immediately after $t = 0$ is given by

$$\tilde{\tau} = k_P A_\theta.$$

Therefore setting

$$k_P = \frac{\tilde{\tau}_{\max}}{A_\theta}$$

will result in the fast possible rise time for a step of size A_θ . From a practical point of view, A_θ will be a tuning parameter.

As seen in Problem A.8, the closed loop characteristic polynomial is given by

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

Since the desired closed loop characteristic polynomial is given by

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

we have that

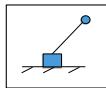
$$\omega_n = \sqrt{66.67k_P} = \sqrt{66.67 \frac{\tilde{\tau}_{\max}}{A_\theta}}.$$

Similarly, the derivative gain is given by

$$k_D = \frac{2\zeta\omega_n - 0.667}{66.67}.$$

The wiki page associated with the book contains the Simulink files.

10.2 Design Study B. Inverted Pendulum



Homework Problem B.12

Suppose that the size of the input force on the cart is limited to $F_{\max} = 5$ N. The objective of this problem is to revise the successive loop closure design from Homework B.11 to maximize performance (with this design method).

- (a) Since the equilibrium force $F_e = 0$, the bound on the linearized force is $|\tilde{F}| \leq F_{\max}$.
- (b) Select the proportional gain k_{P_θ} so that F just saturates when a step of size A_{th} is placed on $\tilde{\theta}^d$.
- (c) Find the proportional gain k_{D_θ} so that the desired closed loop characteristic polynomial is $\Delta_{cl}^d(s) = s^2 + 2\zeta_\theta\omega_{n_\theta}s + \omega_{n_\theta}^2$, where $\zeta_\theta = 0.707$.
- (d) Find the DC gain k_{DC_θ} of the inner loop.
- (e) Replacing the inner loop with its DC gain, find the proportional gain k_{P_z} of the outer loop so that $\tilde{\theta}^d$ just saturates at A_{th} when a step of size $A_z = 1$ meters is placed on \tilde{z}^d .
- (f) Find the derivative gain k_{D_z} so that the damping ratio of the outer loop is $\zeta_z = 0.707$.
- (g) Modify the Simulink diagram developed in HW B.11 to include a saturation block on the force F , and implement the successive loop closure design using the revised gains.

Solution

When a step size of $\tilde{\theta}^d = A_\theta$ is placed on the closed loop system, the size of the force immediately after $t = 0$ is

$$F = k_{P_\theta}A_\theta.$$

Therefore setting

$$k_{P_\theta} = \frac{F_{\max}}{A_\theta}$$

will result in the fast possible rise time for a step of size A_θ . From a practical point of view, A_θ will be a tuning parameter.

As seen in Problem B.8, the closed loop characteristic polynomial is given by

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

Since the desired closed loop characteristic polynomial is given by

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

we have that

$$\omega_n = \sqrt{-\frac{(m_1 + m_2)g}{m_2 \ell} - \frac{k_{P_\theta}}{m_2 \ell}},$$

and the derivative gain is given by

$$k_D = -2\zeta \omega_{n_\theta} m_2 \ell.$$

The DC-gain of the inner loop is

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

When a step size of A_z is placed on z^d , immediately after time $t = 0$ we have that $\theta^d = k_{P_z} A_z$. To achieve maximum performance we assign $\theta^d = -A_\theta$ at $t = 0$. Therefore, the proportional gain is

$$k_{P_z} = -\frac{A_\theta}{A_z}.$$

From HW B.11 the closed loop characteristic polynomial is

$$\Delta_{cl}(s) = s^2 + \left(\frac{b}{m_2 \frac{m_1 g k_{DC_\theta} k_{P_z}}{m_2}} \right) - \frac{m_1 g k_{DC_\theta} k_{P_z}}{m_2}.$$

Since the desired closed loop characteristic polynomial is

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

the natural frequency of the outer loop is

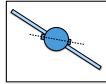
$$\omega_{n_z} = \sqrt{-\frac{m_1 g k_{DC_\theta} k_{P_z}}{m_2}},$$

and the derivative gain is given by

$$k_{D_z} = \frac{m_2}{m_1 g k_{DC_\theta}} \left(2\zeta_z \omega_{n_z} - \frac{b}{m_2} \right).$$

The wiki page associated with the book contains the Simulink files. The tuning parameters for the controller are A_θ , ζ_θ and ζ_z .

10.3 Design Study C. Satellite Attitude Control



Homework Problem C.12

Suppose that the size of the input torque on the satellite body is limited to $\tau_{\max} = 5$ Nm. The objective of this problem is to revise the successive loop closure design from Homework C.11 to maximize performance (with this design method).

- (a) Since the equilibrium torque $\tau_e = 0$, the bound on the torque is $|\tau| \leq \tau_{\max}$.
- (b) Select the proportional gain k_{P_θ} so that τ just saturates when a step of size A_θ is placed on $\tilde{\theta}^d$.
- (c) Find the proportional gain k_{D_θ} so that the damping ratio of the inner loop is $\zeta_\theta = 0.707$.
- (d) Find the DC gain k_{DC_θ} of the inner loop.
- (e) Replacing the inner loop with its DC gain, find the proportional gain k_{P_ϕ} of the outer loop so that θ^d just saturates at A_θ when a step of size $A_\phi = 30$ degrees is placed on ϕ^d .

- (f) Modify the Simulink diagram developed in HW C.11 to include a saturation block on the torque τ , and implement the successive loop closure design using the revised gains.

Solution

When a step size of $\theta^d = A_\theta$ is placed on the closed loop system, the size of the torque immediately after $t = 0$ is

$$\tau = k_{P_\theta} A_\theta.$$

Therefore setting

$$k_{P_\theta} = \frac{\tau_{\max}}{A_\theta}$$

will result in the fast possible rise time for a step of size A_θ . From a practical point of view, A_θ will be a tuning parameter.

As seen in Problem C.8, the closed loop characteristic polynomial is given by

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

Since the desired closed loop characteristic polynomial is given by

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

we have that

$$\omega_n = \sqrt{\frac{k + k_{P_\theta}}{J_s}},$$

and the derivative gain is given by

$$k_{D_\theta} = 2\zeta\omega_{n_\theta} J_s - b.$$

The DC-gain of the inner loop is

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

When a step size of A_ϕ is placed on ϕ^d , immediately after time $t = 0$ we have that $\theta^d = k_{P_\phi} A_\phi$. To achieve maximum performance we assign $\theta^d = A_\theta$ at $t = 0$. Therefore, the proportional gain is

$$k_{P_\phi} = \frac{A_\theta}{A_\phi}.$$

From HW C.11 the closed loop characteristic polynomial of the outer loop is

$$\Delta_{cl}(s) = s + \left(\frac{b + bk_{DC_\theta}k_{P_\phi}}{J_p} \right).$$

Note that the DC-gain of the outer loop is

$$k_{DC_\phi} = \frac{k_{DC_\theta}k_{P_\phi}}{1 + k_{DC_\theta}k_{P_\phi}} \neq 1.$$

The wiki page associated with the book contains the Simulink files. The tuning parameters for the controller are A_θ and ζ_θ .

Notes and References

Chapter 11

System Type and Integrators

11.1 Theory

The key theoretical tool to understanding the steady state error and the effect of integrators is the final value theorem.

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

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

Consider the closed loop system shown in Figure 11.1

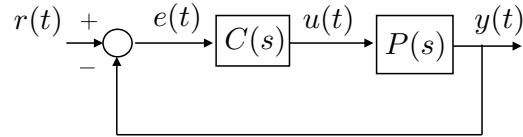


Figure 11.1: Block diagram for reference tracking.

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

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

Step Input.

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

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

Assuming that the closed loop system is stable and therefore that $\lim_{t \rightarrow \infty} e(t)$ is finite, then the final value theorem gives

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

Let

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

then

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

Now suppose that

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

then

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

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

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

In this case we have

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

and the steady state error is

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

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

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

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

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

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

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

Defining

$$M_v = \lim_{s \rightarrow 0} sG(s)P(s),$$

gives

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

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

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

then

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

Therefore

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

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

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

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 $G(s)P(s)$ 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 $G(s)P(s)$ 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 11.1, the steady state tracking error when $R(s) = \frac{1}{s^{q+1}}$ is given by

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

Table 11.1 summarizes the tracking error for different inputs as a function of system type, where

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

$$M_v = \lim_{s \rightarrow 0} sC(s)P(s)$$

$$M_a = \lim_{s \rightarrow 0} s^2C(s)P(s).$$

Table 11.1: 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 | ... |
| : | : | : | : | |

As an example, consider the feedback system shown in Figure 11.2. The

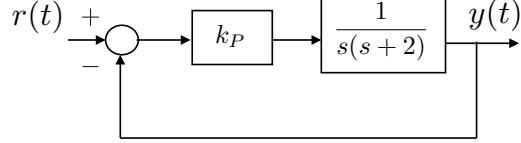


Figure 11.2: The system type for reference tracking for this system is type 1.

open loop system

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

has one free integrator (pole at zero) and therefore the system is type 1. From Table 11.1 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

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

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 11.3.

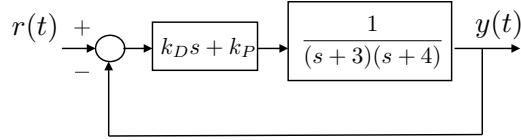


Figure 11.3: The system type for reference tracking for this system is type 0.

In this case, the open loop system

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

does not have any free integrators (poles at zero) and therefore the system is type 0. From Table 11.1 the tracking error when the input is a step is

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

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 11.4 where

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

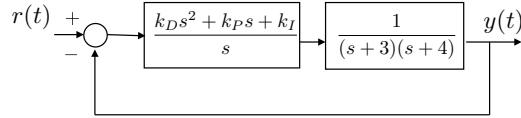


Figure 11.4: The system type for reference tracking for this system is type 1.

then the open loop system

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

has one free integrator (pole at zero) and therefore the system becomes type 1 by design. From Table 11.1 the tracking error when the input is a step is zero, and the tracking error when the input is a ramp is

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

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

11.1.1 System Type for Input Disturbances

Consider the general feedback loop shown in Figure 11.5. 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.

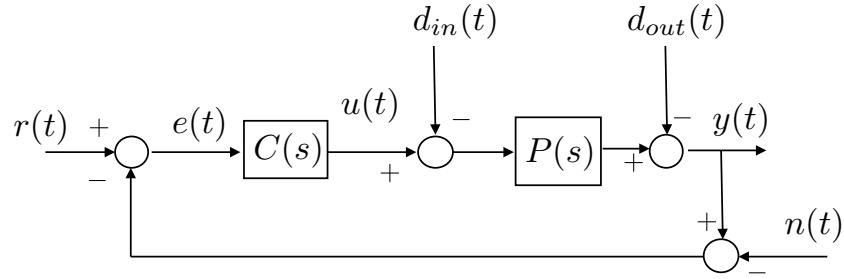


Figure 11.5: General feedback loop with reference input $r(t)$, input disturbance $d_{in}(t)$, output disturbance $d_{out}(t)$, and sensor noise $n(t)$.

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

$$E(s) = R(s) + N(s) + D_{out}(s) + P(s)D_{in}(s) - C(s)P(s)E(s).$$

Solving for $E(s)$ gives

$$\begin{aligned} E(s) &= \frac{1}{1 + C(s)P(s)}R(s) + \frac{1}{1 + C(s)P(s)}N(s) \\ &\quad + \frac{1}{1 + C(s)P(s)}D_{out}(s) + \frac{P(s)}{1 + C(s)P(s)}D_{in}(s). \end{aligned}$$

Since the transfer function from $N(s)$ and $D_{out}(s)$ to $E(s)$ is identical to the transfer function from $R(s)$ to $E(s)$. Therefore, 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 is

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

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 11.6. Recall from the discussion above, that the system type with respect to the reference input is type 1.

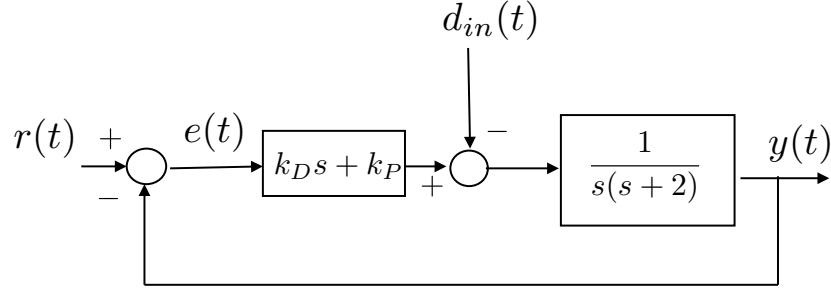


Figure 11.6: The system type for input disturbances for this system is type 0.

For the input disturbance we have

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \left(\frac{P(s)}{1 + C(s)P(s)} \right) \left(\frac{1}{s^q} \right) \\ &= \lim_{s \rightarrow 0} \left(\frac{\frac{1}{s(s+2)}}{1 + (k_Ds + k_P) \left(\frac{1}{s(s+2)} \right)} \right) \left(\frac{1}{s^q} \right) \\ &= \lim_{s \rightarrow 0} \left(\frac{1}{s(s+2) + (k_Ds + k_P)} \right) \left(\frac{1}{s^q} \right) \\ &= \lim_{s \rightarrow 0} \left(\frac{1}{k_P} \right) \left(\frac{1}{s^q} \right) \end{aligned}$$

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 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 11.7, then

$$\begin{aligned}
 \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \left(\frac{P(s)}{1 + C(s)P(s)} \right) \left(\frac{1}{s^q} \right) \\
 &= \lim_{s \rightarrow 0} \left(\frac{\frac{1}{s(s+2)}}{1 + \left(\frac{k_D s^2 + k_P s + k_I}{s} \right) \left(\frac{1}{s(s+2)} \right)} \right) \left(\frac{1}{s^q} \right) \\
 &= \lim_{s \rightarrow 0} \left(\frac{s}{s^2(s+2) + (k_D s^2 + k_P s + k_I)} \right) \left(\frac{1}{s^q} \right) \\
 &= \lim_{s \rightarrow 0} \left(\frac{1}{k_I} \right) \left(\frac{1}{s^{q-1}} \right)
 \end{aligned}$$

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 is zero and the steady state error with respect to a ramp is $\frac{1}{k_I}$ whereas the steady state error with respect to a parabola and higher order polynomials is infinite.

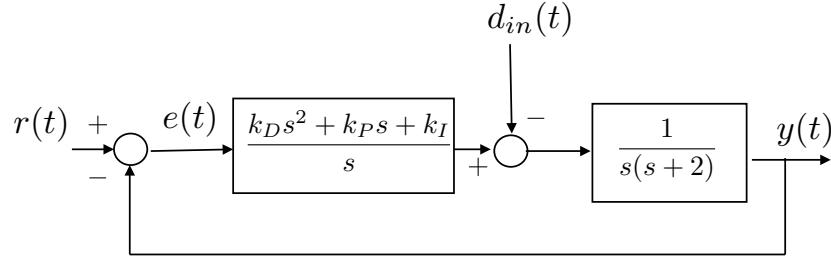
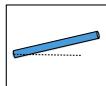


Figure 11.7: The system type for input disturbances for this system is type 1.

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 $C(s)P(s)$. As a consequence, while an integrator may not be necessary to steady state tracking, it is often necessary to reject constant input disturbances.

11.2 Design Study A. Single Link Robot Arm



Homework Problem A.13

- (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 11.8.

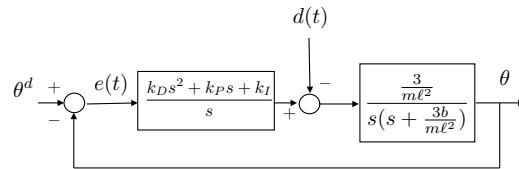


Figure 11.8: Closed loop system for problem HW A.13.

Without the integrator, the open-loop transfer function is given by

$$C(s)P(s) = (k_D s + k_P) \left(\frac{\frac{3}{m\ell^2}}{s(s + \frac{3b}{m\ell^2})} \right).$$

The system has one free integrator and is therefore type 1, which, from Table 11.1 implies that the tracking error when the input is a step is zero, and the tracking error when the input is a ramp is

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

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

With the integrator, the open loop transfer function is

$$C(s)P(s) = \left(\frac{k_D s^2 + k_P s + k_I}{s} \right) \left(\frac{\frac{3}{m\ell^2}}{s(s + \frac{3b}{m\ell^2})} \right).$$

which has two free integrators and is therefore type 2. Therefore, from Table 11.1 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

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{M_a} = \frac{1}{\lim_{s \rightarrow 0} s^2 C(s) P(s)} = \frac{1}{\frac{k_I}{b}} = \frac{b}{k_I}.$$

The tracking error when the input is t^3 or a higher order polynomial, is ∞ .

For the input disturbance, the transfer function from $D(s)$ to $E(s)$ is given by

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

Without the integrator, and when $D(s) = \frac{A}{s^{q+1}}$, the steady state error is given by

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{P}{1 + CP} \frac{A}{s^q} \\ &= \lim_{s \rightarrow 0} \left(\frac{\left(\frac{3}{m\ell^2} \right)}{1 + (k_D s + k_P) \left(\frac{3}{m\ell^2} \right)} \right) \left(\frac{A}{s^q} \right) \\ &= \lim_{s \rightarrow 0} \left(\frac{\frac{3}{m\ell^2}}{s(s + \frac{3b}{m\ell^2}) + (k_D s + k_P) \frac{3}{m\ell^2}} \right) \left(\frac{A}{s^q} \right) \\ &= \frac{A \frac{3}{m\ell^2}}{\frac{3k_P}{m\ell^2}} \\ &= \frac{A}{k_P}, \end{aligned}$$

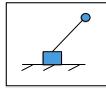
if $q = 0$. Therefore, to an input disturbance the system is type 0. The steady state error when a constant step of size A is placed on $d(t)$ is $\frac{A}{k_P}$.

With the integrator, and when $D(s) = \frac{A}{s^{q+1}}$, the steady state error is given by

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{P}{1 + CP} \frac{A}{s^q} \\ &= \lim_{s \rightarrow 0} \left(\frac{\left(\frac{\frac{3}{m\ell^2}}{s(s + \frac{3b}{m\ell^2})} \right)}{1 + \left(\frac{k_D s^2 + k_P s + k_I}{s} \right) \left(\frac{\frac{3}{m\ell^2}}{s(s + \frac{3b}{m\ell^2})} \right)} \right) \left(\frac{A}{s^q} \right) \\ &= \lim_{s \rightarrow 0} \left(s \frac{\frac{3}{m\ell^2}}{s^2(s + \frac{3b}{m\ell^2}) + (k_D s^2 + k_P s + k_I) \frac{3}{m\ell^2}} \right) \left(\frac{A}{s^q} \right) \\ &= \frac{A \frac{3}{m\ell^2}}{\frac{3k_I}{m\ell^2}} \\ &= \frac{A}{k_I}, \end{aligned}$$

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 $\frac{A}{k_I}$.

11.3 Design Study B. Inverted Pendulum



Homework Problem B.13

- (a) When the inner loop controller for the inverted pendulum 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 inverted pendulum 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?

Solution

The block diagram for the inner loop is shown in Figure 11.9.

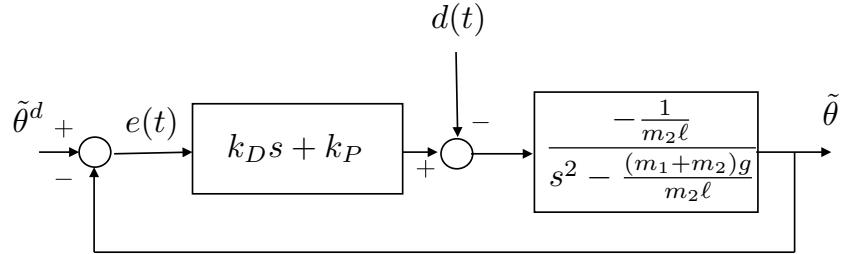


Figure 11.9: Inner loop system for problem HW B.13.

The open loop system is given by

$$P(s)C(s) = \left(\frac{-\frac{1}{m_2 \ell}}{s^2 - \frac{(m_1 + m_2)g}{m_2 \ell}} \right) (k_D s + k_P).$$

Since there are no free integrators, the system is type 0, and from Table 11.1 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 ∞ .

For the disturbance input, the steady state error to a step on $d(t)$ is

$$\begin{aligned}
 \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} s \frac{P(s)}{1 + P(s)C(s)} \frac{1}{s} \\
 &= \lim_{s \rightarrow 0} \frac{\left(\frac{-\frac{1}{m_2 \ell}}{s^2 - \frac{(m_1+m_2)g}{m_2 \ell}} \right)}{1 + \left(\frac{-\frac{1}{m_2 \ell}}{s^2 - \frac{(m_1+m_2)g}{m_2 \ell}} \right) (k_D s + k_P)} \\
 &= \lim_{s \rightarrow 0} \frac{\left(-\frac{1}{m_2 \ell} \right)}{\left(s^2 - \frac{(m_1+m_2)g}{m_2 \ell} \right) + \left(-\frac{1}{m_2 \ell} \right) (k_D s + k_P)} \\
 &= \frac{\left(-\frac{1}{m_2 \ell} \right)}{\left(-\frac{(m_1+m_2)g}{m_2 \ell} \right) + \left(-\frac{k_P}{m_2 \ell} \right)} \\
 &= \frac{1}{(m_1 + m_2)g + k_P}.
 \end{aligned}$$

The system is type 0 with respect to the input disturbance.

The block diagram for the outer loop is shown in Figure 11.10.

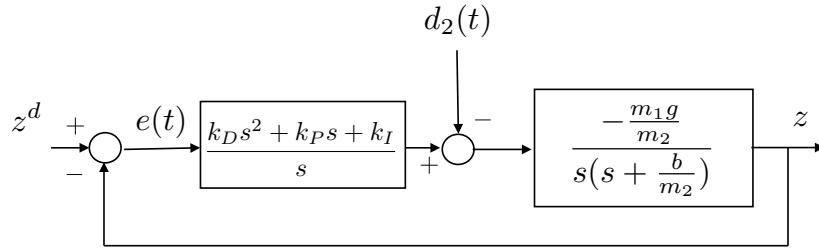


Figure 11.10: Outer loop system for problem HW B.13.

The open loop system is given by

$$P(s)C(s) = \left(\frac{-\frac{m_1 g}{m_2}}{s(s + \frac{b}{m_2})} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right).$$

When $k_I = 0$ there is one free integrator in $P(s)C(s)$ and the system is type 1, and from Table 11.1 the tracking error when the input is a ramp is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{M_v} = \frac{1}{\lim_{s \rightarrow 0} s P(s) C(s)} = \frac{b}{k_P m_1 g}.$$

When $k_I \neq 0$, there are two free integrators in $P(s)C(s)$ and the system is type 2. From Table 11.1 the tracking error when the input is a parabola is

$$\lim_{t \rightarrow \infty} e(t) = \frac{1}{M_a} = \frac{1}{\lim_{s \rightarrow 0} s^2 P(s)C(s)} = \frac{b}{k_I m_1 g}.$$

For the disturbance input, the steady state error when $D(s) = \frac{1}{s^{q+1}}$ is

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} s \frac{P(s)}{1 + P(s)C(s)} \frac{1}{s^{q+1}} = \lim_{s \rightarrow 0} \frac{\left(\frac{-\frac{m_1 g}{m_2}}{s(s + \frac{b}{m_2})} \right)}{1 + \left(\frac{-\frac{m_1 g}{m_2}}{s(s + \frac{b}{m_2})} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right)} \frac{1}{s^q}.$$

Without the integrator, i.e., when $k_I = 0$ we have

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{\left(-\frac{m_1 g}{m_2} \right)}{s(s + \frac{b}{m_2}) + \left(-\frac{m_1 g}{m_2} \right)(k_D s + k_P)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{\left(-\frac{m_1 g}{m_2} \right)}{\left(-\frac{m_1 g}{m_2} \right)(k_P)} \frac{1}{s^q} \end{aligned}$$

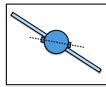
which is finite when $q = 0$. Therefore, the system is type 0 with respect to the input disturbance and the steady state error when $d_2(t)$ is a unit step is $1/k_P$.

When $k_I \neq 0$, we have

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{s \left(-\frac{m_1 g}{m_2} \right)}{s^2(s + \frac{b}{m_2}) + \left(-\frac{m_1 g}{m_2} \right)(k_D s^2 + k_P s + k_I)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{\left(-\frac{m_1 g}{m_2} \right)}{\left(-\frac{m_1 g}{m_2} \right)(k_I)} \frac{1}{s^{q-1}} \end{aligned}$$

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 $1/k_I$.

11.4 Design Study C. Satellite Attitude Control



Homework Problem C.13

- (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 P 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 11.11.

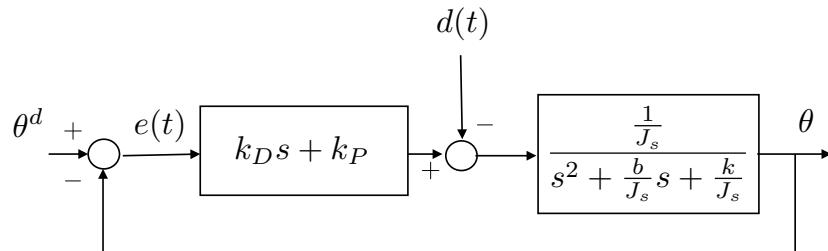


Figure 11.11: Inner loop system for problem HW C.13.

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 11.1 the tracking error when the input is a step is

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

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

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} s \frac{P(s)}{1 + P(s)C(s)} \frac{1}{s^{q+1}} \\ &= \lim_{s \rightarrow 0} \frac{\left(\frac{1}{J_s} \right)}{1 + \left(\frac{1}{s^2 + \frac{b}{J_s}s + \frac{k}{J_s}} \right) (k_D s + k_P)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{\frac{1}{J_s}}{\left(s^2 + \frac{b}{J_s}s + \frac{k}{J_s} \right) + \frac{1}{J_s} (k_D s + k_P)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{\frac{1}{J_s}}{\frac{k}{J_s} + \frac{k_P}{J_s}} \frac{1}{s^q} \end{aligned}$$

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 11.12.

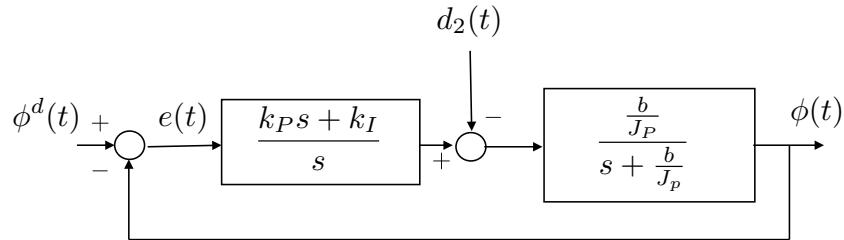


Figure 11.12: Outer loop system for problem HW C.13.

The open loop system is given by

$$P(s)C(s) = \left(\frac{\frac{b}{J_p}}{s + \frac{b}{J_p}} \right) \left(\frac{k_P s + k_I}{s} \right).$$

When $k_I = 0$ there are no free integrator in $P(s)C(s)$ and the system is type 0, and from Table 11.1 the tracking error when the input is a step is

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

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 11.1 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^d is a step is zero, and it is infinite when ϕ^d is a parabola or higher order polynomial.

For the disturbance input, the steady state error when $D(s) = \frac{1}{s^{q+1}}$ is

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

Without the integrator, i.e., when $k_I = 0$ we have

$$\begin{aligned} \lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{\frac{b}{J_p}}{\left(s + \frac{b}{J_p} \right) + \frac{bk_P}{J_p}} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{1}{1 + k_P} \frac{1}{s^q} \end{aligned}$$

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 $1/(1 + k_P)$.

When $k_I \neq 0$, we have

$$\begin{aligned}\lim_{t \rightarrow \infty} e(t) &= \lim_{s \rightarrow 0} \frac{s \frac{b}{J_p}}{s(s + \frac{b}{J_p}) + \left(\frac{b}{J_p}\right)(k_P s + k_I)} \frac{1}{s^q} \\ &= \lim_{s \rightarrow 0} \frac{\frac{b}{J_p}}{\frac{b}{J_p} k_I} \frac{1}{s^{q-1}} \\ &= \lim_{s \rightarrow 0} \frac{1}{k_I} \frac{1}{s^{q-1}}\end{aligned}$$

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

Chapter 12

Root Locus

12.1 Theory

We begin this chapter by noting that for the closed loop system shown in Figure 12.1, the transfer function from $Y^d(s)$ to $Y(s)$ can be computed as

$$\begin{aligned} Y(s) &= P(s)C(s)(Y^d(s) - Y(s)) \\ \implies Y(s) &= \frac{P(s)C(s)}{1 + P(s)C(s)}Y^d(s). \end{aligned}$$

Therefore, the characteristic equation of the closed loop system is given by

$$\Delta_{cl} = 1 + P(s)C(s) = 0.$$

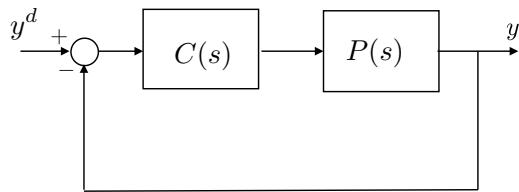


Figure 12.1: General feedback system with plant $P(s)$ and controller $C(s)$.

We have seen in Chapter 7 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 12.2. The closed-loop

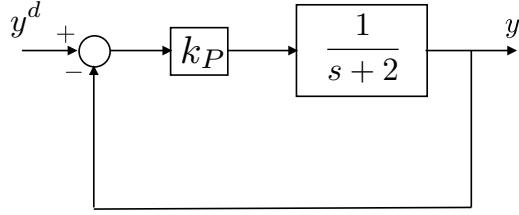


Figure 12.2: A first order plant under proportional control.

transfer function is given by

$$Y(s) = \frac{k_P}{s + (2 + k_P)} Y^d(s)$$

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 12.3. 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 12.3. 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 12.4. In this case the closed loop transfer function is given by

$$Y(s) = \frac{k_P}{s^2 + 2s + k_P} Y^d(s)$$

Note that the open loop poles are located at 0 and -2 . The closed loop characteristic polynomial is given by

$$\Delta_{cl}(s) = s^2 + 2s + k_P \quad (12.1)$$

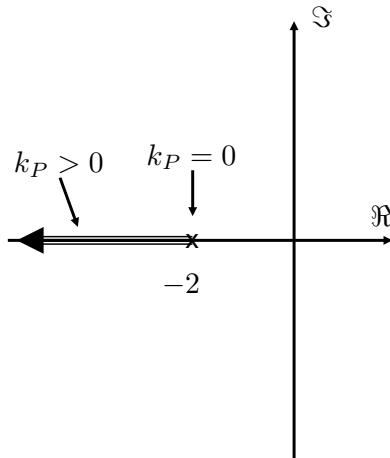


Figure 12.3: A plot of the closed loop pole associated with Figure 12.2 as a function of $k_P > 0$.

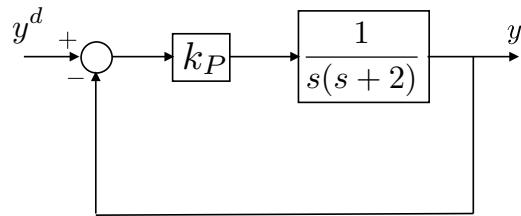


Figure 12.4: A second order plant under proportional control.

and the closed loop poles are located at

$$p_{cl} = -1 \pm \sqrt{1 - k_P}.$$

The position of the closed loop poles as a function of $k_P > 0$ are shown in Figure 12.5. 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 12.5.

The Matlab command `rlocus` will plot the root locus for complex system and can be used interactively to find the gain associated with specific pole locations. To use the `rlocus` command in Matlab, the characteristic

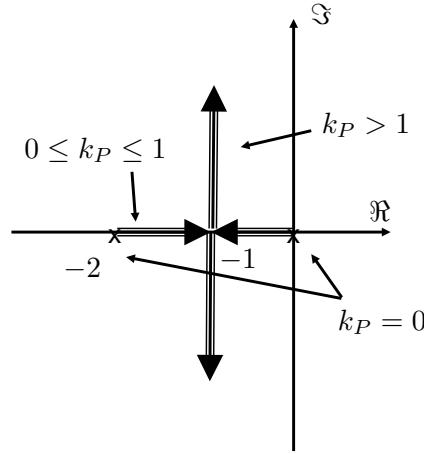


Figure 12.5: A plot of the closed loop pole associated with Figure 12.4 as a function of $k_P > 0$.

equation must be put in *Evan's form*, which is

$$1 + kL(s) = 0,$$

were k is the gain of interest. For example, the characteristic equation associated with the characteristic polynomial in Equation (12.1) is

$$\Delta_{cl}(s) = s^2 + 2s + k_P = 0.$$

The characteristic equation can be put in Evan's form by dividing both sides 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 12.6 so that the closed loop poles are located at $-2 \pm j2$, where $k_P = 8$ and $k_D = 2$. Now

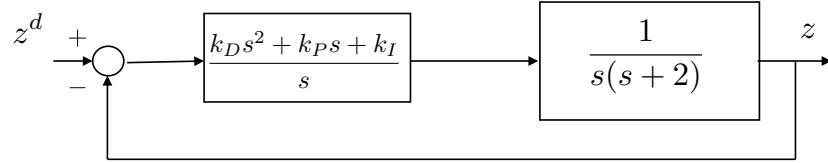


Figure 12.6: Closed loop system where the proportional gains have been selected as $k_P = 8$ and $k_D = 2$.

suppose that an integrator is to be added to 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 equation for the closed loop system including the integrator is given by

$$\begin{aligned}
 & 1 + \left(\frac{1}{s^2 + 2s} \right) \left(\frac{k_D s^2 + k_P s + k_I}{s} \right) = 0 \\
 \Rightarrow & s^3 + (2 + k_D)s^2 + k_P s + k_I = 0 \\
 \Rightarrow & 1 + k_I \left(\frac{1}{s^3 + (2 + k_D)s^2 + k_P s} \right) = 0 \\
 \Rightarrow & 1 + k_I \left(\frac{1}{s^3 + 4s^2 + 8s} \right) = 0,
 \end{aligned}$$

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 12.7. Note from Figure 12.7 that the 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 12.7 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 $-1.67 \pm j1.74$.

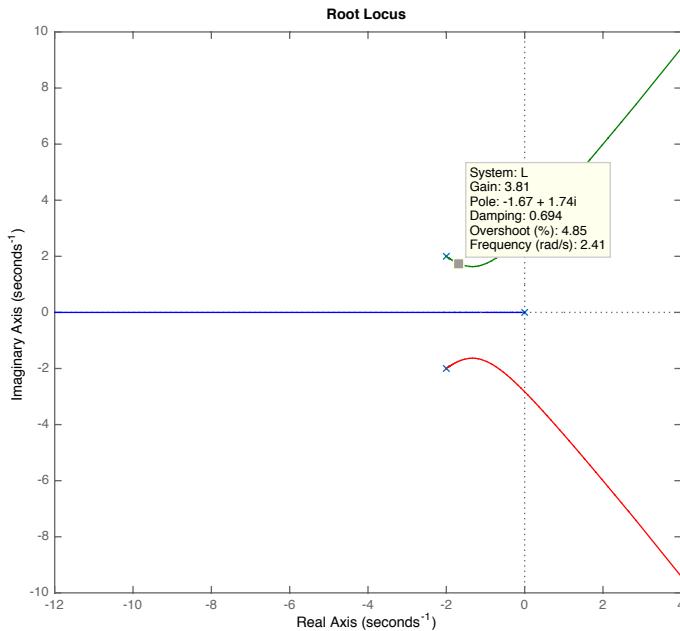
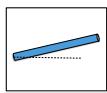


Figure 12.7: The result of Matlab's `rlocus` command, and by interactively selecting pole locations close to $-2 \pm j2$.

12.2 Design Study A. Single Link Robot Arm



Homework Problem A.14

For the single link robot arm, use the PD gains derived in HW A.12. Change the parameter file `param.m` so that the system parameters known to the controller are incorrect by upto $\pm 5\%$ of the actual parameters and note the associated steady state error. Add an integrator to the controller to obtain PID control. Put the resulting closed loop 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. Implement the controller in Simulink and further tune k_I to get satisfactory response.

Solution

The closed loop block diagram including an integrator is shown in Figure 11.8. The closed loop transfer function is given by

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

The characteristic equation is therefore

$$s^3 + \left(\frac{3b+3k_D}{m\ell^2}\right)s^2 + \frac{3k_P}{m\ell^2}s + \frac{3k_I}{m\ell^2} = 0.$$

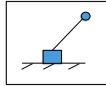
In Evan's form we have

$$1 + k_I \left(\frac{\frac{3}{m\ell^2}}{s^3 + \left(\frac{3b+3k_D}{m\ell^2}\right)s^2 + \frac{3k_P}{m\ell^2}s} \right) = 0.$$

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);
```

12.3 Design Study B. Inverted Pendulum



Homework Problem B.14

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 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. Implement the controller in Simulink and further tune k_I to get satisfactory response.

Solution

The closed loop block diagram for the outer loop including an integrator is shown in Figure 11.10. The closed loop transfer function is given by

$$Z(s) = \frac{\frac{m_1 g}{m_2} (k_{Dz} s^2 + k_{Pz} s + k_{Iz})}{s^3 + \left(\frac{b}{m_2} + \frac{m_1 g}{m_2} k_{Dz}\right) s^2 + \frac{m_1 g}{m_2} k_{Pz} s + \frac{m_1 g}{m_2} k_{Iz}} Z^d(s).$$

The characteristic equation is therefore

$$s^3 + \left(\frac{b}{m_2} + \frac{m_1 g}{m_2} k_{Dz}\right) s^2 + \frac{m_1 g}{m_2} k_{Pz} s + \frac{m_1 g}{m_2} k_{Iz} = 0.$$

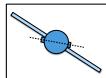
In Evan's form we have

$$1 + k_{Iz} \left(\frac{\frac{m_1 g}{m_2}}{s^3 + \left(\frac{b}{m_2} + \frac{m_1 g}{m_2} k_{Dz}\right) s^2 + \frac{m_1 g}{m_2} k_{Pz} s} \right) = 0.$$

The appropriate Matlab command is therefore

```
1 >> L = tf([m1*g/m2], [1, (b/m2+m1*g*kd/m2), m1*g*kp/m2, 0]);
2 >> figure(1), clf, rlocus(L);
```

12.4 Design Study C. Satellite Attitude Control



Homework Problem C.14

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. Implement the controller in Simulink and further tune k_I to get satisfactory response.

Solution

The closed loop block diagram for the outer loop including an integrator is shown in Figure 11.12. The closed loop transfer function is given by

$$\Phi(s) = \frac{\frac{b}{J_p}(k_{P_\phi}s + k_{D_\phi})}{s^2 + \left(\frac{b}{J_p} + \frac{bk_{P_\phi}}{J_p}\right)s + \frac{bk_{I_\phi}}{J_p}}\Phi^d(s).$$

The characteristic equation is therefore

$$s^2 + \left(\frac{b}{J_p} + \frac{bk_{P_\phi}}{J_p}\right)s + \frac{bk_{I_\phi}}{J_p} = 0.$$

In Evan's form we have

$$1 + k_{I_\phi} \left(\frac{\frac{b}{J_p}}{s^2 + \left(\frac{b}{J_p} + \frac{bk_{P_\phi}}{J_p}\right)s} \right) = 0.$$

The appropriate Matlab command is therefore

```

1 >> L = tf([b/Jp], [1, (b/Jp+b*kp/Jp), 0]);
2 >> figure(1), clf, rlocus(L);

```

Notes and References

Chapter 13

Digital Implementation of PID Controllers

13.1 Theory

13.2 Digital Implementation of PID Loops

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 $e(t) = y^d(t) - y(t)$ is the error between the desired output $y^d(t)$ 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 13.1 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

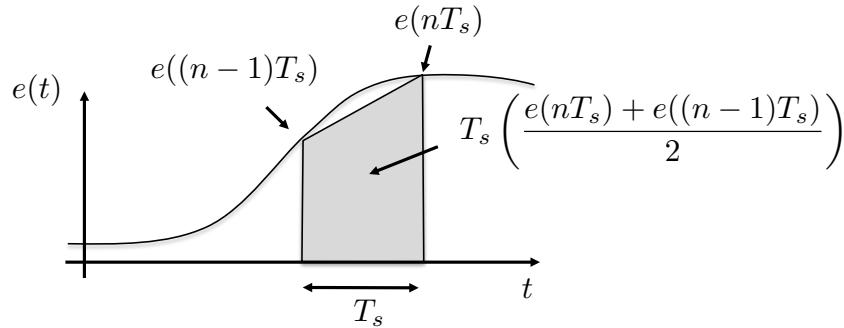


Figure 13.1: Trapezoidal rule for digital integration.

$$u_I(nT_s) \approx u_I((n-1)T_s) + \frac{T_s}{2} (e(nT_s) + e((n-1)T_s)).$$

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

$$U_I(z) = \frac{T_s}{2} \left(\frac{1+z^{-1}}{1-z^{-1}} \right) E(z). \quad (13.1)$$

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

$$s \mapsto \frac{2}{T_s} \left(\frac{1 - z^{-1}}{1 + z^{-1}} \right), \quad (13.2)$$

where T_s is the sample period [?]. Taking the inverse z -transform of Equation (13.1) gives the discrete time implementation of an integrator as

$$u_I[n] = u_I[n - 1] + \frac{T_s}{2} (e[n] + e[n - 1]). \quad (13.3)$$

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 a band-limited, or dirty derivative

$$U_D(s) = \frac{s}{\tau s + 1} E(s),$$

where τ is small, and $\frac{1}{\tau}$ defines the bandwidth of the differentiator. In practical terms, the dirty derivative differentiates signals with frequency content less than $\frac{1}{\tau}$ radians per second. Using the the Tustin approximation (13.2), the dirty derivative in the z -domain becomes

$$\begin{aligned} u_D(z) &= \frac{\frac{2}{T_s} \left(\frac{1 - z^{-1}}{1 + z^{-1}} \right)}{\frac{2\tau}{T_s} \left(\frac{1 - z^{-1}}{1 + z^{-1}} \right) + 1} E(z) \\ &= \frac{\left(\frac{2}{2\tau + T_s} \right) (1 - z^{-1})}{1 - \left(\frac{2\tau - T_s}{2\tau + T_s} \right) z^{-1}} E(z). \end{aligned}$$

Transforming to the time domain, we have

$$u_D[n] = \left(\frac{2\tau - T_s}{2\tau + T_s} \right) u_D[n - 1] + \left(\frac{2}{2\tau + T_s} \right) (e[n] - e[n - 1]). \quad (13.4)$$

Matlab code that implements a general PID loop is shown below.

```

1      % main PID loop
2      function u = pidloop(y_d,y,flag,kp,ki,kd,limit,Ts,tau)
3          persistent integrator;
4          persistent differentiator;
5          persistent error_d1;
6          % reset (initialize) persistent variables when flag==1
7          if flag==1,
8              integrator = 0;
9              differentiator = 0;
10             error_d1 = 0; % _d1 means delayed by one time step
11         end
12         % compute the current error
13         error = y_d - y;
14         % update integrator
15         integrator = integrator + (Ts/2)*(error + error_d1);
16         % update differentiator
17         a1 = (2*tau-Ts)/(2*tau+Ts);
18         a2 = 2/(2*tau+Ts);
19         differentiator = a1*differentiator+a2*(error-error_d1);
20         u = sat(... % implement PID control
21                 kp * error +... % proportional term
22                 ki * integrator +... % integral term
23                 kd * differentiator,... % derivative term
24                 limit... % ensure abs(u)<=limit
25             );
26         % implement integrator anti-windup
27         if ki≠0
28             u_unsat = kp*error+ki*integrator+kd*differentiator;
29             integrator = integrator + Ts/ki * (u - u_unsat);
30         end
31         % update the delayed error for next time through loop
32         error_d1 = error;
33     end % end pidloop
34
35     % function to saturate the output
36     function out = sat(in, limit)
37         if in > limit,           out = limit;
38         elseif in < -limit;    out = -limit;
39         else                   out = in;
40         end
41     end % end sat

```

The inputs on Line 1 are the commanded output y^d ; the current output y ; a flag used to reset the integrator; the PID gains k_P , k_I , and k_D ; the limit

of the saturation command; the sample time T_s ; and the time constant τ of the differentiator. Line 15 implements Equation (13.3), and Lines 17–19 implement Equation (13.4).

A potential problem with a straight-forward implementation of PID controllers is integrator wind up. When the error $y^d - y$ is large and a large error persists for an extended period of time, the value of the integrator, as computed in Line 15, can become large, or “wind up.” A large integrator will cause u , as computed in Line 20–25, 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. A particularly simple scheme, which is shown in Lines 26–30, 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 updating the integrator, where u_I^- is the value of the integrator before applying the anti-wind-up scheme, and let

$$u_{\text{unsat}}^+ = k_P e + k_D u_D + k_I u_I^+$$

denote the unsaturated control value after updating the integrator, where

$$u_I^+ = u_I^- + \Delta u_I,$$

and Δu_I is the update. The objective is to find Δu_I so that $u_{\text{unsat}}^+ = u$, where u is value of the control after the saturation command is applied. Noting that

$$u_{\text{unsat}}^+ = u_{\text{unsat}}^- + k_I \Delta u_I,$$

we can solve for Δu_I to obtain

$$\Delta u_I = \frac{1}{k_I} (u - u_{\text{unsat}}^-).$$

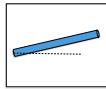
The multiplication by T_s in Line 29 is to account for the sampled-data implementation.

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 ?? is to be used, where $u_D(t) = -\frac{dy}{dt}$, then Line 19 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.

13.3 Design Study A. Single Link Robot Arm



Homework Problem A.15

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Change the parameter file param.m to specify actual parameters and the parameters known to the controller. Change the values for mass, length, and damping coefficient known to the controller by 5% to simulate imperfect knowledge of the plant.
- (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 desired angle θ^d .
- (c) Implement the PID controller designed in Problems A.14 using an m-function called arm_ctrl.m. Use the dirty derivative gain of $\tau = 0.05$. Compare the result with and without the integrator (set $k_I = 0$).

Solution

The solution is on the wiki page associated with the book.

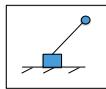
The Matlab code for the controller is shown below.

```

46    % update integral of error
47    if abs(thetadot)<0.05,
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    theta_d1 = theta;
53
54    % compute the pid control signal
55    u_unsat = kp*error + ki*integrator - kd*thetadot;
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 function out = sat(in,limit)
65     if      in > limit,      out = limit;
66     elseif in < -limit,     out = -limit;
67     else                  out = in;
68     end
69 end

```

13.4 Design Study B. Inverted Pendulum



Homework Problem B.15

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Change the parameter file param.m to specify actual parameters and the parameters known to the controller. Change the values for m_1 , m_2 , ℓ and b known to the controller by 5% to simulate imperfect knowledge of the plant.
- (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 desired position z^d .

- (e) Implement the nested PID loops designed in Problems B.14 using an m-function called pendulum_ctrl.m. Use the dirty derivative gain of $\tau = 0.05$.

Solution

The solution is on the wiki page associated with the book.

The Matlab code for the controller is shown below.

```

1 function F=pendulum_ctrl(in,P)
2     z_d    = 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,
11        flag = 1;
12    else
13        flag = 0;
14    end
15
16    % outer loop to compute desired angle
17    theta_d = PID_z(z_d,z,flag,P.kp_z,P.ki_z,P.kd_z, ...
18                    30*pi/180,P.Ts,P.tau);
19    % inner loop to compute the force
20    F = PD_th(theta_d,theta,flag,P.kp_th,P.kd_th,P.F_max, ...
21                P.Ts,P.tau);
22
23 end
24
25 %
26 % PID control for position
27 function u = PID_z(z_c,z,flag,kp,ki,kd,limit,Ts,tau)
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 integral of error
44 integrator = integrator + (Ts/2)*(error+error_d1);
45 % update derivative of z
46 zdot = (2*tau-Ts)/(2*tau+Ts)*zdot + 2/(2*tau+Ts)*(z-z_d1);
47 % update delayed variables for next time through the loop
48 error_d1 = error;
49 z_d1      = z;
50
51 % compute the pid control signal
52 u_unsat = kp*error + ki*integrator - kd*zdot;
53 u = sat(u_unsat,limit);
54
55 % integrator anti-windup
56 if ki≠0,
57     integrator = integrator + Ts/ki*(u-u_unsat);
58 end
59 end
60
61
62 %—————
63 % PD control for angle
64 function u = PD_th(theta_c,theta,flag,kp,kd,limit,Ts,tau)
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_c-theta;
76 % update derivative of y
77 thetadot = (2*tau-Ts)/(2*tau+Ts)*thetadot...
78             + 2/(2*tau+Ts)*(theta-theta_d1);
79 % update delayed variables for next time through the loop

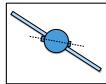
```

```

80     error_d1 = error;
81     theta_d1 = theta;
82
83     % compute the pid control signal
84     u_unsat = kp*error - kd*thetadot;
85     u = sat(u_unsat,limit);
86
87 end
88
89 %
90 % saturation function
91 function out = sat(in,limit)
92     if      in > limit,      out = limit;
93     elseif in < -limit,    out = -limit;
94     else                  out = in;
95     end
96 end

```

13.5 Design Study C. Satellite Attitude Control



Homework Problem C.15

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Change the parameter file param.m to specify actual parameters and the parameters known to the controller. Change the values for J_s , J_p , k and b known to the controller by 5%, to simulate imperfect knowledge of the plant.
- (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 desired angle ϕ^d .
- (c) Implement the nested PID loops designed in Problems C.14 using an m-function called satellite_ctrl.m. Use the dirty derivative gain of $\tau = 0.05$.

Solution

The solution is on the wiki page associated with the book.

Notes and References

A standard reference for digital implementation of PID controllers is [?].
Simple anti-wind-up schemes are discussed in [?, ?].

Part IV

Observer Based Control Design

RWB: Write an overview of what this part is all about, and how the chapters fit together.

Chapter 14

Full State Feedback

14.1 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

$$\dot{x} = Ax + Bu \quad (14.1)$$

$$y = Cx + Du, \quad (14.2)$$

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

$$u = -Kx + \nu, \quad (14.3)$$

where $K \in \mathbb{R}^{m \times n}$ is the feedback gains, and ν will be a signal that is specified later. Using (14.3) in Equation (14.1) results in the closed-loop state space equations

$$\dot{x} = Ax + B(-Kx + \nu) = (A - BK)x + B\nu \quad (14.4)$$

$$y = Cx + Du, \quad (14.5)$$

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 (14.1)–(14.2) is equivalent to the transfer function model given by

$$Y(s) = (C(sI - A)^{-1}B + D)U(s). \quad (14.6)$$

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

$$z = P^{-1}x, \quad (14.7)$$

where P is an invertible matrix. Then

$$\dot{z} = P^{-1}\dot{x} = P^{-1}(Ax + Bu) = P^{-1}APz + P^{-1}Bu \quad (14.8)$$

$$y = Cx + Du = CPz + Du \quad (14.9)$$

is an alternative state space representation of system (14.1)–(14.2). To show that these two representations have the same input-output behavior, or in other words, the same transfer function, note from Equation (14.6) that the transfer function of system (14.8)–(14.9) is

$$\begin{aligned} Y(s) &= (CP(sI - P^{-1}AP)^{-1}P^{-1}B + D)U(s) \\ &= (CP(sP^{-1}P - P^{-1}AP)^{-1}P^{-1}B + D)U(s) \\ &= (CP(P^{-1}(sI - A)P)^{-1}P^{-1}B + D)U(s) \\ &= (CPP^{-1}(sI - A)^{-1}PP^{-1}B + D)U(s) \\ &= (C(sI - A)^{-1}B + D)U(s), \end{aligned}$$

which is identical to Equation (14.6). Therefore Equations (14.1)–(14.2) and Equations (14.8)–(14.9) are realizations of the same system.

14.1.1 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

$$Y(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} U(s). \quad (14.10)$$

The block diagram of the system is shown in Figure 14.1.

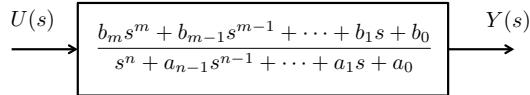


Figure 14.1: General transfer function for a SISO system.

The first step in deriving the control canonic state space equation is to artificially decompose the transfer function as shown in Figure 14.2, where the numerator polynomial and the denominator polynomial appear in separate cascaded blocks, and where $Z(s)$ is an artificial intermediate variable.

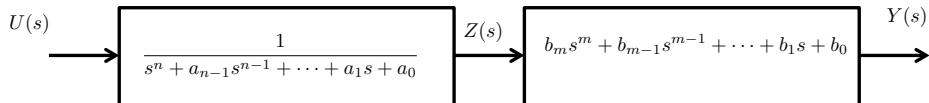


Figure 14.2: General transfer function for a SISO system decomposed into state dynamics and output dynamics.

In the s -domain, the equations corresponding to Figure 14.2 can be written as

$$\begin{aligned} Z(s) &= \frac{1}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} U(s) \\ Y(s) &= (b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0) Z(s). \end{aligned}$$

Taking the inverse Laplace transform of these equations gives

$$\frac{d^n z}{dt^n} = u(t) - a_{n-1} \frac{d^{n-1} z}{dt^{n-1}} - \cdots - a_1 \dot{z} - a_0 z(t) \quad (14.11)$$

$$y(t) = b_m \frac{d^m z}{dt^m} + b_{m-1} \frac{d^{m-1} z}{dt^{m-1}} + \cdots + b_1 \dot{z} + b_0 z(t). \quad (14.12)$$

The next step in the derivation of the state space equations is to draw the analog computer implementation of Equations (14.11) and (14.12), which is shown in Figure 14.3.

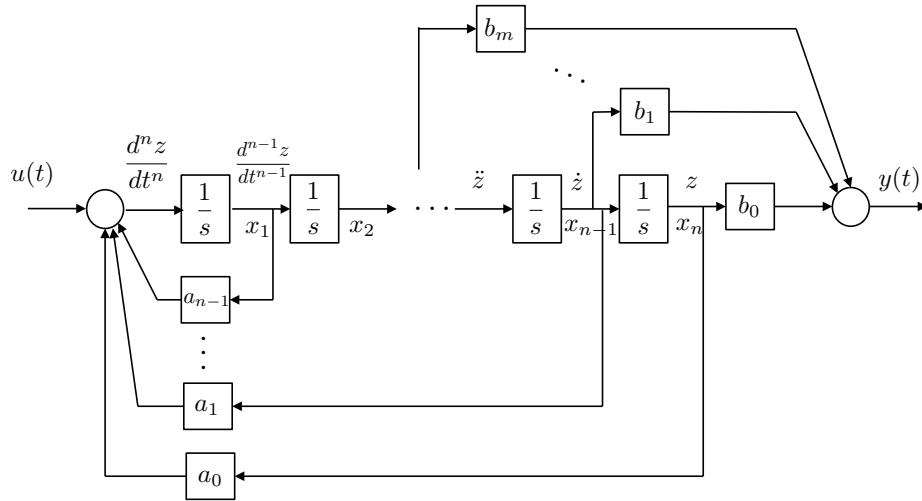


Figure 14.3: Block diagram showing the analog computer implementation of Equations (14.11) and (14.12).

Labeling the output of the integrators as the state variables x_1, \dots, x_n ,

as shown in Figure 14.3, we can write the state space equations as

$$\begin{aligned} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{pmatrix} &= \begin{pmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_1 & -a_0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} u \quad (14.13) \\ y &= (0 \ \dots \ 0 \ b_m \ \dots \ b_1 \ b_0) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + 0 \cdot u. \end{aligned}$$

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

$$A_c \triangleq \begin{pmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_1 & -a_0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix} \quad (14.14)$$

$$B_c \triangleq \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad (14.15)$$

$$C_c \triangleq (0 \ \dots \ 0 \ b_m \ \dots \ b_1 \ b_0). \quad (14.16)$$

Note that the negative of the coefficients of the denominator of (14.10) 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 (14.10) 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

$$\begin{aligned} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} &= \begin{pmatrix} -6 & 11 & -8 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u \quad (14.17) \\ y &= (0 \ 9 \ 20) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + 0 \cdot u. \end{aligned}$$

14.1.2 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 (14.14) and their characteristic polynomial

$$\Delta_{ol}(s) = s^n + a_{n-1}s^{n-1} + \cdots + a_1s + a_0.$$

Suppose that we let

$$u = -K_c x_c + \nu = -\begin{pmatrix} k_1 & k_2 & \dots & k_{n-1} & k_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} + \nu. \quad (14.18)$$

Substituting into Equation (14.13) 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

$$\Delta_{cl}(s) = s^n + (a_{n-1} + k_1)s^{n-1} + \cdots + (a_1 + k_{n-1})s + (a_0 + k_n).$$

If the desired characteristic polynomial is given by

$$\Delta_{cl}^d(s) = s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1s + \alpha_0, \quad (14.19)$$

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

$$K_c = \boldsymbol{\alpha} - \mathbf{a}_A, \quad (14.20)$$

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 (14.17). 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 $\mathbf{a}_A = (6, -11, 8)$, $\boldsymbol{\alpha} = (5, 12, 8)$. The feedback gains that place the poles at the desired locations are therefore given by

$$K_c = \boldsymbol{\alpha} - \mathbf{a}_A = (5, 12, 8) - (6, -11, 8) = (-1, 23, 0).$$

14.1.3 Full State Feedback: General Case

The formula given in Equation (14.20) 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 (14.7)–(14.9) that state space equations can be transformed using the change of variables in Equation (14.7) 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 that 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

$$\dot{x} = Ax + Bu, \quad (14.21)$$

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

$$x_c = P^{-1}x, \quad (14.22)$$

and

$$\dot{x}_c = P^{-1}APx_c + P^{-1}Bu = A_cx_c + B_cu, \quad (14.23)$$

where A_c and B_c have the form given by Equations (14.14) and (14.15), respectively. Therefore, we need to find P so that $P^{-1}B = B_c$, and $P^{-1}AP = A_c$, 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 is the i^{th} column of P . The requirement that $B = PB_c$ implies that

$$B = (\mathbf{p}_1 \ \mathbf{p}_2 \ \dots \ \mathbf{p}_n) \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \mathbf{p}_1.$$

Therefore, $\mathbf{p}_1 = B$. The requirement that $AP = PA_c$ implies that

$$(\mathbf{p}_1 \ \mathbf{p}_2 \ \dots \ \mathbf{p}_n) \begin{pmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_1 & -a_0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \vdots & 1 & 0 \end{pmatrix} = A (\mathbf{p}_1 \ \mathbf{p}_2 \ \dots \ \mathbf{p}_n).$$

After multiplication, each column can be equated to get

$$-a_{n-1}\mathbf{p}_1 + \mathbf{p}_2 = A\mathbf{p}_1 \quad (14.24)$$

$$-a_{n-2}\mathbf{p}_1 + \mathbf{p}_3 = A\mathbf{p}_2 \quad (14.25)$$

$$\vdots \quad (14.26)$$

$$-a_1\mathbf{p}_1 + \mathbf{p}_n = A\mathbf{p}_{n-1}. \quad (14.27)$$

Equation (14.24), and the previous conclusion that $\mathbf{p}_1 = B$, implies that

$$\mathbf{p}_2 = a_{n-1}B + AB.$$

Using this results in Equation (14.25) gives

$$\mathbf{p}_3 = a_{n-2}B + a_{n-1}AB + A^2B.$$

We can continue to solve for each column of P until the last column, which from Equation (14.27) 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

$$\mathcal{C}_{A,B} = (B \ AB \ A^2B \ \dots \ A^{n-1}B), \quad (14.28)$$

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

$$\mathcal{A}_A = \begin{pmatrix} 1 & a_{n-1} & a_{n-2} & \dots & a_2 & a_1 \\ 0 & 1 & a_{n-1} & \vdots & a_3 & a_2 \\ 0 & 0 & 1 & \vdots & a_4 & a_3 \\ \vdots & & & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & 1 & a_{n-1} \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix}, \quad (14.29)$$

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} is an upper triangular matrix with ones on the diagonal, that $\det(\mathcal{A}) = 1$, implying that \mathcal{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 (14.28) 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 (14.21) is controllable, that there exists an invertible state transformation matrix P so that the change of variables in Equation (14.22) is well defined and results in the transformed equations being in control canonic form via Equations (14.23).

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 (14.22) the state feedback control can be expressed in the original states x as

$$\begin{aligned} u &= -(\boldsymbol{\alpha} - \mathbf{a}_A)P^{-1}x + \nu \\ &= -(\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1}x + \nu \\ &= -Kx + \nu, \end{aligned} \quad (14.30)$$

where

$$K = (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1}. \quad (14.31)$$

Equation (14.31) 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 (14.30) is the state equations (14.1)-(14.2) results in the closed loop system

$$\begin{aligned}\dot{x} &= (A - BK)x + B\nu \\ y &= Cx,\end{aligned}$$

where we have assumed that $D = 0$. The resulting transfer function from ν to y is given by

$$Y(s) = (C(sI - (A - BK))^{-1}B)\nu(s). \quad (14.32)$$

A block diagram of the resulting closed loop system is shown in Figure 14.4 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 14.4. The feedforward gain k_r is selected so that the DC-gain from r to y is equal to one.

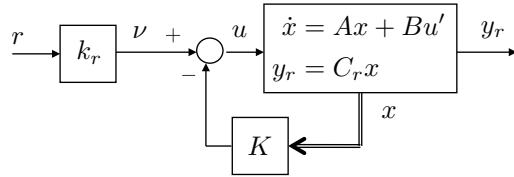


Figure 14.4: Full state feedback with reference input r .

From Equation (14.32), the transfer function from R to Y is given by

$$Y(s) = (C(sI - (A - BK))^{-1}Bk_r)R(s), \quad (14.33)$$

and the DC-gain is

$$k_{DC} = \lim_{s \rightarrow 0} [C(sI - (A - BK))^{-1}Bk_r] = -C(A - BK)^{-1}Bk_r. \quad (14.34)$$

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 some problems, the output measures by 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

$$k_r = -\frac{1}{C_r(A - BK)^{-1}B}. \quad (14.35)$$

Chapter 16 will describe observers that use the sensed output y_r to reconstruct the state x .

The feedforward gain computed in Equation (14.35) is for single input system. More generally, if the size of the reference input 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 $B \in \times \times \gg$, which implies that the transfer matrix in Equation (14.33) is square and has dimension $m \times m$. Similarly, the gain $K_r \in \gg \times \gg$ will also be square, and from Equation (14.34) will be equal to

$$k_r = -[C_r(A - BK)^{-1}B]^{-1}.$$

14.1.4 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

$$\tilde{u} = -K\tilde{x} + k_r\tilde{r} \quad (14.36)$$

where $\tilde{r} = r - y_{re}$ is the reference input around the equilibrium output, and where $\text{eig}(A - BK)$ are in the left half plane, and the DC-gain from \tilde{r} to \tilde{y}_r is one. The controller input to the plant is therefore

$$u = u_e - K(x - x_e) + k_r(r - y_{re}). \quad (14.37)$$

14.2 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

$$\mathcal{A}_A = \begin{pmatrix} 1 & a_{n-1} & a_{n-2} & \cdots & a_2 & a_1 \\ 0 & 1 & a_{n-1} & \cdots & a_3 & a_2 \\ \dots & & \ddots & & & \vdots \\ 0 & 0 & \cdots & 0 & 1 & a_{n-1} \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$$

Step 3. Find the desired closed loop polynomial

$$\Delta_{cl}^d(s) = (s - p_1)(s - p_2) \cdots (s - p_n) = s^2 + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1s + \alpha_0,$$

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);
```

14.2.1 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

$$\Delta_{ol}(s) = \det(sI - A) = \det \begin{pmatrix} s & -1 \\ -2 & s-3 \end{pmatrix} = s^2 - 3s - 2,$$

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

$$\Delta_{cl}^d(s) = (s+1-j)(s+1+j) = s^2 + 2s + 2,$$

which implies that

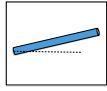
$$\boldsymbol{\alpha} = (2, 2).$$

Step 4. The gains are therefore given as

$$\begin{aligned}K &= (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1} \\ &= ((2, 2) - (-3, -2)) \begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -\frac{3}{4} & \frac{1}{4} \\ \frac{1}{4} & 0 \end{pmatrix} \\ &= (5 \quad 4) \begin{pmatrix} 0 & \frac{1}{4} \\ \frac{1}{4} & 0 \end{pmatrix} \\ &= (1 \quad \frac{5}{4})\end{aligned}$$

$$\begin{aligned}
k_r &= \frac{-1}{C_r(A - BK)^{-1}B} \\
&= \frac{-1}{(5 \ 0) \left(\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} - \begin{pmatrix} 0 \\ 4 \end{pmatrix} \begin{pmatrix} 1 & \frac{5}{4} \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 4 \end{pmatrix} \right)} \\
&= \frac{-1}{(5 \ 0) \left(\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 4 & 5 \end{pmatrix} \right)^{-1} \begin{pmatrix} 0 \\ 4 \end{pmatrix}} \\
&= \frac{-1}{(5 \ 0) \begin{pmatrix} 0 & 1 \\ -2 & -2 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 4 \end{pmatrix}} \\
&= \frac{-1}{-10} = \frac{1}{10}.
\end{aligned}$$

14.3 Design Study A. Single Link Robot Arm



Homework Problem A.16

The objective of this problem is to implement state feedback controller using the full state. Start with the simulation files developed in Homework A.15.

- (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.12.
- (b) Add the state space matrices A , B , C , D derived in Homework A.7 to your param file.
- (e) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}) = n$.
- (c) Assuming that the parameters known to the controller are the true parameters, using the Matlab `place` command, 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 θ^c 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.12. Why?

- (d) Modify the control code `arm_ctrl.m` 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.7, the state space equations for the single link robot arm are given by

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 1.0000 \\ 0 & -0.667 \end{pmatrix} x + \begin{pmatrix} 0 \\ 66.667 \end{pmatrix} u \\ y &= (1 \ 0) x.\end{aligned}$$

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

$$\Delta_{ol}(s) = \det(sI - A) = \det \begin{pmatrix} s & -1 \\ 0 & s + 0.667 \end{pmatrix} = s^2 + 0.667s,$$

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

$$\Delta_{cl}^d(s) = s^2 + 2\zeta\omega_n s + \omega_n^2 = s^2 + 6.3621s + 20.2445$$

which implies that

$$\boldsymbol{\alpha} = (6.3621, 20.2445).$$

Step 4. The gains are therefore given as

$$\begin{aligned} K &= (\alpha - \mathbf{a}_A) \mathcal{A}_A^{-1} \mathcal{C}_{A,B}^{-1} \\ &= (0.3037 \quad 0.0854) \end{aligned}$$

$$\begin{aligned} k_r &= \frac{-1}{C(A - BK)^{-1}B} \\ &= 0.3037. \end{aligned}$$

Alternatively, we could have used the following Matlab script

```

1 % state space model
2 A = [0, 1; 0, -3*P.b/P.m/(P.ell^2)];
3 B = [0; 3/P.m/(P.ell^2) ];
4 C = [1, 0];
5
6 % gains for pole locations
7 charpoly = [1, 2*zeta*wn, wn^2];
8 des_poles = roots(charpoly);
9
10 % is the system controllable?
11 if rank ctrb(A,B) ≠ 2,
12     disp('System Not Controllable');
13 else
14     P.K = place(A,B,des_poles);
15     P.kr = -1/(C*inv(A-B*P.K)*B);
16 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;

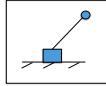
```

```

12     theta_d1    = 0;
13 end
14 thetadot = (2*P.tau-P.Ts) / (2*P.tau+P.Ts)*thetadot...
15     + 2/(2*P.tau+P.Ts)*(theta-theta_d1);
16 theta_d1 = theta;
17
18 % construct the state
19 x = [theta; thetadot];
20 % compute equilibrium torque tau_e
21 tau_e = P.m*P.g*(P.ell/2)*cos(theta);
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

```

14.4 Design Study B. Inverted Pendulum



Homework Problem B.16

The objective of this problem is to implement state feedback controller using the full state. Start with the simulation files developed in Homework B.15.

- (a) Using the values for ω_{n_z} , ζ_z , ω_{n_θ} , and ζ_θ selected in Homework B.12, find the desired closed loop poles.
- (b) Add the state space matrices A , B , C , D derived in Homework B.7 to your param file.
- (c) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}) = n$.
- (d) Assuming that the parameters known to the controller are the true parameters, using the Matlab `place` command, 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^c 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.

Solution

From HW B.7, the state space equations for the inverted pendulum are given by

$$\dot{x} = \begin{pmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0 & -2.4500 & -0.0500 & 0 \\ 0 & 24.5000 & 0.1000 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 1 \\ -2 \end{pmatrix} u$$

$$y = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} x,$$

but the desired reference output is the position z , or

$$y_r = (1 \ 0 \ 0 \ 0) x.$$

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 $\det(\mathcal{C}_{A,B}) = -1536 \neq 0$, therefore the system is controllable.

Step 2. The open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A) = s^4 + 0.0500s^3 - 24.5000s^2 - 0.9800s$$

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

$$\begin{aligned}\Delta_{cl}^d(s) &= (s^2 + 2\zeta_\theta \omega_{n_\theta} s + \omega_{n_\theta}^2)(s^2 + 2\zeta_z \omega_{n_z} s + \omega_{n_z}^2) \\ &= s^4 + 4.4280s^3 + 9.5248s^2 + 9.2280s + 4.0000\end{aligned}$$

which implies that

$$\boldsymbol{\alpha} = (4.4280, 9.5248, 9.2280, 4.0000).$$

Step 4. The gains are therefore given as

$$\begin{aligned}K &= (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1} \\ &= (-0.2041 - 17.1144 - 0.5208 - 2.4494)\end{aligned}$$

To compute the reference gain $k_r = -1/C_r(A - BK)^{-1}B$, we need to use C_r , the output matrix matching the reference input, which since the desired reference input is z_d and since $x = (z, \theta, \dot{z}, \dot{\theta})^\top$, we have $C_r = (1, 0, 0, 0)$, which gives

$$\begin{aligned}k_r &= \frac{-1}{C_r(A - BK)^{-1}B} \\ &= -0.2041.\end{aligned}$$

Alternatively, we could have used the following Matlab script

```

1 % state space model
2 A = [ ...;
3     0, 0, 1, 0; ...
4     0, 0, 0, 1; ...
5     0, -P.m1*P.g/P.m2, -P.b/P.m2, 0; ...
6     0, (P.m1+P.m2)*P.g/P.m2/P.ell, P.b/P.m2/P.ell, 0; ...
7 ];
8 B = [0; 0; 1/P.m2; -1/P.m2/P.ell];
9 C = [ ...;
10    1, 0, 0, 0; ...
11    0, 1, 0, 0; ...
12 ];
13
14 % gains for pole locations
15 wn_th = 2;
```

```

16 zeta_th = 0.707;
17 wn_z     = 1;
18 zeta_z   = 0.8;
19 char_poly = conv([1,2*zeta_z*wn_z,wn_z^2],...
20                 [1,2*zeta_th*wn_th,wn_th^2]);
21 des_poles = roots(char_poly);
22
23 % is the system controllable?
24 if rank(ctrb(A,B))≠4, disp('System Not Controllable'); end
25 P.K = place(A,B,des_poles);
26 P.kr = -1/(C(1,:)*inv(A-B*P.K)*B);

```

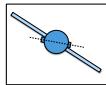
The Matlab code for the controller is given by

```

1 function F=pendulum_ctrl(in,P)
2     z_d    = 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.tau-P.Ts)/(2*P.tau+P.Ts)*zdot...
20        + 2/(2*P.tau+P.Ts)*(z-z_d1);
21    thetadot = (2*P.tau-P.Ts)/(2*P.tau+P.Ts)*thetadot...
22        + 2/(2*P.tau+P.Ts)*(theta-theta_d1);
23    z_d1 = z;
24    theta_d1 = theta;
25
26    % construct the state
27    x = [z; theta; zdot; thetadot];
28    % compute the state feedback controller
29    F = sat( -P.K*x + P.kr*z_d, P.F_max);
30 end

```

14.5 Design Study C. Satellite Attitude Control



Homework Problem C.16

The objective of this problem is to implement state feedback controller using the full state. Start with the simulation files developed in Homework C.15.

- (a) Using the values for ω_{n_ϕ} , ζ_ϕ , ω_{n_θ} , and ζ_θ selected in Homework C.12, find the desired closed loop poles.
- (b) Add the state space matrices A , B , C , D derived in Homework C.7 to your param file.
- (c) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}) = n$.
- (d) Assuming that the parameters known to the controller are the true parameters, using the Matlab `place` command, 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 ϕ^d to ϕ 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.

Solution

From HW C.7, the state space equations for the satellite are given by

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ -0.0300 & 0.0300 & -0.0100 & 0.0100 \\ 0.1500 & -0.1500 & 0.0500 & -0.0500 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 0.2 \\ 0 \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix} x.\end{aligned}$$

Step 1. The controllability matrix is therefore

$$\mathcal{C}_{A,B} = [B, AB, A^2B, A^3B] = \begin{pmatrix} 0 & 0.2000 & -0.0020 & -0.0059 \\ 0 & 0 & 0.0100 & 0.0294 \\ 0.2000 & -0.0020 & -0.0059 & 0.0007 \\ 0 & 0.0100 & 0.0294 & -0.0036 \end{pmatrix}.$$

The determinant is $\det(\mathcal{C}_{A,B}) = -36,000 \neq 0$, therefore the system is controllable.

Step 2. The open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A) = s^4 + 0.0600s^3 + 0.18s^2$$

which implies that

$$\mathbf{a}_A = (0.06, 0.18, 0, 0)$$

$$\mathcal{A}_A = \begin{pmatrix} 1 & 0.06 & 0.18 & 0 \\ 0 & 1 & 0.06 & 0.18 \\ 0 & 0 & 1 & 0.06 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Step 3. The desired closed loop polynomial

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

$$= s^4 + 4.9275s^3 + 12.1420s^2 + 14.6702s + 8.8637,$$

when $\omega_\theta = 1.9848$, $\zeta_\theta = 0.707$, $\omega_\phi = 1.5$, $\zeta_\phi = 0.707$, which implies that

$$\boldsymbol{\alpha} = (4.9275, 12.1420, 14.6702, 8.8637).$$

Step 4. The gains are therefore given as

$$K = (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1}$$

$$= (40.2842, 255.1732, 24.3375, 366.1825)$$

To compute the reference gain $k_r = -1/C_r(A - BK)^{-1}B$, we need to use C_r , the output matrix matching the reference input, which since the desired reference input is ϕ_d and since $x = (\theta, \phi, \dot{\theta}, \dot{\phi})^\top$, we have $C_r = (0, 1, 0, 0)$, which gives

$$k_r = \frac{-1}{C_r(A - BK)^{-1}B}$$

$$= 295.4573.$$

Alternatively, we could have used the following Matlab script

```

1 % state space design
2 A = [ ...
3     0, 0, 1, 0; ...
4     0, 0, 0, 1; ...
5     -P.k/P.Js, P.k/P.Js, -P.b/P.Js, P.b/P.Js; ...
6     P.k/P.Jp, -P.k/P.Jp, P.b/P.Jp, -P.b/P.Jp; ...
7
8 ];
9 B = [0; 0; 1/P.Js; 0];
10 C = [ ...
11     1, 0, 0, 0; ...
12     0, 1, 0, 0; ...
13 ];
14
15
16 % gains for pole locations
17 wn_th = 2*0.9924;
18 zeta_th = 0.707;
19 wn_phi = 1*1.5;
20 zeta_phi = 0.707;
21 ol_char_poly = charpoly(A);
22 des_char_poly = conv([1,2*zeta_th*wn_th,wn_th^2],...
23 [1,2*zeta_phi*wn_phi,wn_phi^2]);
24 des_poles = roots(des_char_poly);
25
26 % is the system controllable?
27 if rank ctrb(A,B) ≠ 4, disp('System Not Controllable'); end
28 P.K = place(A,B,des_poles);
29 P.kr = -1/([0, 1, 0, 0]*inv(A-B*P.K)*B);

```

The Matlab code for the controller is given by

```

1 function tau=satellite_ctrl(in,P)
2     phi_d    = in(1);
3     phi      = in(2);
4     theta    = 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.tau-P.Ts)/(2*P.tau+P.Ts)*phidot...
20     + 2/(2*P.tau+P.Ts)*(phi-phi_d1);
21 thetadot = (2*P.tau-P.Ts)/(2*P.tau+P.Ts)*thetadot...
22     + 2/(2*P.tau+P.Ts)*(theta-theta_d1);
23 phi_d1 = phi;
24 theta_d1 = theta;
25
26 % construct the state
27 x = [phi; theta; phidot; thetadot];
28 % compute the state feedback controller
29 tau = sat( -P.K*x + P.kr*phi_d, P.taumax);
30 end

```

Notes and References

RWB: talk more generally about controllability. More general definition (move origin to any state) -> equivalent for LTI system. For multi-input system, the controllability matrix needs to be full rank. Generalizing Ackermann's formula to multi-input case. etc.

Chapter 15

Integrator with Full State Feedback

15.1 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 11 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 14.

A block diagram of the system with full state feedback and an integrator is shown in Figure 15.1.

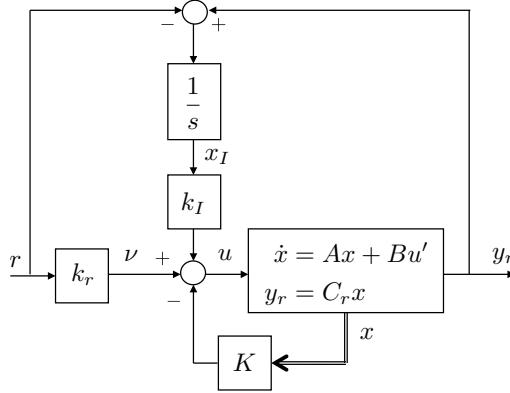


Figure 15.1: Full state feedback with integrator. Note that the error into the integrator is defined as $e = y - r$ which is the negative of the error defined in Part III.

Let the state of the integrator be defined as x_I , as shown in Figure 15.1, then

$$\dot{x}_I = y_r - r = C_r x - r.$$

In Chapter 12 we argued using root locus, that adding an integrator after 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 systems, 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 14 was not limited to second order systems. As such, there should be a way to use the methods of Chapter 14 to simultaneously place both the system poles and the integrator poles. 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

$$\begin{pmatrix} \dot{x} \\ \dot{x}_I \end{pmatrix} = \begin{pmatrix} A & \mathbf{0} \\ C & \mathbf{0} \end{pmatrix} \begin{pmatrix} x \\ x_I \end{pmatrix} + \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} u + \begin{pmatrix} 0 \\ -1 \end{pmatrix} r,$$

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

$$\mathcal{C}_{A_1, B_1} = \begin{pmatrix} B & AB & A^2B & \dots & A^{n-1}B & A^nB \\ 0 & C_r B & C_r AB & \dots & C_r A^{n-2}B & C_r A^{n-1}B \end{pmatrix},$$

which will have rank $n+1$ if one of the following conditions is true:

$$\begin{aligned} C_r B &\neq 0 \\ C_r AB &\neq 0 \\ &\vdots \\ C_r A^{n-1}B &\neq 0. \end{aligned}$$

When the augmented system is controllable, we can use the pole placement technique described in Chapter 14 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

$$\begin{aligned} u(t) &= -K_1 \begin{pmatrix} x \\ x_I \end{pmatrix} + k_r r = -\begin{pmatrix} K & k_I \end{pmatrix} \begin{pmatrix} x \\ x_I \end{pmatrix} + k_r r = -Kx - k_I x_I + k_r r \\ &= \underbrace{-Kx(t)}_{\text{state feedback}} - \underbrace{k_I \int_0^t (y_r(\tau) - r(\tau)) d\tau}_{\text{integral feedback}} + \underbrace{k_r r(t)}_{\text{feedforward term}}. \end{aligned}$$

15.1.1 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 $\tilde{x} = x - x_e$ 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 $y_r \in \mathbb{R}^p$ is the reference output, $y_{re} = C_r x_e \in \mathbb{R}^p$ 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 $\tilde{x}_I = \int(\tilde{y}_r - \tilde{r})d\tau$, 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

$$\tilde{u} = -K\tilde{x} + k_r\tilde{r} + k_I \int_{-\infty}^t \tilde{e}(\tau)d\tau.$$

Noting that

$$\tilde{e} = \tilde{y} - \tilde{r} = (y_r - y_{re}) - (r - y_{re}) = y_r - r$$

we get that the controller for the linearized system is given by

$$u = u_e - K(x - x_e) + k_r(r - y_{re}) + k_I \int_{-\infty}^t (y_r(\tau) - r(\tau))d\tau.$$

15.2 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

$$\dot{x} = Ax + Bu$$

$$y_r = C_r x,$$

where we desire that y_r tracks the reference r using integral action using the controller

$$u(t) = -Kx(t) + k_r r(t) + k_I \int_{-\infty}^t (y_r(\tau) - r(\tau))d\tau,$$

with desired closed loop gains at $p_1, p_2, \dots, p_n, p_I$, and where k_r ensures that the DC-gain is equal to one.

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

$$\begin{aligned} A_1 &= \begin{pmatrix} A & \mathbf{0} \\ C_r & \mathbf{0} \end{pmatrix} \\ B_1 &= \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} \end{aligned}$$

Step 2. Follow the pole placement procedure outlined in Chapter 14 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

$$\begin{aligned} K &= K_1(1 : n) \\ k_I &= K_1(n + 1). \end{aligned}$$

Step 4. Find the reference gain k_r using

$$k_r = \frac{-1}{C_r(A - BK)^{-1}B}.$$

where A , B , K , and C_r , are for the original, and not the augmented system.

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)+1,
8     disp('The system is controllable')
9 else
10    % place the poles at p1,p2,...,pn, pI
11    K1 = place(A,B,p1,p2,p3,p4,pI,pI);
12    % extract the state feedback and integral gains from KK
13    K = K1(1:n);
14    kI = K1(n+1);
15    % compute the reference gain
16    kr = -1/(Cr*inv(A-B*K)*B);
17 end

```

15.2.1 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, and add an integrator with gain at $p_I = -0.1$.

Step 1. Form the augmented system

$$\begin{aligned}A_1 &= \begin{pmatrix} A & \mathbf{0} \\ C_r & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 3 & 0 \\ 1 & 0 & 0 \end{pmatrix} \\ B_1 &= \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix}\end{aligned}$$

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

$$\begin{aligned} K_1 &= (\boldsymbol{\alpha} - \mathbf{a}_{A_1}) \mathcal{A}_{A_1, B_1}^{-1} \\ &= ((2.1, 2.2, 0.2) - (-3, -2, 0)) \begin{pmatrix} 1 & -3 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 4 & 12 \\ 4 & 12 & 44 \\ 0 & 0 & 4 \end{pmatrix}^{-1} \\ &= (1.05, 1.275, 0.05) \end{aligned}$$

Step 3. The feedback gains are therefore given by

$$\begin{aligned} K &= K_1(1 : 2) = (1.05, 1.275) \\ k_I &= K_1(3) = 0.05 \end{aligned}$$

Step 4. The reference gain is given by

$$\begin{aligned} k_r &= \frac{-1}{C_r(A - BK)^{-1}B} \\ &= \frac{-1}{(5 \ 0) \left(\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} - \begin{pmatrix} 0 \\ 4 \end{pmatrix} (1.05 \ 1.275) \right)^{-1} \begin{pmatrix} 0 \\ 4 \end{pmatrix}} \\ &= 0.11. \end{aligned}$$

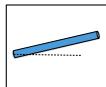
Alternatively, we could have used the following Matlab script

```

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     % compute the reference gain
19     kr = -1/(Cr*inv(A-B*K)*B);
20 end

```

15.3 Design Study A. Single Link Robot Arm



Homework Problem A.17

- (a) Modify the state feedback solution developed in Homework A.16 to add an integrator with anti-windup.
- (b) Change the parameters known to the controller by $\pm 5\%$.
- (c) Tune the integrator pole to get good tracking performance.

Solution

Step 1. The original state space equations are

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 1.0000 \\ 0 & -0.667 \end{pmatrix} x + \begin{pmatrix} 0 \\ 66.667 \end{pmatrix} u \\ y &= (1 \ 0) x,\end{aligned}$$

therefore the augmented system is

$$\begin{aligned}A_1 &= \begin{pmatrix} A & \mathbf{0} \\ C & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 & 1.0000 & 0 \\ 0 & -0.667 & 0 \\ 1 & 0 & 0 \end{pmatrix} \\ B_1 &= \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 \\ 66.667 \\ 0 \end{pmatrix}\end{aligned}$$

Step 2. After the design in HW A.16, the closed loop poles were located at $p_{1,2} = -3.1191 \pm j3.1200$. We will add the integrator pole at $p_I = -0.5$. 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

$$\Delta_{cl}^d(s) = (s + 3.1191 - j3.12)(s + 3.1191 + j3.12)(s + 0.5)$$

$$= s^3 + 6.7381s^2 + 22.5822s + 9.7315,$$

which implies that

$$\boldsymbol{\alpha} = (6.7381, 22.5822, 9.7315).$$

The augmented gains are therefore given as

$$K_1 = (\boldsymbol{\alpha} - \mathbf{a}_{A_1})\mathcal{A}_{A_1}^{-1}\mathcal{C}_{A_1, B_1}^{-1}$$

$$= (0.3548, 0.0962, 0.1529)$$

Step 3. The feedback gains are therefore given by

$$K = K_1(1 : 2) = (0.3548, 0.0962)$$

$$k_I = K_1(3) = 0.1529$$

Step 4. The reference gain is given by

$$k_r = \frac{-1}{C(A - BK)^{-1}B} = 0.3548.$$

Alternatively, we could have used the following Matlab script

```

1 % original state space system
2 A = [...
3     0, 1; ...
4     0, -3*P.b/P.m/(P.ell^2); ...
5 ];
6 B = [0; 3/P.m/(P.ell^2)];

```

```

7 C = [...
8     1, 0;...
9 ];
10 % form augmented system
11 A1 = [A, zeros(2,1); C, 0];
12 B1 = [B; 0];
13
14 % desired pole locations
15 wn = 4.4117;
16 zeta = 0.707;
17 des_char_poly = conv([1,2*zeta*wn,wn^2],poly(-.5));
18 des_poles = roots(des_char_poly);
19
20 % is the system controllable?
21 if rank ctrb(A1,B1) ≠ 3,
22     disp('System Not Controllable');
23 else % if so, compute gains
24     K1 = place(A1,B1,des_poles);
25     P.K = K1(1:2);
26     P.ki = K1(3);
27     P.kr = -1/(C*inv(A-B*P.K)*B);
28 end

```

Matlab code that implements the associated controller listed below.

```

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 = (2*P.tau-P.Ts)/(2*P.tau+P.Ts)*thetadot...
15        + 2/(2*P.tau+P.Ts)*(theta-theta_d1);
16    theta_d1 = theta;
17
18    % integrator
19    error = theta_c - theta

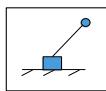
```

```

20 persistent integrator
21 persistent error_d1
22 if t<P.Ts==1,
23     integrator = 0;
24     error_d1 = 0;
25 end
26 if abs(thetadot)<0.05,
27     integrator = integrator + (P.Ts/2)*(error+error_d1);
28 end
29 error_d1 = error;
30
31 % construct the state
32 x = [theta; thetadot];
33 % compute equilibrium torque tau_e
34 tau_e = P.m*P.g*(P.ell/2)*cos(theta);
35 % compute the state feedback controller
36 tau_tilde = - P.K*x + P.kr*theta_c + P.ki*integrator;
37 % compute total torque
38 tau = sat( tau_e + tau_tilde, P.tau_max);
39
40 % integrator anti-windup
41 if P.ki≠0,
42     tau_unsat = tau_e + tau_tilde;
43     integrator = integrator + P.Ts/P.ki*(tau-tau_unsat);
44 end
45 end

```

15.4 Design Study B. Inverted Pendulum



Homework Problem B.17

- (a) Modify the state feedback solution developed in Homework B.16 to add an integrator with anti-windup to the position feedback.
- (b) Change the parameters known to the controller by $\pm 5\%$.
- (c) Add a constant input disturbance of 0.5 Newtons to the input of the plant and verify that when the integrator gain is set to zero, there is a steady state tracking error.

- (c) Tune the integrator to get zero steady state tracking performance. Note that the integrator gain will be negative.

Solution

Step 1. The original state space equations are

$$\begin{aligned}\dot{x} &= \begin{pmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0 & -2.2167 & -0.0500 & 0 \\ 0 & 24.5238 & 0.1020 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 0.9524 \\ -1.9436 \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} x.\end{aligned}$$

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.16, 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 integrator pole at $p_I = -0.5$. 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

$$\begin{aligned}\Delta_{cl}^d(s) &= (s + 1.4140 - j1.4144)(s + 1.4140 + j1.4144) \dots \\ &\quad (s + 0.8 - j0.6)(s + 0.8 + j0.6)(s + 0.5) \\ &= s^5 + 4.9280s^4 + 11.7388s^3 + 13.9904s^2 + 8.6140s + 2,\end{aligned}$$

which implies that

$$\boldsymbol{\alpha} = (4.9280, 11.7388, 13.9904, 8.6140, 2).$$

The augmented gains are therefore given as

$$\begin{aligned}K_1 &= (\boldsymbol{\alpha} - \mathbf{a}_{A_1})\mathcal{A}_{A_1}^{-1}\mathcal{C}_{A_1, B_1}^{-1} \\ &= (-0.4522, -18.8787, -0.7922, -2.8979, -0.1050)\end{aligned}$$

Step 3. The feedback gains are therefore given by

$$\begin{aligned}K &= K_1(1 : 4) = (-0.4522, -18.8787, -0.7922, -2.8979) \\ k_I &= K_1(5) = -0.1050\end{aligned}$$

Step 4. The reference gain is given by

$$k_r = \frac{-1}{C_r(A - BK)^{-1}B} = -0.4522.$$

Alternatively, we could have used the following Matlab script

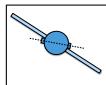
```

1 % original state space system
2 % state space design
3 A = [ ...
4     0, 0, 1, 0; ...
5     0, 0, 0, 1; ...
6     0, -P.m1*P.g/P.m2, -P.b/P.m2, 0; ...
7     0, (P.m1+P.m2)*P.g/P.m2/P.ell, P.b/P.m2/P.ell, 0; ...
8 ];
9 B = [0; 0; 1/P.m2; -1/P.m2/P.ell];
10 C = [ ...
11     1, 0, 0, 0; ...
12     0, 1, 0, 0; ...
13 ];
14
15 % form augmented system
16 Cr = [1,0,0,0];
17 A1 = [A, zeros(4,1); Cr, 0];
18 B1 = [B; 0];
19
20 % desired poles
21 wn_th = 2;
22 zeta_th = 0.707;
23 wn_z = 1;
24 zeta_z = 0.8;
25 des_char_poly = conv(conv([1,2*zeta_z*wn_z,wn_z^2], ...
26 [1,2*zeta_th*wn_th,wn_th^2]), ...
27 poly(-0.5));
28 des_poles = roots(des_char_poly);
29
30 % is the system controllable?
31 if rank ctrb(A1,B1) < 5,
32     disp('System Not Controllable');
33 else % if so, compute gains
34     K1 = place(A1,B1,des_poles);
35     P.K = K1(1:4);
36     P.ki = K1(5);
37     P.kr = -1/(Cr*inv(A-B*P.K)*B);

```

| |
|--------|
| 38 end |
|--------|

15.5 Design Study C. Satellite Attitude Control



Homework Problem C.17

- (a) Modify the state feedback solution developed in Homework 15 to add an integrator with anti-windup to the feedback loop for ϕ .
- (b) Change the parameters known to the controller by $\pm 5\%$.
- (c) Add a constant input disturbance of 1 Newtons-meters to the input of the plant and verify that when the integrator gain is set to zero, there is a steady state tracking error.
- (c) Tune the integrator to get zero steady state 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

$$C_r = (0 \ 1 \ 0 \ 0).$$

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.16, 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 $p_I = -0.1$. 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

$$\begin{aligned}\Delta_{cl}^d(s) &= (s + 1.0605 - j1.0608)(s + 1.0605 + j1.0608)\dots \\ &\quad (s + 0.7016 - j0.7018)(s + 0.7016 + j0.7018)(s + 0.1) \\ &= s^5 + 3.6243s^4 + 6.5636s^3 + 5.8673s^2 + 2.7406s + 0.2216,\end{aligned}$$

which implies that

$$\boldsymbol{\alpha} = (3.6243, \ 6.5636, \ 5.8673, \ 2.7406, \ 0.2216).$$

The augmented gains are therefore given as

$$\begin{aligned}K_1 &= (\boldsymbol{\alpha} - \mathbf{a}_{A_1})\mathcal{A}_{A_1, B_1}^{-1} \mathcal{C}_{A_1, B_1}^{-1} \\ &= (21.8364, \ 71.2255, \ 16.9252, \ 154.1450, \ 7.7558).\end{aligned}$$

Step 3. The feedback gains are therefore given by

$$\begin{aligned}K &= K_1(1 : 2) = (21.8364, \ 71.2255, \ 16.9252, \ 154.1450) \\ k_I &= K_1(3) = 7.7558.\end{aligned}$$

Step 4. The reference gain is given by

$$k_r = \frac{-1}{C_r(A - BK)^{-1}B} = 93.0619.$$

Alternatively, we could have used the following Matlab script

```

1 % original state space system
2 A = [...
3     0, 0, 1, 0; ...
4     0, 0, 0, 1; ...
5     -P.k/P.Js, P.k/P.Js, -P.b/P.Js, P.b/P.Js; ...
6     P.k/P.Jp, -P.k/P.Jp, P.b/P.Jp, -P.b/P.Jp; ...
7
8 ];
9 B = [0; 0; 1/P.Js; 0];
10 C = [...
11     1, 0, 0, 0; ...
12     0, 1, 0, 0; ...
13 ];

```

```

14
15 % form augmented system
16 Cr = [0,1,0,0];
17 A1 = [A, zeros(4,1); Cout, 0];
18 B1 = [B; 0];
19
20 % gains for pole locations
21 wn_th = 0.9924;
22 zeta_th = 0.707;
23 wn_phi = 1.5;
24 zeta_phi = 0.707;
25 ol_char_poly = charpoly(A);
26 des_char_poly = conv(conv([1,2*zeta_th*wn_th,wn_th^2],...
27 [1,2*zeta_phi*wn_phi,wn_phi^2]),...
28 poly(-.1));
29 des_poles = roots(des_char_poly);
30
31 % is the system controllable?
32 if rank ctrb(A1,B1) ≠ 5,
33 disp('System Not Controllable');
34 else % if so, compute gains
35 K1 = place(A1,B1,des_poles);
36 P.K = K1(1:4);
37 P.ki = K1(5);
38 P.kr = -1/(Cr*inv(A-B*P.K)*B);
39 end

```

Notes and References

Chapter 16

Observers

16.1 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

$$\dot{x} = Ax + Bu \quad (16.1)$$

$$y_r = C_r x \\ y_m = C_m x, \quad (16.2)$$

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. The objective of the observer is to estimate the state $x(t)$ given a history of the input $u(t)$ and the measured output $y_m(t)$. 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 16.1. Comparing Figure 16.1 to the case of full state feedback in Figure 14.4 note that the controller uses the estimated state \hat{x} instead of the full state x . In other words, the controller in Equation (14.36) becomes

$$u = -K\hat{x} + k_r r.$$

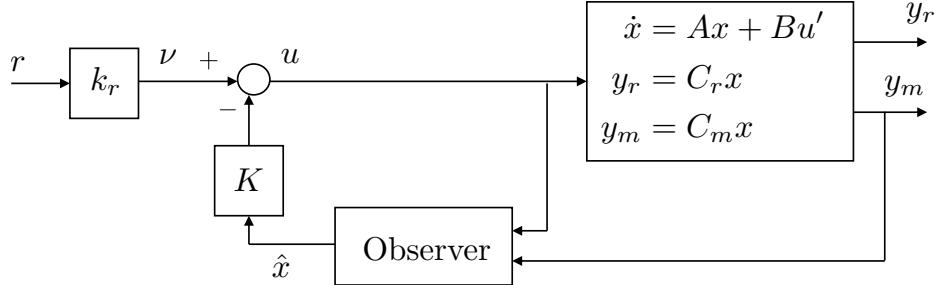


Figure 16.1: Closed loop system with observer based feedback.

In this chapter we will design a so-called Luenberger observer that has the structure given by

$$\dot{\hat{x}} = \underbrace{A\hat{x} + Bu}_{\text{predictor}} + \underbrace{L(y_m - C_m\hat{x})}_{\text{corrector}}. \quad (16.3)$$

The first term in Equation (16.3), i.e., $\dot{\hat{x}} = A\hat{x} + Bu$ is a copy of the equations of motion for the plant given in Equation (16.1). The second term given by $L(y_m - C_m\hat{x})$ is a correction term based, where L is the observer gain, and $(y_m - C_m\hat{x})$ is the difference between the measured output $y_m = C_mx$ and the predicted value of the measured output $C_m\hat{x}$.

Defining the observer error as $e = x - \hat{x}$ and differentiating with respect to time gives

$$\begin{aligned} \dot{e} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + Bu - (A\hat{x} + Bu + Ly_m - LC_m\hat{x}) \\ &= A(x - \hat{x}) + LC_mx - LC_m\hat{x} \\ &= (A - LC_m)(x - \hat{x}) \\ &= (A - LC_m)e. \end{aligned}$$

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 14.

16.1.1 Observer Design using Observer Canonic Form

Given the general SISO monic transfer function

$$Y(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} U(s), \quad (16.4)$$

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

$$\frac{d^n y}{dt^n} = -a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} - \cdots + \left(b_m \frac{d^m u}{dt^m} - a_m \frac{d^m y}{dt^m} \right) + \cdots + (b_1 \dot{u} - a_1 \dot{y}) + (b_0 u - a_0 y).$$

Integrating n times gives

$$y(t) = \int \left[-a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} - \cdots + \int \left[\left(b_m \frac{d^m u}{dt^m} - a_m \frac{d^m y}{dt^m} \right) + \cdots + \int \left[(b_1 \dot{u} - a_1 \dot{y}) + \int (b_0 u - a_0 y) \right] \dots \right] \dots \right]. \quad (16.5)$$

The analog computer implementation of Equation (16.5) is shown in Figure 16.2.

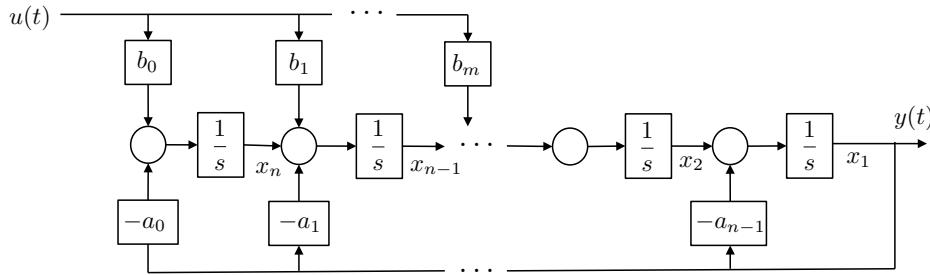


Figure 16.2: Block diagram showing the analog computer implementation of Equations (??) and (??).

Labeling the states as the output of the integrators, as shown in Figure 16.2 and writing the equations in linear state space form gives

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{pmatrix} = \begin{pmatrix} -a_{n-1} & 1 & \dots & 0 & 0 \\ -a_{n-2} & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ -a_1 & 0 & \dots & 0 & 1 \\ -a_0 & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_m \\ \vdots \\ b_1 \\ b_0 \end{pmatrix} u \quad (16.6)$$

$$= (1 \ 0 \ \dots \ 0 \ 0) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}. \quad (16.7)$$

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

$$A_o \triangleq \begin{pmatrix} -a_{n-1} & 1 & \dots & 0 & 0 \\ -a_{n-2} & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ -a_1 & 0 & \dots & 0 & 1 \\ -a_0 & 0 & \dots & 0 & 0 \end{pmatrix} \quad (16.8)$$

$$B_o \triangleq \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_m \\ \vdots \\ b_1 \\ b_0 \end{pmatrix} \quad (16.9)$$

$$C_o \triangleq (1 \ 0 \ \dots \ 0 \ 0). \quad (16.10)$$

Comparing the state space equations in observer canonic form in Equations (16.8)–(16.10) with the state space equations for control canonic form in Equations (14.14)–(14.16), 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 (16.6) with the transfer function in Equation (16.4) we see that the eigenvalues of A_o are the roots of the open loop characteristic polynomial

$$\Delta_{ol}(s) = \det(sI - A_o) = s^n + a_{n-1}s^{n-1} + \cdots + a_1s + a_0.$$

Since C_o in Equation (16.7) 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

$$\Delta_{cl,obs}(s) = \det(sI - (A_o - L_o C_o)) = s^n + (a_{n-1} + \ell_1)s^{n-1} + \cdots + (a_1 + \ell_{n-1}s + (a_0 + \ell_n)).$$

If the desired characteristic polynomial for the closed loop observation error is

$$\Delta_{cl,obs}^d(s) = s^n + \beta_{n-1}s^{n-1} + \cdots + \beta_1s + \beta_0, \quad (16.11)$$

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

$$L_o = \boldsymbol{\beta} - \mathbf{a}_A^\top, \quad (16.12)$$

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 is given by

$$L_o = \boldsymbol{\beta} - \mathbf{a}_{A_o}^\top = \begin{pmatrix} 50 \\ 1200 \\ 8000 \end{pmatrix} - \begin{pmatrix} 6 \\ -11 \\ 8 \end{pmatrix} = \begin{pmatrix} 44 \\ 1211 \\ 7992 \end{pmatrix}.$$

16.1.2 Observer Design: General Case

The formula given in Equation (16.12) assumes that the state space equations are in observer canonic form. Similar to the general full state feedback case discussed in Section 14.1.3, a formula for the observer gains can be derived for general state space equations. A general formula can be derived by following a similar method to that used in Section 16.12 by finding a state transformation matrix that converts general state space equations into observer canonic form. As an alternative, we will derive the formula 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 (14.19) is given by Equation (14.31) or

$$K = (\boldsymbol{\alpha} - \mathbf{a}_A)\mathcal{A}_A^{-1}\mathcal{C}_{A,B}^{-1}, \quad (16.13)$$

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 (16.11). Noting 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 (16.11). 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 Ackerman's formula (16.13) gives

$$L^\top = (\boldsymbol{\beta}^\top - \mathbf{a}_{A^\top})\mathcal{A}_{A^\top}^{-1}\mathcal{C}_{A^\top,C^\top}^{-1} \quad (16.14)$$

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 (16.14) gives

$$L = \mathcal{C}_{A^\top, C^\top}^{-\top} \mathcal{A}_A^{-\top} (\boldsymbol{\beta} - \mathbf{a}_A^\top), \quad (16.15)$$

where we have used the notation

$$M^{-\top} \triangleq (M^{-1})^\top = (M^\top)^{-1}.$$

Defining the *observability matrix*

$$\begin{aligned} \mathcal{O}_{A,C} &\stackrel{\Delta}{=} \mathcal{C}_{A^\top, C^\top}^\top \\ &= (C^\top \ A^\top C^\top \ (A^\top)^2 C^\top \ \dots \ (A^\top)^{n-1} C^\top)^\top \\ &= \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{pmatrix}, \end{aligned}$$

the observer gain (16.15) can be written as

$$L = \mathcal{O}_{A,C}^{-1} \mathcal{A}_A^{-\top} (\boldsymbol{\beta} - \mathbf{a}_A^\top), \quad (16.16)$$

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 (16.16) is Ackermann's formula for the observer gain L . Ackermann's formula is only valid for single input 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(obsv(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 (16.14).

The complete observer based controller is shown in Figure 16.3, where an integrator has also been included. 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 these two outputs are identical.

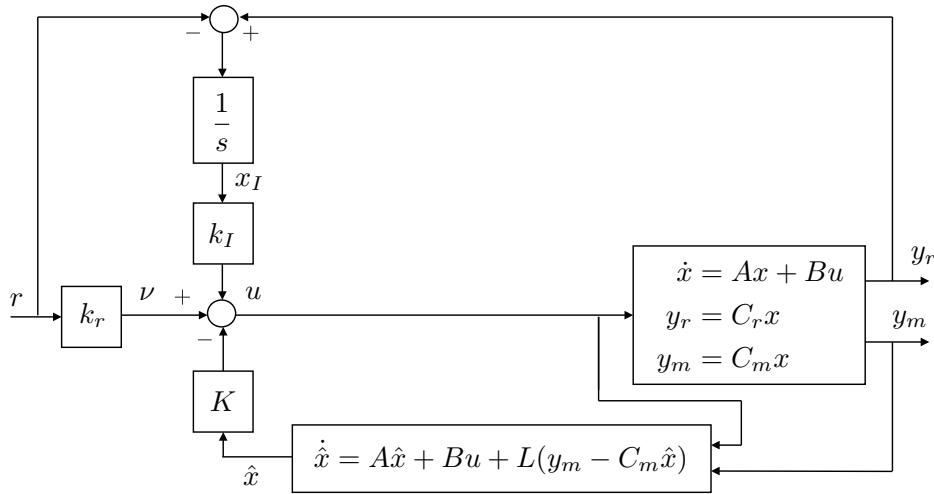


Figure 16.3: Closed loop system with observer based feedback, including integral feedback.

16.1.3 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 16.3 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 D, 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.

16.1.4 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 $y_m \in \mathbf{R}^p$ is the measured output, $y_{me} = C_m x_e \in \mathbf{R}^p$ 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

$$\dot{\hat{x}} = A\hat{x} + B\tilde{u} + L(\tilde{y}_m - C_m \hat{x}),$$

where $\hat{x} = \hat{x} - x_e$. Writing the observer in terms of \hat{x} we get

$$\dot{\hat{x}} - \dot{x}_e = A(\hat{x} - x_e) + B(u - u_e) + L((y_m - y_{me}) - C_m(\hat{x} - x_e)).$$

Noting that $\dot{x}_e = 0$ and that $y_{me} = C_m x_e$ gives

$$\dot{\hat{x}} = A(\hat{x} - x_e) + B(u - u_e) + L(y_m - C_m \hat{x}).$$

16.2 Summary of Design Process - Observer Design

The design process 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

$$\Delta_{ol}(s) = \det(sI - A) = s^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0,$$

and construct the row vector

$$\mathbf{a}_A = (a_{n-1}, a_{n-2}, \dots, a_1, a_0)$$

and the matrix

$$\mathcal{A}_A = \begin{pmatrix} 1 & a_{n-1} & a_{n-2} & \cdots & a_2 & a_1 \\ 0 & 1 & a_{n-1} & \cdots & a_3 & a_2 \\ \dots & & \ddots & & & \vdots \\ 0 & 0 & \cdots & 0 & 1 & a_{n-1} \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$$

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

```

16.2.1 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

$$\dot{\hat{x}} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \hat{x} + \begin{pmatrix} 0 \\ 4 \end{pmatrix} u + \begin{pmatrix} 23/5 \\ 271/5 \end{pmatrix} (y - (5 \ 0) \hat{x}), \quad (16.17)$$

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 (16.17) 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 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

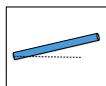
$$\dot{\hat{x}}(t) \approx \frac{\hat{x}(t) - \hat{x}(t-T)}{T}$$

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 .

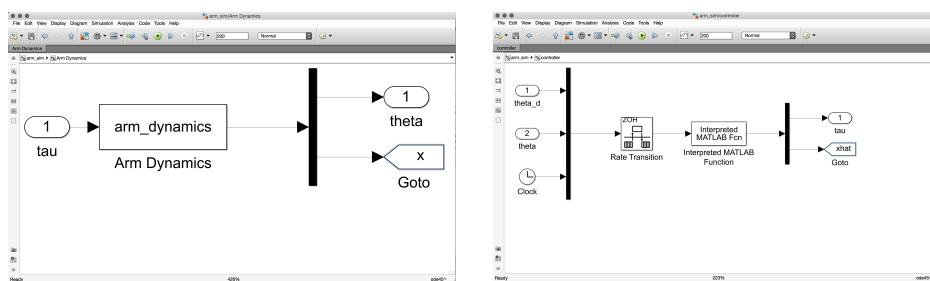
16.3 Design Study A. Single Link Robot Arm



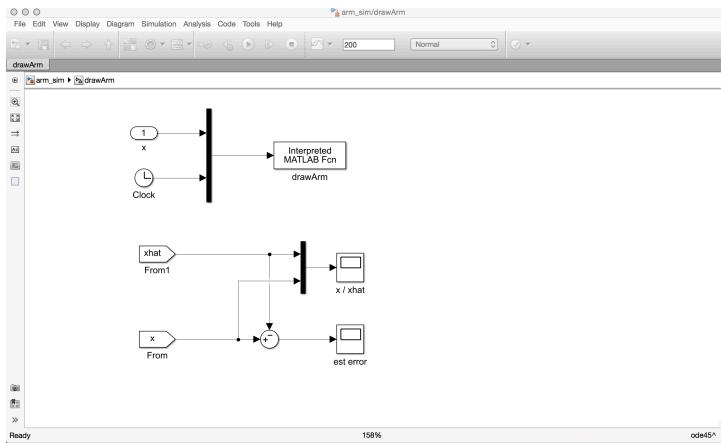
Homework Problem A.18

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 A.17.

- (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.17 so that the output of the s-function defining the dynamics is both y and x as shown in Figure 16.3. In addition, modify the Simulink diagram so that the controller outputs both u and \hat{x} , as also shown in Figure 16.3.



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 16.3.



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, within an input disturbance, and therefore remove the integrator. Modify your parameter file so that the parameters known to the controller are the actual plant parameters.

- (c) Verify that the state space system is observable by checking that $\text{rank}(\mathcal{O}) = n$.
- (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.

Solution

Matlab code used to design the observer based controller is shown below:

```

1 %——————
2 % state space design
3 P.A = [...
4     0, 1;...
5     0, -3*P.b/P.m/(P.ell^2);...
6 ];
7 P.B = [0; 3/P.m/(P.ell^2) ];
8 P.C = [...
9     1, 0;...
10 ];
11
12 % desired pole locations
13 wn = 4.4117;
14 zeta = 0.707;
15 des_char_poly = [1,2*zeta*wn,wn^2];
16 des_poles = roots(des_char_poly);
17
18 % is the system controllable?
19 if rank(ctrb(A,B))≠2,
20     disp('System Not Controllable');
21 else % if so, compute gains
22     P.K = place(A,B,des_poles);
23     P.kr = -1/(P.C*inv(P.A-P.B*P.K)*P.B);
24 end
25
26 % observer design
27 % pick observer poles
28 a = 10;
29 q1 = -a + j*a;
30 q2 = -a - j*a;
31
32 % is the system observable?
```

```

33 if rank(obsv(P.A,P.C))<2,
34     disp('System Not Observable');
35 else % if so, compute gains
36     P.L = place(P.A',P.C',[q1,q2])';
37 end

```

Lines 1–24 are identical to the state feedback design from Homework A.16. The poles of the observation error are selected in Lines 28–30 in a form that allows a to be tuned for performance. The observability check is in Lines 33–34, and the observer gains are computed using the `place` command in Line 36. Note the transposes on this line.

Matlab code for the observer based control (without integrator) is shown below:

```

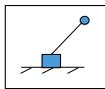
1 function out=arm_ctrl(in,P)
2     theta_c = in(1);
3     theta    = in(2);
4     t        = in(3);
5
6     % compute equilibrium torque tau_e
7     theta_e = 0;
8     tau_e = P.m*P.g*(P.ell/2)*cos(theta);
9     x_e = [theta_e; 0];
10
11    % implement observer
12    persistent xhat           % estimated state (for observer)
13    persistent tau            % delayed input (for observer)
14    if t<P.Ts,
15        xhat = [0; 0];
16        tau = 0;
17    end
18    N = 10;
19    for i=1:N,
20        xhat = xhat + P.Ts/N*...
21            (P.A*(xhat-x_e)+P.B*(tau-tau_e)+ P.L*(theta-P.C*xhat));
22    end
23
24    % compute the state feedback controller
25    tau_tilde = - P.K*(xhat-x_e) + P.kr*(theta_c-theta_e);
26    % compute total torque
27    tau = sat( tau_e + tau_tilde, P.tau_max);
28
29    out = [tau; xhat];

```

| |
|-----------|
| 30 end |
|-----------|

The equilibrium values are defined in Lines 7–9. The observer is implemented in Lines 11–22. The observer-based control is implemented in Lines 24–27, including saturation on the control. See the wiki for the complete solution.

16.4 Design Study B. Inverted Pendulum



Homework Problem B.18

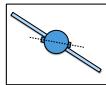
The objective of this problem is to design an observer that estimates the state of the system and to use the estimated state in the controller designed in Homework B.17.

- (a) Modify the simulink diagram from HW B.17 so that the output of the s-function defining the dynamics is both y and x as shown in Figure 16.3. In addition, modify the Simulink diagram so that the controller outputs both u and \hat{x} , as also shown in Figure 16.3.
- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, within an input disturbance, and therefore remove the integrator. Modify your parameter file so that the parameters known to the controller are the actual plant parameters.
- (c) Verify that the state space system is observable by checking that $\text{rank}(\mathcal{O}) = n$.
- (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.

Solution

See the wiki page for a complete solution to this problem.

16.5 Design Study C. Satellite Attitude Control



Homework Problem C.18

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.17.

- (a) Modify the simulink diagram from HW C.17 so that the output of the s-function defining the dynamics is both y and x as shown in Figure 16.3. In addition, modify the Simulink diagram so that the controller outputs both u and \hat{x} , as also shown in Figure 16.3.
- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, within an input disturbance, and therefore remove the integrator. Modify your parameter file so that the parameters known to the controller are the actual plant parameters.
- (c) Verify that the state space system is observable by checking that $\text{rank}(\mathcal{O}) = n$.
- (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.

Solution

See the wiki page for a complete solution to this problem.

Notes and References

RWB: - reference to Luenberger observer.

Chapter 17

Disturbance Observers

17.1 Theory

In previous sections we have discussed that modelling errors and other real-world effects result in an input disturbance to the system as shown in Figure 17.1. The objective of this section is explain the effect of the disturbance

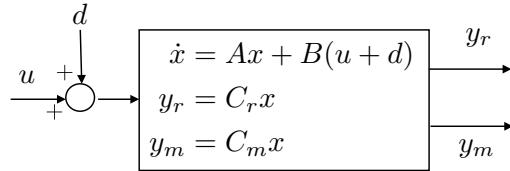


Figure 17.1: Input disturbance to plant modeled by state space equations.

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

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y_m - C_m \hat{x}). \quad (17.1)$$

Defining the observation error as $e = x - \hat{x}$ and taking the derivative with respect to time gives

$$\dot{e} = \dot{x} - \dot{\hat{x}} \quad (17.2)$$

$$\begin{aligned} &= Ax + Bu + Bd - A\hat{x} - Bu - LC_m x + LC_m \hat{x} \\ &= (A - LC_m)e + Bd. \end{aligned} \quad (17.3)$$

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 (17.1) as

$$\dot{\hat{x}} = A\hat{x} + B(u + d) + L(y_m - C_m \hat{x}), \quad (17.4)$$

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 (17.3) 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 the observer structure.

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

$$A_2 = \begin{pmatrix} A & B \\ \mathbf{0} & 0 \end{pmatrix}$$

$$C_2 = \begin{pmatrix} C_m & 0 \end{pmatrix}.$$

If (A_2, C_2) are observable, then using the techniques discussed in Chapter 16 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 L = place(A2', C2', desired_observer_poles)'
```

17.1.1 Simple Example

Consider the example in Section 14.2.1, 15.2.1, and 16.2.1. A block diagram for the system is shown in Figure 17.2, where a constant input disturbance has been added to the system.

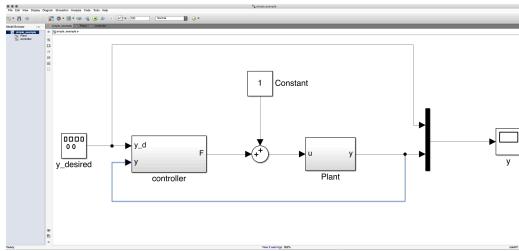


Figure 17.2: Block diagram for simple example, where a constant input disturbance has been added to the system.

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_obs = 10;
14 zeta_obs = 0.707;
15 charpoly = [1,2*zeta_obs*wn_obs,wn_obs^2];
16 des_obs_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
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_observ_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*y_d - P.K*xhat;

```

The step response for this controller is shown in Figure 17.3, 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.

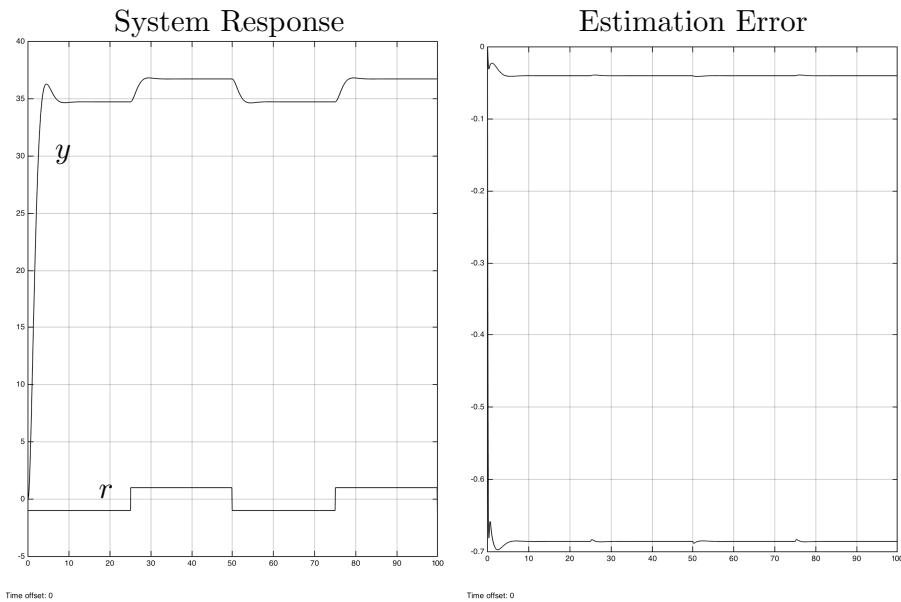


Figure 17.3: Step response for standard observer based control without integrator and without disturbance observer.

If an integrator is added to the system using the following design file

```

1 integrator_pole = -.5; % 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 P.kr = -1/(P.C*inv(P.A-P.B*P.K)*P.B);
13 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
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 = y - y_d;
19 if abs(xhat(2))<.1,
20     integrator = integrator + (P.Ts/2)*(error+error_d1);
21 end
22 error_d1 = error;
23 % observer based feedback controller
24 u = P.kr*y_d - P.K*xhat - P.ki*integrator;

```

Note the anti-windup scheme in Line 19. If the anti-wind-up scheme is present, then the system response is identical to Figure 17.3 because the bias in the state estimate prevents the condition from being true. Removing the anti-windup scheme results in the system response is shown in Figure 17.4

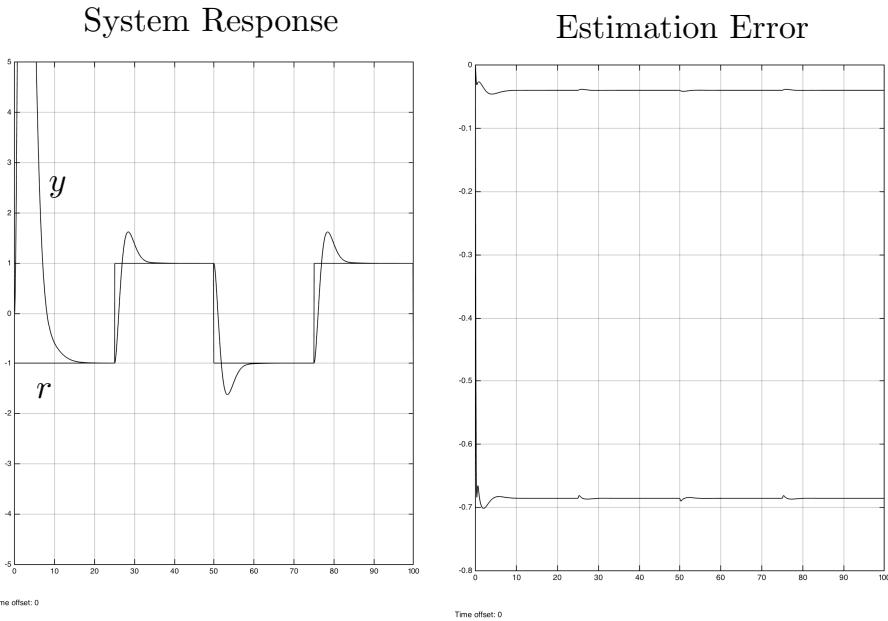


Figure 17.4: Step response for observer based control with integrator but without disturbance observer.

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];
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.

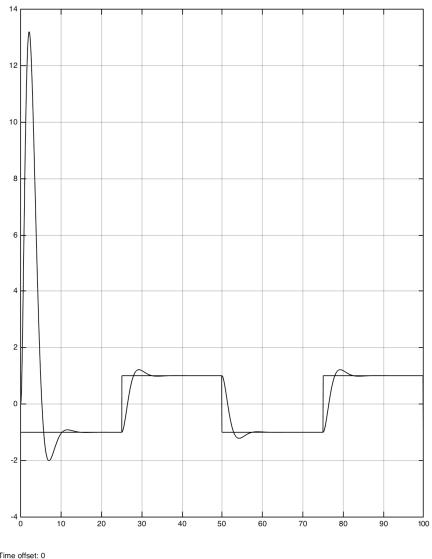
```

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*y_d - 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 17.5

System Response



Estimation Error

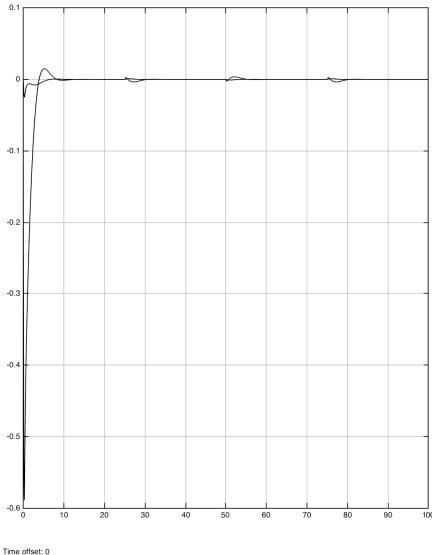
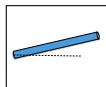


Figure 17.5: Step response for observer based control without integrator but disturbance observer.

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 could be added back into the system, and the anti-windup scheme can be included because the bias in the states has been removed.

17.2 Design Study A. Single Link Robot Arm



Homework Problem A.19

- (a) Modify the parameter file from HW A.18 to use inexact parameters, as in HW A.15 and HW A.17. Explicitly add an additional input disturbance of 0.05 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 $\dot{\theta}$ 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

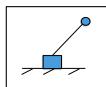
large steady state error.

- (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.

Solution

See the associated wiki page for a complete solution.

17.3 Design Study B. Inverted Pendulum



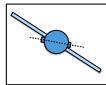
Homework Problem B.19

- (a) Modify the parameter file from HW B.18 to use inexact parameters, as in HW B.15 and HW B.17. Explicitly add an additional input disturbance of 0.5 Newton to the cart. 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{z} 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.

Solution

See the associated wiki page for a complete solution.

17.4 Design Study C. Satellite Attitude Control



Homework Problem C.19

- (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.

Solution

See the associated wiki page for a complete solution.

Notes and References

Part V

Loopshaping Control Design

RWB: Write an overview of what this part is all about, and how the chapters fit together.

Chapter 18

Bode Plots

18.1 Theory

18.1.1 Manipulating Complex Numbers

Any complex number z can be represented in rectangular 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

$$z = |z| e^{j\angle z}, \quad (18.1)$$

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 18.1.

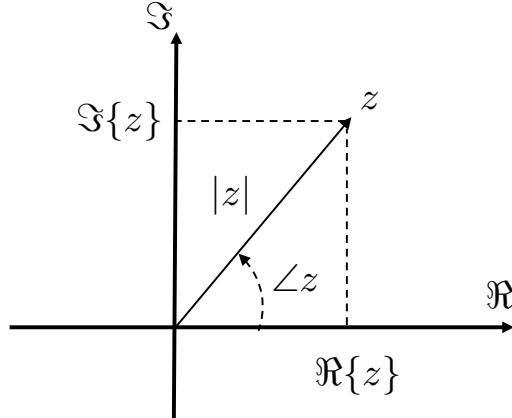


Figure 18.1: Relationship between the real and imaginary parts of z , and the magnitude and phase of z .

From the geometry shown in Figure 18.1 we can see that

$$\begin{aligned}|z| &= \sqrt{\Re\{z\}^2 + \Im\{z\}^2} \\ \angle z &= \tan^{-1} \frac{\Im\{z\}}{\Re\{z\}}.\end{aligned}$$

Similarly, from Euler's relationship

$$e^{j\theta} = \cos \theta + j \sin \theta,$$

we have from Equation (18.1) that

$$z = |z| e^{j\angle z} = |z| \cos \angle z + j |z| \sin \angle z$$

which implies that

$$\begin{aligned}\Re\{z\} &= |z| \cos \angle z \\ \Im\{z\} &= |z| \sin \angle z.\end{aligned}$$

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

$$\frac{z_1}{z_2} = \frac{|z_1| e^{j\angle z_1}}{|z_2| e^{j\angle z_2}} = \frac{|z_1|}{|z_2|} \frac{e^{j\angle z_1}}{e^{j\angle z_2}} = \frac{|z_1|}{|z_2|} e^{j(\angle z_1 - \angle z_2)},$$

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

$$\begin{aligned} H(s) &= \frac{K(s + z_1)(s + z_2) \dots (s + z_m)}{(s + p_1)(s + p_2) \dots (s + p_n)} \\ &= \frac{|K| e^{j\angle K} |s + z_1| e^{j\angle(s+z_1)} |s + z_2| e^{j\angle(s+z_2)} \dots |s + z_m| e^{j\angle(s+z_m)}}{|s + p_1| e^{j\angle(s+p_1)} |s + p_2| e^{j\angle(s+p_2)} \dots |s + p_n| e^{j\angle(s+p_n)}} \\ &= \frac{|K| |s + z_1| |s + z_2| \dots |s + z_m| e^{j\angle K} e^{j\angle(s+z_1)} e^{j\angle(s+z_2)} \dots e^{j\angle(s+z_m)}}{|s + p_1| |s + p_2| \dots |s + p_n|} \frac{e^{j\angle(s+p_1)} e^{j\angle(s+p_2)} \dots e^{j\angle(s+p_n)}}{} \\ &= \frac{|K| |s + z_1| |s + z_2| \dots |s + z_m|}{|s + p_1| |s + p_2| \dots |s + p_n|} e^{j(\angle K + \sum_{i=1}^m \angle(s+z_i) - \sum_{i=1}^n \angle(s+p_i))}. \end{aligned}$$

Therefore

$$|H(s)| = \frac{|K| |s + z_1| |s + z_2| \dots |s + z_m|}{|s + p_1| |s + p_2| \dots |s + p_n|} \quad (18.2)$$

$$\angle H(s) = \angle K + \sum_{i=1}^m \angle(s + z_i) - \sum_{i=1}^n \angle(s + p_i). \quad (18.3)$$

18.1.2 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 the a physical system with input $u(t)$ and output $y(t)$ as shown in Figure 18.2.

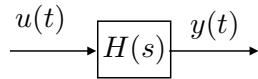


Figure 18.2: LTI system represented by the transfer function $H(s)$ with input $u(t)$ and output $y(t)$.

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

$$y(t) = A |H(j\omega_0)| \sin(\omega_0 t + \angle H(j\omega_0)). \quad (18.4)$$

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 18.3 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 (18.4). 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

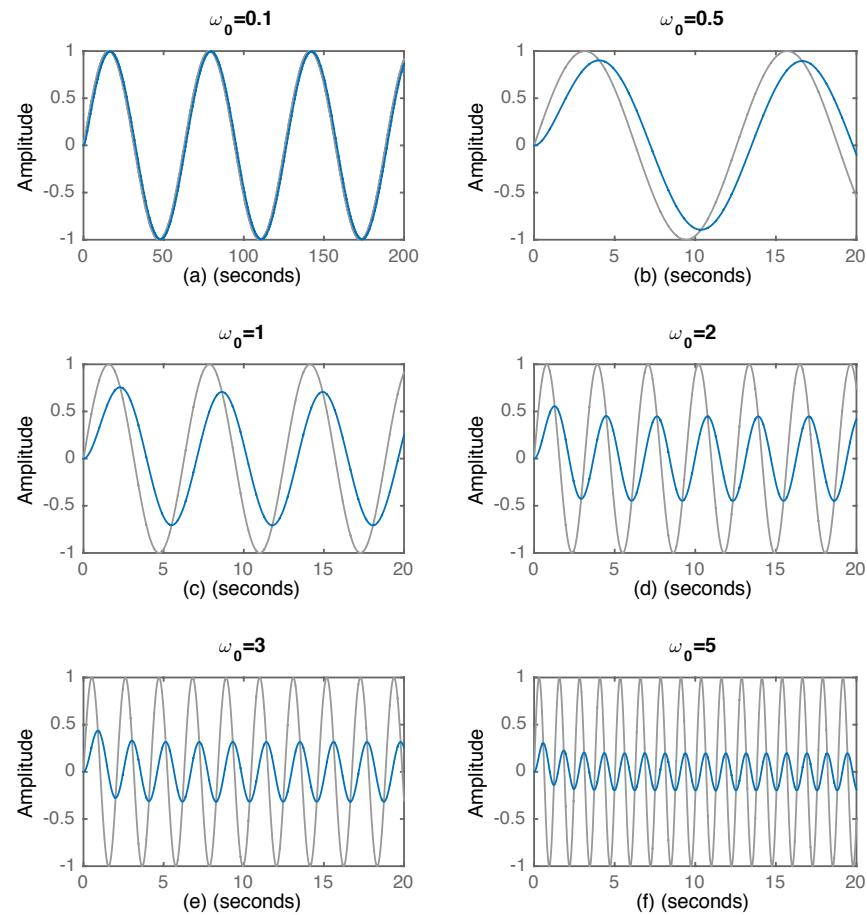


Figure 18.3: Frequency response to LTI system with transfer function $H(s) = 1/(s + 1)$.

$\omega_0 = 5$, $H(j\omega_0) \approx 0.2e^{j80^\circ}$, 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 ???. 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 18.4 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.

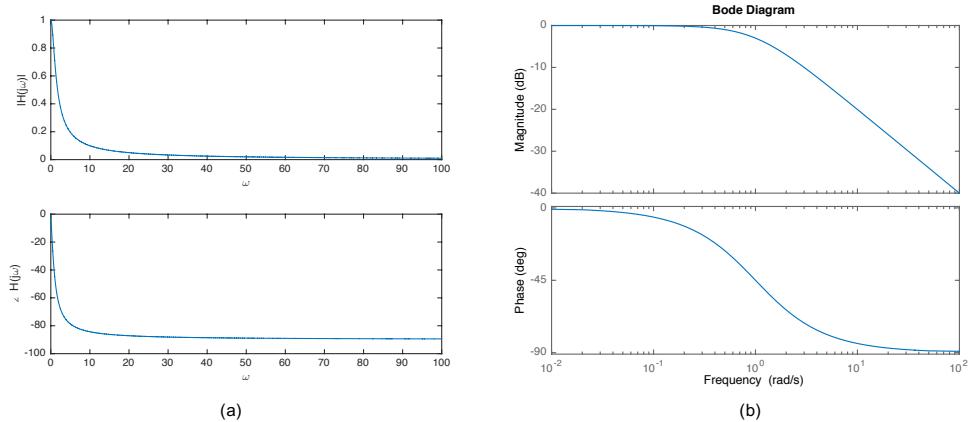


Figure 18.4: Frequency response of $H(s) = 1/(s + 1)$ on a linear scale (a), and a log plot (b).

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,

$$20 \log |H(j\omega)| = 20 * n \text{ dB} \implies |H(j\omega)| = 10^n.$$

18.1.3 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 follows:

$$\begin{aligned} H(s) &= \frac{K(s + z_1)(s + z_2) \dots (s + z_m)}{(s + p_1)(s + p_2) \dots (s + p_n)} \\ &= \frac{Kz_1z_2 \dots z_m}{p_1p_2 \dots p_n} \frac{(1 + s/z_1)(1 + s/z_2) \dots (1 + s/z_m)}{(1 + s/p_1)(1 + s/p_2) \dots (1 + s/p_n)}. \end{aligned}$$

Therefore, following Equations (18.2) and (18.3) we have

$$|H(j\omega)| = \left| \frac{Kz_1z_2 \dots z_m}{p_1p_2 \dots p_m} \right| \frac{|1 + j\omega/z_1| |1 + j\omega/z_2| \dots |1 + j\omega/z_m|}{|1 + j\omega/p_1| |1 + j\omega/p_2| \dots |1 + j\omega/p_n|} \quad (18.5)$$

$$\angle H(j\omega) = \angle \left(\frac{Kz_1z_2 \dots z_m}{p_1p_2 \dots p_m} \right) + \sum_{i=1}^m \angle(1 + j\omega/z_i) - \sum_{i=1}^n \angle(1 + j\omega/p_i). \quad (18.6)$$

Since $20 \log |AB| = 20 \log |A| + 20 \log |B|$, Equation (18.5) gives

$$\begin{aligned} 20 \log |H(j\omega)| &= 20 \log \left| \frac{Kz_1z_2 \dots z_m}{p_1p_2 \dots p_m} \right| \\ &\quad + \sum_{i=1}^m 20 \log |1 + j\omega/z_i| - \sum_{i=1}^n 20 \log |1 + j\omega/p_i|. \end{aligned} \quad (18.7)$$

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

$$20 \log \left| \frac{Kz_1z_2 \dots z_m}{p_1p_2 \dots p_m} \right|$$

is not a function of ω and is therefore a constant line on the Bode plot. Also note that

$$\angle \left(\frac{Kz_1z_2 \dots z_m}{p_1p_2 \dots p_m} \right)$$

is also a constant and is either 0 degrees if $\left(\frac{Kz_1z_2 \dots z_m}{p_1p_2 \dots p_m} \right) > 0$, or 180 degrees if $\left(\frac{Kz_1z_2 \dots z_m}{p_1p_2 \dots p_m} \right) < 0$.

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

$$H(j\omega) = z(1 + \alpha\omega/z).$$

Therefore, from Equation (18.7) we have

$$20 \log |H(j\omega)| = 20 \log |z| + 20 \log |1 + j\omega/z| = 20 \log |z| + 20 \log \sqrt{1 + (\omega/z)^2}.$$

When $\omega/z \ll 1$, i.e., when $\sqrt{1 + (\omega/z)^2} \approx 1$, we therefore have

$$20 \log |H(j\omega)| \approx 20 \log |z|.$$

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 $\omega/z \gg 1$, i.e. when $\sqrt{1 + (\omega/z)^2} \approx |\omega/z|$, we therefore have

$$20 \log |H(j\omega)| \approx 20 \log |z| + 20 \log |\omega/z|.$$

Note that the second term is a straight line on a Bode plot that intersects the 0 dB axis when $\omega = z$ and grows 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 18.5, along with the Bode plot produced by Matlab.

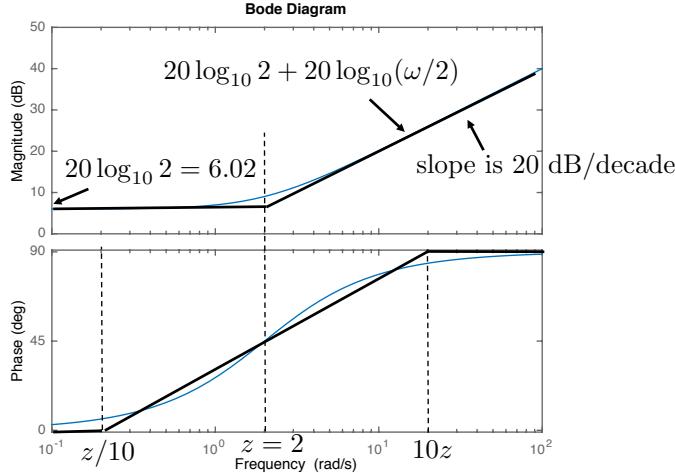


Figure 18.5: Straight line approximation for a single zero.

Note that the largest mismatch between the straight line approximation and the actual Bode is at $\omega = z$. In that case we have

$$20 \log |H(jz)| \approx 20 \log |z| + 20 \log \sqrt{2} = 20 \log |z| + 3 \text{ dB}.$$

Therefore, at the corner frequency z the Bode plot is 3 dB above the low frequency response.

To approximate the phase plot, note that

$$\angle H(j\omega) = \angle z(1 + j\omega/z) = \tan^{-1} \frac{\omega}{z}. \quad (18.8)$$

When $\omega = z$ we have that $\tan^{-1} 1 = 45$ degrees. When $\omega \ll z$, $\tan^{-1}(\text{small number}) \approx 0$ degrees, and when $\omega \gg z$, $\tan^{-1}(\text{large number}) \approx 90$ degrees. The phase plot as calculated by Matlab is shown in Figure 18.5. 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 Figure 18.5.

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 (18.6) 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 18.6 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.

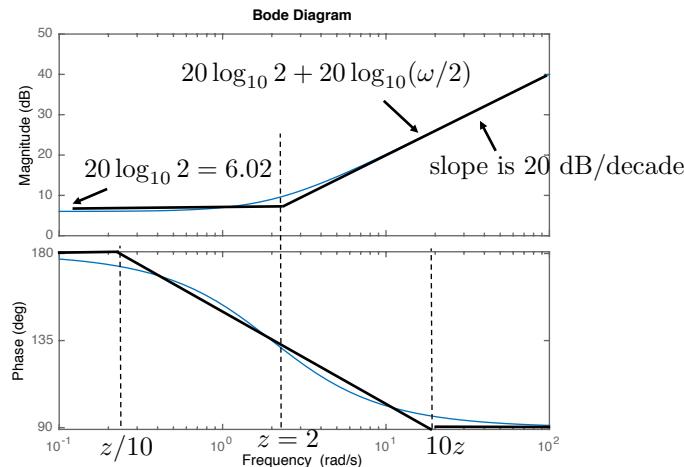


Figure 18.6: Straight line approximation for a zero in the right half of the complex plane.

Now consider drawing the Bode plot for a single pole, i.e., $H(s) = \frac{1}{s+p}$, which has a single pole at $s = -p$ which is in the right half plane when $p > 0$. In Bode canonical form $H(s)$ becomes

$$H(j\omega) = \frac{1}{1 + j\omega/p}.$$

Therefore, from Equation (18.7) we have

$$20 \log |H(j\omega)| = 20 \log |1| - 20 \log |1 + j\omega/p| = -20 \log \sqrt{1 + (\omega/p)^2}.$$

When $\omega/p \ll 1$, i.e., when $\sqrt{1 + (\omega/p)^2} \approx 1$, we therefore have

$$20 \log |H(j\omega)| \approx 0.$$

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 $\omega/p \gg 1$, i.e. when $\sqrt{1 + (\omega/p)^2} \approx |\omega/p|$, we therefore have

$$20 \log |H(j\omega)| \approx -20 \log |\omega/p|.$$

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 18.7, along with the Bode plot produced by Matlab.

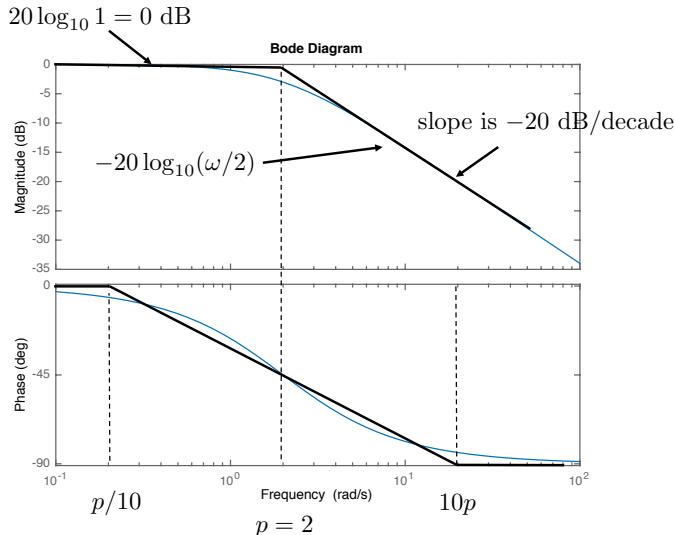


Figure 18.7: Straight line approximation for a single pole.

Note that the largest mismatch between the straight line approximation and the actual Bode is at $\omega = p$. In that case we have

$$20 \log |H(jp)| \approx -20 \log \sqrt{2} = -3 \text{ dB}.$$

Therefore, at the corner frequency p the Bode plot is 3 dB below the low frequency response.

To approximate the phase plot, note that

$$\angle H(j\omega) = -\angle(1 + j\omega/p) = -\tan^{-1} \frac{\omega}{p}. \quad (18.9)$$

When $\omega = p$ we have that $-\tan^{-1} 1 = -45$ degrees. When $\omega \ll z$, $-\tan^{-1}(\text{small number}) \approx 0$ degrees, and when $\omega \gg z$, $-\tan^{-1}(\text{large number}) \approx -90$ degrees. The phase plot as calculated by Matlab is shown in Figure 18.7. 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 18.7.

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 (18.6) 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 rather. Figure 18.8 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.

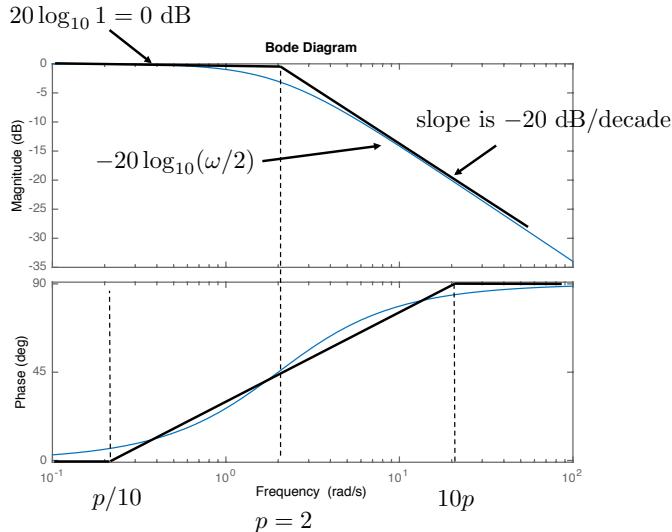


Figure 18.8: Straight line approximation for a pole in the right half of the complex plane.

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 $\zeta = 0.707 = \sqrt{2}/2$, then $20 \log_{10} |H(j\omega)| = -20 \log_{10} 1/\sqrt{2} = -3 \text{ dB}$. For the angle plot, when $\omega \ll \omega_n$ we have $\angle H(j\omega) \approx \tan^{-1} \frac{0}{1} = 0 \text{ degrees}$. When $\omega \gg \omega_n$ we have $\angle H(j\omega) \approx \tan^{-1} (2\zeta \frac{\omega_n}{\omega}) \approx -180 \text{ degrees}$, and when $\omega = \omega_n$ we have $\angle H(j\omega) \approx -\tan^{-1} \frac{2\zeta}{0} = -90 \text{ degrees}$.

The Bode plot for different value of ζ is shown in Figure 18.9

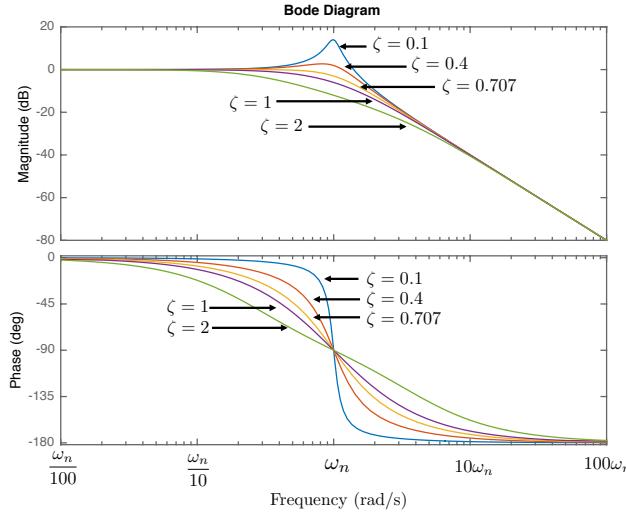


Figure 18.9: The Bode plot for a second order system for various values of ζ .

18.1.4 Example

Find the straight line approximation for the Bode plot for the transfer function

$$H(s) = \frac{20}{s(s+10)}. \quad (18.10)$$

In Bode canonical form we have

$$H(j\omega) = \frac{20}{10} \frac{1}{j\omega(1 + j\frac{\omega}{10})}.$$

Therefore

$$20 \log_{10} |H(j\omega)| = 20 \log_{10} 2 - 20 \log_{10} |\omega| - 20 \log_{10} \sqrt{1 + \left(\frac{\omega}{10}\right)^2}. \quad (18.11)$$

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 increase in ω . 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 18.10.

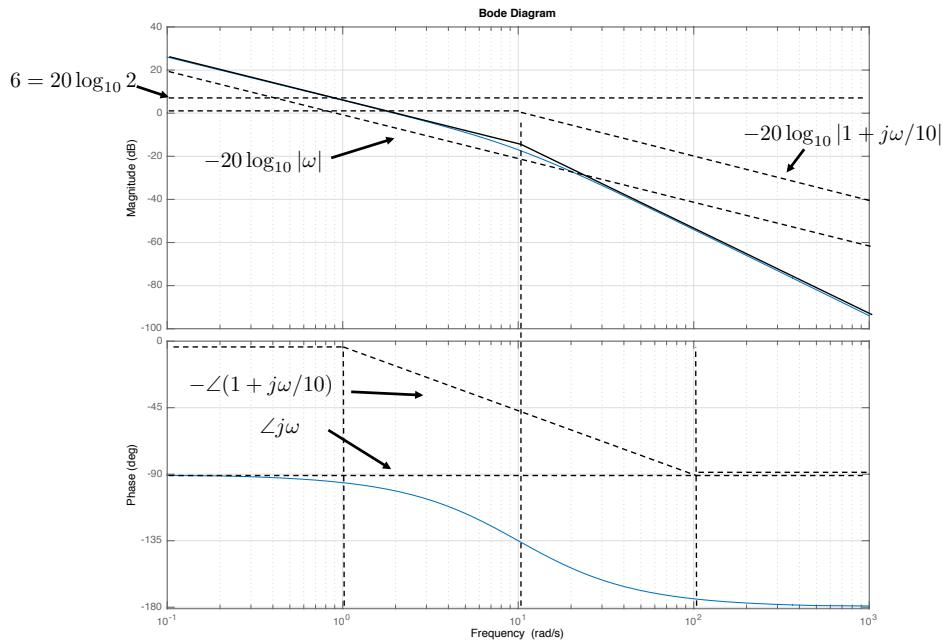


Figure 18.10: Bode plot for the transfer function given in Equation (18.10).

18.1.5 Example

Find the straight line approximation for the Bode plot for the transfer function

$$H(s) = \frac{8000(s+1)(s+10)}{(s+2)(s+20)(s+200)}. \quad (18.12)$$

In Bode canonical form we have

$$\begin{aligned} H(j\omega) &= \frac{8000 \cdot 10}{2 \cdot 20 \cdot 200} \frac{(1+j\omega)(1+j\frac{\omega}{10})}{(1+j\frac{\omega}{2})(1+j\frac{\omega}{20})(1+j\frac{\omega}{200})} \\ &= 10 \frac{(1+j\omega)(1+j\frac{\omega}{10})}{(1+j\frac{\omega}{2})(1+j\frac{\omega}{20})(1+j\frac{\omega}{200})}. \end{aligned}$$

Therefore

$$\begin{aligned} 20 \log_{10} |H(j\omega)| &= 20 \log_{10} 1020 \log_{10} |1+j\omega| + 20 \log_{10} \left| 1+j\frac{\omega}{10} \right| \\ &\quad - 20 \log_{10} \left| 1+j\frac{\omega}{2} \right| - 20 \log_{10} \left| 1+j\frac{\omega}{20} \right| - 20 \log_{10} \left| 1+j\frac{\omega}{200} \right|. \end{aligned} \quad (18.13)$$

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

$$\begin{aligned} \angle H(j\omega) &= \angle 10 + \angle(1+j\omega) + \angle(1+j\frac{\omega}{10}) \\ &\quad - \angle(1+j\frac{\omega}{2}) - \angle(1+j\frac{\omega}{20}) - \angle(1+j\frac{\omega}{200}). \end{aligned}$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 18.11.

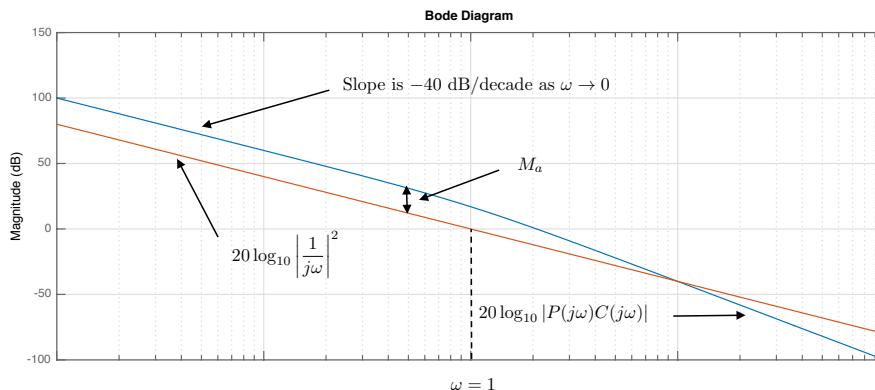
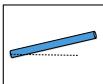


Figure 18.11: Bode plot for the transfer function given in Equation (18.12).

18.2 Design Study A. Single Link Robot Arm



Homework Problem A.20

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

The transfer function for the single link robot arm is

$$P(s) = \frac{3/m\ell^2}{s(s + 3b/m\ell^2)} = \frac{54.72}{s(s + 0.6019)}. \quad (18.14)$$

In Bode canonical form we have

$$P(j\omega) = \frac{90.9121}{(j\omega)(1 + j\frac{\omega}{0.6019})}$$

Therefore

$$20 \log_{10} |P(j\omega)| = 20 \log_{10} 90.9121 - 20 \log_{10} |j\omega| - 20 \log_{10} \left| 1 + j\frac{\omega}{0.6019} \right|. \quad (18.15)$$

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

$$\angle P(j\omega) = \angle 90.9121 - \angle(j\omega) - \angle(1 + j\frac{\omega}{0.6019}).$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 18.12.

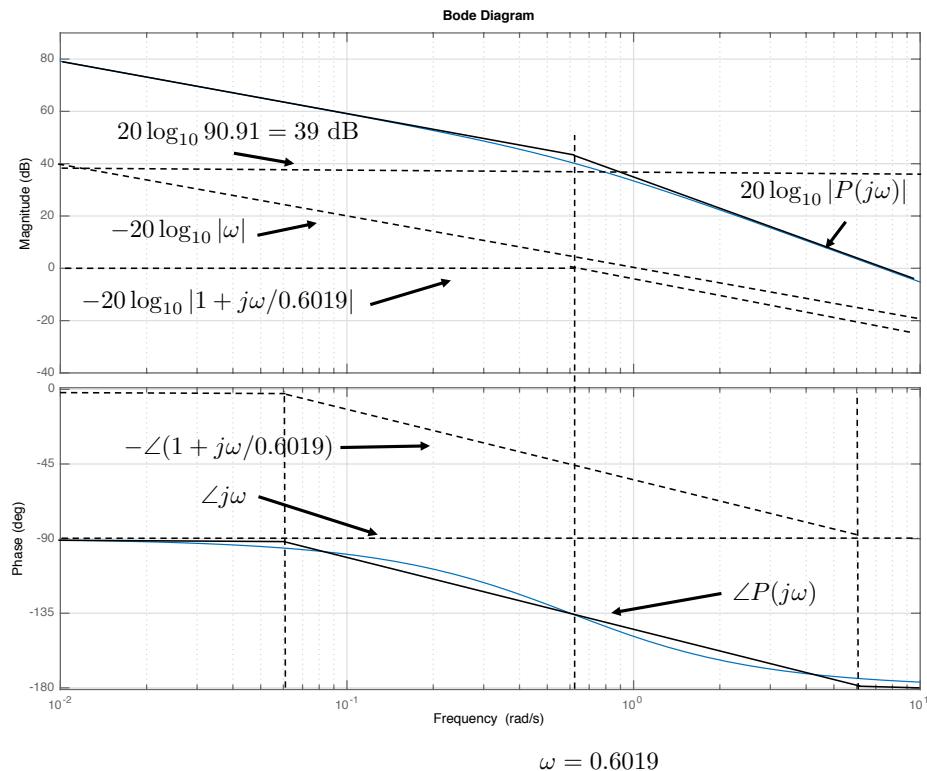


Figure 18.12: Bode plot for the transfer function given in Equation (18.14).

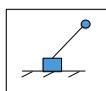
The Matlab command to generate the Bode plot is

```

1 >> P = tf([90.9121], [1, 0.6019, 0]);
2 >> figure(1), clf, bode(P), grid on

```

18.3 Design Study B. Inverted Pendulum



Homework Problem B.20

- (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

The transfer function for the inner loop of the inverted pendulum is

$$P_{in}(s) = \frac{-1/m_2\ell}{s^2 - \frac{(m_1+m_2)g}{m_2\ell}} = \frac{-2.023}{s^2 - 25.12} = \frac{-2.023}{(s + 5.012)(s - 5.012)}. \quad (18.16)$$

In Bode canonical form we have

$$P_{in}(j\omega) = \frac{0.0805}{(1 + j\frac{\omega}{5.012})(1 - j\frac{\omega}{5.012})}$$

Therefore

$$\begin{aligned} 20 \log_{10} |P_{in}(j\omega)| &= 20 \log_{10} 0.0805 \\ &\quad - 20 \log_{10} \left| 1 + j\frac{\omega}{5.012} \right| - 20 \log_{10} \left| 1 - j\frac{\omega}{5.012} \right|. \end{aligned} \quad (18.17)$$

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

$$\angle P_{in}(j\omega) = \angle 0.0805 - \angle \left(1 + j\frac{\omega}{5.012} \right) - \angle \left(1 - j\frac{\omega}{5.012} \right).$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 18.13.

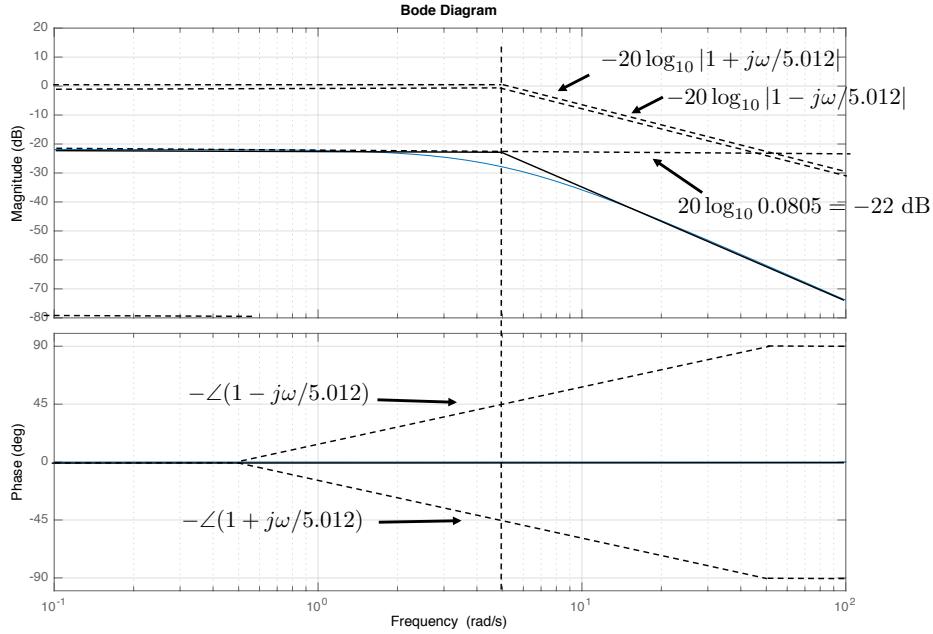


Figure 18.13: Bode plot for the transfer function given in Equation (18.17).

The Matlab command to generate the Bode plot is

```

1 >> Pin = tf([-2.023], [1, 0, -25.129]);
2 >> figure(1), clf, bode(Pin), grid on

```

The transfer function for the outer loop of the inverted pendulum is

$$P_{out}(s) = \frac{-m_1 g / m_2}{s(s + \frac{b}{m_2})} = \frac{-2.26}{s(s + 0.051)}. \quad (18.18)$$

In Bode canonical form we have

$$P_{out}(j\omega) = \frac{-44.3}{(j\omega)(1 + j\frac{\omega}{0.051})}$$

Therefore

$$\begin{aligned} 20 \log_{10} |P_{out}(j\omega)| &= 20 \log_{10} 44.3 \\ &\quad - 20 \log_{10} |j\omega| - 20 \log_{10} \left| 1 + j\frac{\omega}{0.051} \right|. \end{aligned} \quad (18.19)$$

Similarly, the phase is given by

$$\angle P_{out}(j\omega) = \angle -44.3 - \angle(j\omega) - \angle(1 + j\frac{\omega}{0.051}).$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 18.14.

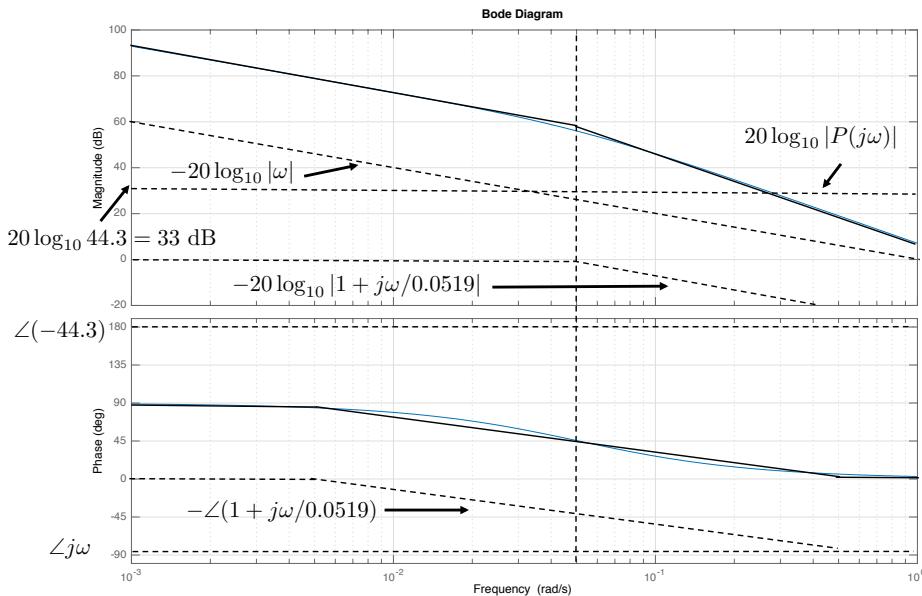


Figure 18.14: Bode plot for the transfer function given in Equation (18.19).

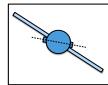
The Matlab command to generate the Bode plot is

```

1 >> Pout = tf([-2.26], [1, 0.051, 0]);
2 >> figure(1), clf, bode(Pout), grid on

```

18.4 Design Study C. Satellite Attitude Control



Homework Problem C.20

- (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

The transfer function for the inner loop of the satellite attitude problem is

$$P_{in}(s) = \frac{1/J_s}{s^2 + \frac{b}{J_s}s + \frac{k}{J_s}} = \frac{0.2}{s^2 + 0.01s + 0.03}. \quad (18.20)$$

In Bode canonical form we have

$$P_{in}(j\omega) = \frac{6.67}{1 + 0.33j\omega + \left(j\frac{\omega}{0.173}\right)}$$

Therefore

$$20 \log_{10} |P_{in}(j\omega)| = 20 \log_{10} 6.67 - 20 \log_{10} \left| 1 + 0.33j\omega + \left(j\frac{\omega}{0.173}\right) \right| \quad (18.21)$$

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

$$\angle P_{in}(j\omega) = \angle 6.67 - \angle \left(1 + 0.33j\omega + \left(j\frac{\omega}{0.173}\right) \right).$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 18.15.

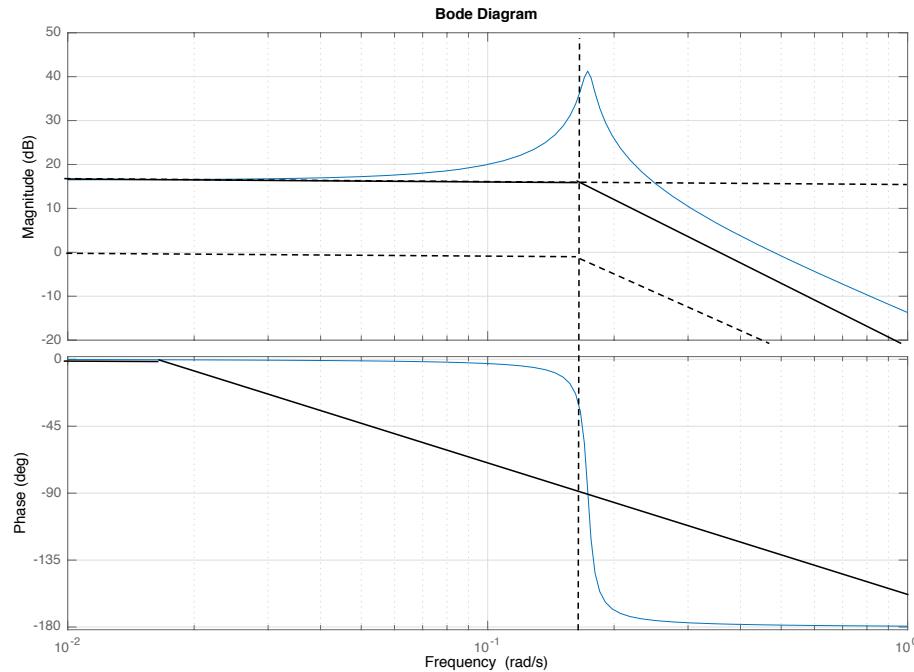


Figure 18.15: Bode plot for the transfer function given in Equation (18.21).

The Matlab command to generate the Bode plot is

```

1 >> Pin = tf([0.2], [1, 0.01, 0.03]);
2 >> figure(1), clf, bode(Pin), grid on

```

The transfer function for the outer loop of the satellite is

$$P_{out}(s) = \frac{b/J_p}{s + \frac{b}{J_p}} = \frac{0.05}{s + 0.05}. \quad (18.22)$$

In Bode canonical form we have

$$P_{out}(j\omega) = \frac{1}{1 + j\frac{\omega}{0.05}}$$

Therefore

$$20 \log_{10} |P_{out}(j\omega)| = 20 \log_{10} 1 - 20 \log_{10} \left| 1 + j\frac{\omega}{0.05} \right|. \quad (18.23)$$

Similarly, the phase is given by

$$\angle P_{out}(j\omega) = \angle 1 - \angle(1 + j\frac{\omega}{0.05}).$$

The straight line approximation as well as the Bode plot generated by Matlab are shown in Figure 18.16.

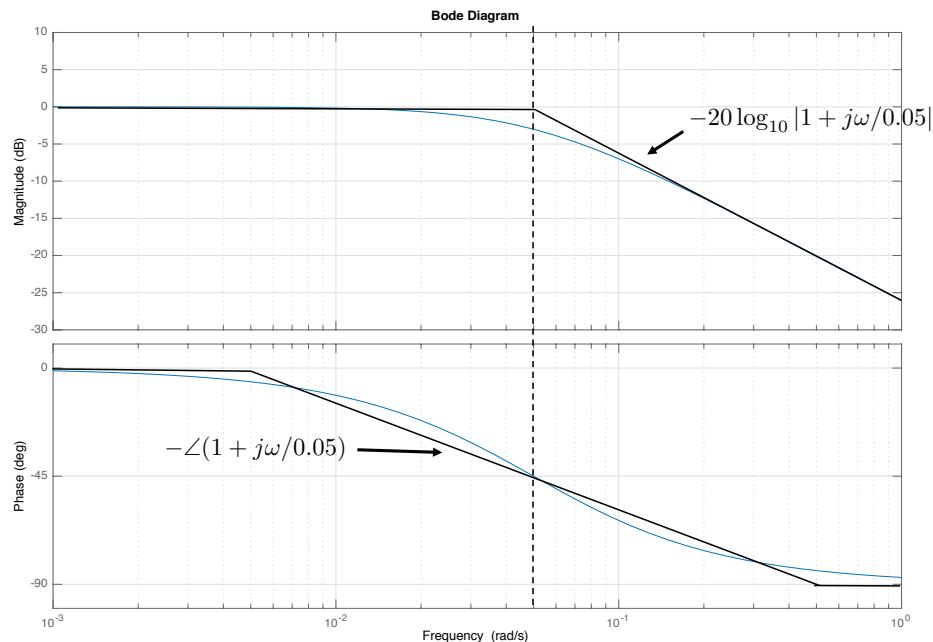


Figure 18.16: Bode plot for the transfer function given in Equation (18.23).

The Matlab command to generate the Bode plot is

```

1 >> Pout = tf([0.05], [1, 0.05]);
2 >> figure(1), clf, bode(Pout), grid on

```

Notes and References

Chapter 19

Tracking, Disturbance Rejection, Noise Attenuation

19.1 Theory

In this chapter we show how the magnitude of the frequency response of the open loop system can be used to assess the performance of the closed-loop feedback system. A general feedback loop is shown in Figure 19.1. In this feedback system, $r(t)$ is a known 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 $C(s)$. 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 function to y . Starting at $Y(s)$ and following the loop backward to Y we get

$$\begin{aligned} Y(s) &= -D_{out}(s) + P(-D_{in}(s) + C(R(s) - N(s) + Y(s))) \\ \implies (1+PC)Y(s) &= PCR(s) - PCN(s) - D_{out}(s) - PD_{in}(s) \\ Y(s) &= \frac{PC}{1+PC}R(s) - \frac{PC}{1+PC}N(s) - \frac{1}{1+PC}D_{out}(s) - \frac{P}{1+PC}D_{in}(s). \end{aligned} \tag{19.1}$$

Defining the error to be $E(s) \triangleq R(s) - Y(s)$ we get

$$\begin{aligned} E(s) &= R(s) - Y(s) \\ &= \frac{1+PC}{1+PC}R(s) - \frac{PC}{1+PC}R(s) + \frac{PC}{1+PC}N(s) + \frac{1}{1+PC}D_{out}(s) + \frac{P}{1+PC}D_{in}(s) \\ &= \frac{1}{1+PC}R(s) + \frac{PC}{1+PC}N(s) + \frac{1}{1+PC}D_{out}(s) + \frac{P}{1+PC}D_{in}(s), \end{aligned} \quad (19.2)$$

$$(19.3)$$

Therefore,

1. The transfer function from the reference r to the error e is $\frac{1}{1+P(s)C(s)}$,
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 $\frac{P(s)}{1+P(s)C(s)}$.

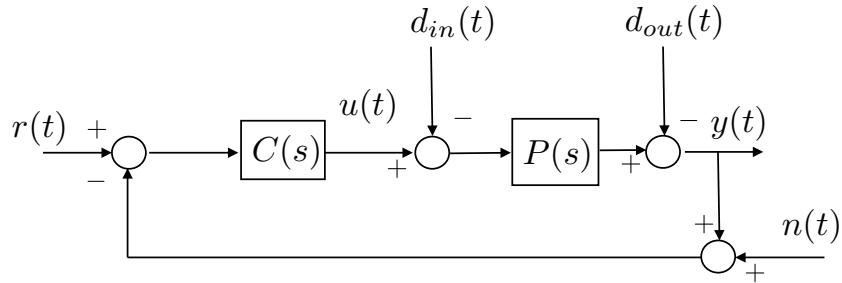


Figure 19.1: General feedback loop.

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 loop gain $20 \log_{10} |P(j\omega)C(j\omega)|$.

19.1.1 Tracking

The relationship between the reference signal $r(t)$ and 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

$$20 \log_{10} |P(j\omega)C(j\omega)| \geq 20 \log_{10} 1/\gamma_r, \quad (19.4)$$

which needs to be satisfied for all $\omega \leq \omega_r$. On a Bode plot, the constraint represented by Equation (19.4) is shown graphically in Figure 19.2.

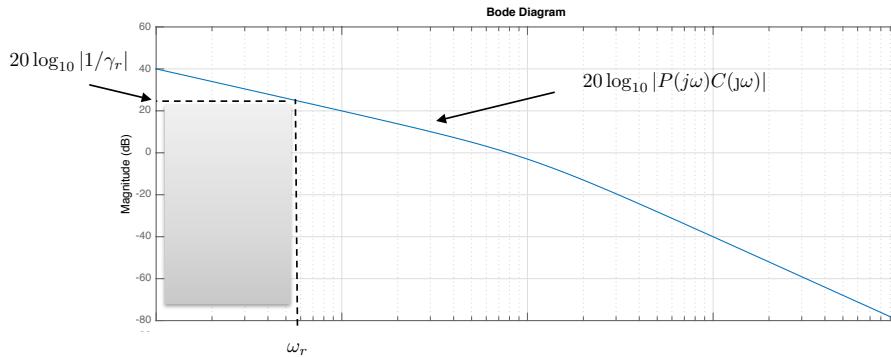


Figure 19.2: Bode plot representation of the loopshaping constraint corresponding to tracking $r(t)$ so that the corresponding error satisfies $|e(t)| \leq \gamma_r |r(t)|$.

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

$$|e(t)| \leq 10^{-B_r/20} |r(t)|, \quad (19.5)$$

for all $r(t)$ with frequency content below ω_r , which implies that

$$\gamma_r = 10^{-B_r/20}.$$

19.1.2 Noise Attenuation

The relationship between the noise signal $n(t)$ and the error $e(t)$ is

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

Therefore

$$|E(j\omega)| \leq \left| \frac{P(j\omega)C(j\omega)}{1 + P(j\omega)C(j\omega)} \right| |N(j\omega)|.$$

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

$$20 \log_{10} |P(j\omega)C(j\omega)| \leq 20 \log_{10} \gamma_n, \quad (19.6)$$

which needs to be satisfied for all $\omega \geq \omega_{no}$. On a Bode plot, the constraint represented by Equation (19.6) is shown graphically in Figure 19.3.

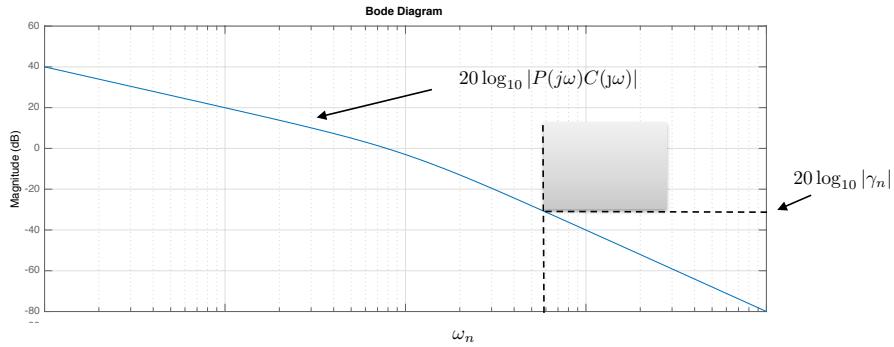


Figure 19.3: Bode plot representation of the loopshaping constraint corresponding to attenuating the noise $n(t)$ so that the corresponding error satisfies $|e(t)| \leq \gamma_n |n(t)|$.

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

$$\gamma_n = 10^{-B_n/20}.$$

19.1.3 Reject Output Disturbances

The relationship between the output disturbance signals $d_{out}(t)$ and 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

$$20 \log_{10} |P(j\omega)C(j\omega)| \geq 20 \log_{10} 1/\gamma_{d_{out}}, \quad (19.7)$$

which needs to be satisfied for all $\omega \leq \omega_{d_{out}}$. On a Bode plot, the constraint represented by Equation (19.7) is shown graphically in Figure 19.4.

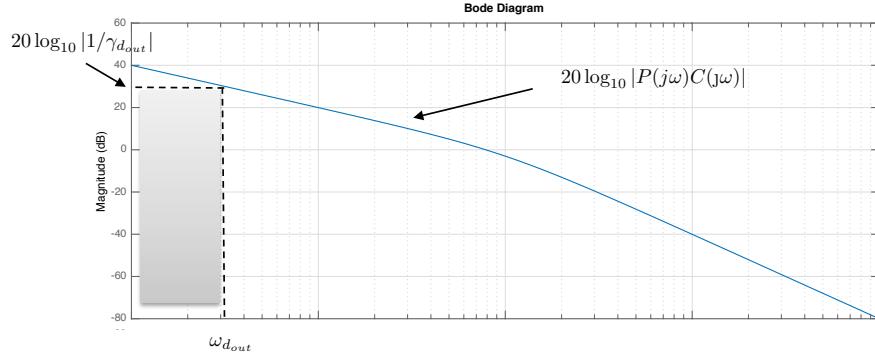


Figure 19.4: Bode plot representation of the loopshaping constraint corresponding to rejecting the output disturbance $d_{out}(t)$ so that the corresponding error satisfies $|e(t)| \leq \gamma_{d_{out}} |d_{out}(t)|$.

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

$$\gamma_{d_{out}} = 10^{-B_{d_{out}}/20}.$$

19.1.4 Reject Input Disturbances

The relationship between the input disturbance signals $d_{in}(t)$ and the error $e(t)$ is

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

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

$$20 \log_{10} |P(j\omega)C(j\omega)| - 20 \log_{10} |P(j\omega)| \geq 20 \log_{10} 1/\gamma_{d_{in}}, \quad (19.8)$$

which needs to be satisfied for all $\omega \leq \omega_{d_{in}}$. On a Bode plot, the constraint represented by Equation (19.8) is shown graphically in Figure 19.5, where it can be seen that the loop gain $|PC|$ must be above magnitude of the plant $|P|$ by the specified amount.

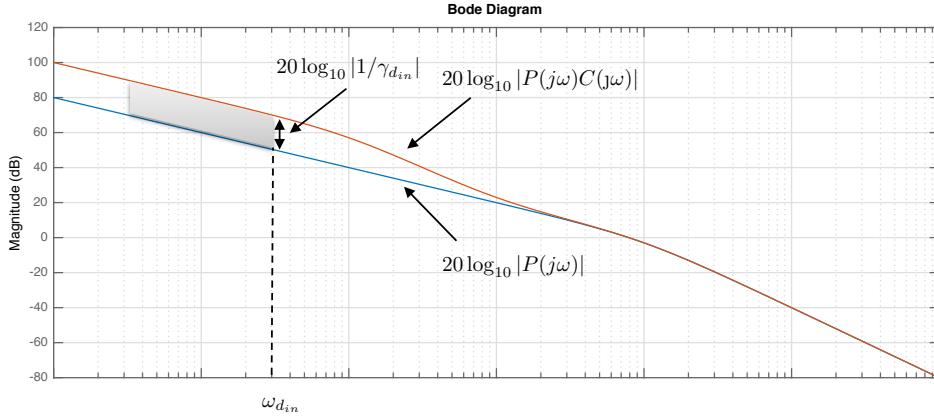


Figure 19.5: Bode plot representation of the loopshaping constraint corresponding to rejecting the input disturbance $d_{in}(t)$ so that the corresponding error satisfies $|e(t)| \leq \gamma_{d_{in}} |d_{in}(t)|$.

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

$$|e(t)| \leq 10^{-B_{d_{in}}/20} |d_{in}(t)|, \quad (19.9)$$

for all $d_{in}(t)$ with frequency content below $\omega_{d_{in}}$, which implies that

$$\gamma_{d_{in}} = 10^{-B_{d_{in}}/20}.$$

19.1.5 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 19.6.

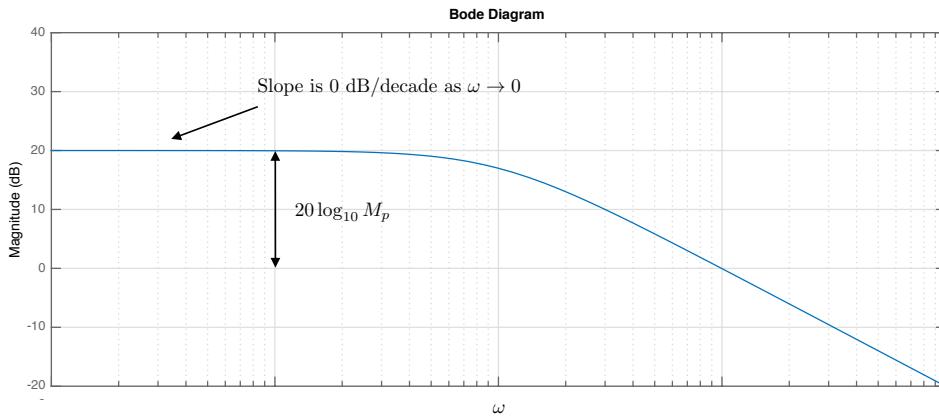


Figure 19.6: Bode plot of a type 0 system.

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 19.6, $20 \log_{10} M_p$ 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

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{1}{1 + M_p} A. \quad (19.10)$$

Type 1 System

For a type 1 system since $P(s)C(s)$ has one free integrator

$$\begin{aligned} & \lim_{s \rightarrow 0} sP(s)C(s) = \text{constant} \\ \implies & \lim_{\omega \rightarrow 0} |(j\omega)P(j\omega)C(j\omega)| = \text{constant} \\ \implies & \lim_{\omega \rightarrow 0} \frac{|P(j\omega)C(j\omega)|}{\left| \frac{1}{j\omega} \right|} = \text{constant} \\ \implies & \lim_{\omega \rightarrow 0} \left[20 \log_{10} |P(j\omega)C(j\omega)| - 20 \log_{10} \left| \frac{1}{j\omega} \right| \right] = \text{constant}. \end{aligned}$$

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 19.7.

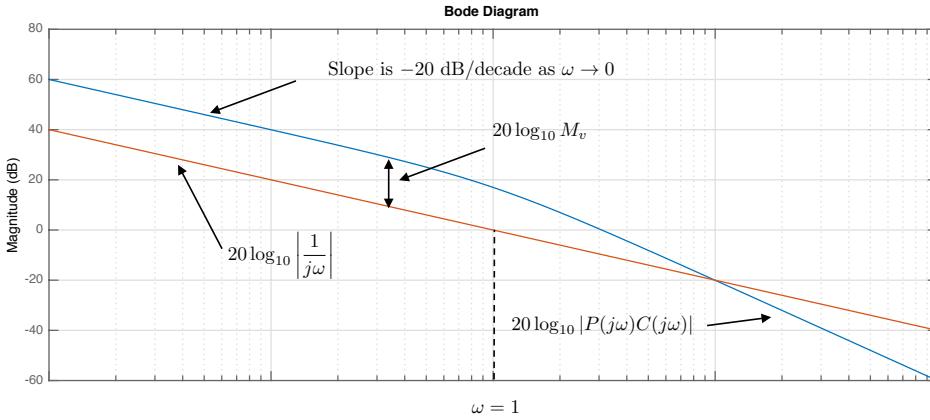


Figure 19.7: Bode plot of a type 1 system.

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 19.7, $20 \log_{10} M_v$ 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

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{1}{M_v} A. \quad (19.11)$$

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 as $\omega \rightarrow 0$, as shown in Figure 19.8.$

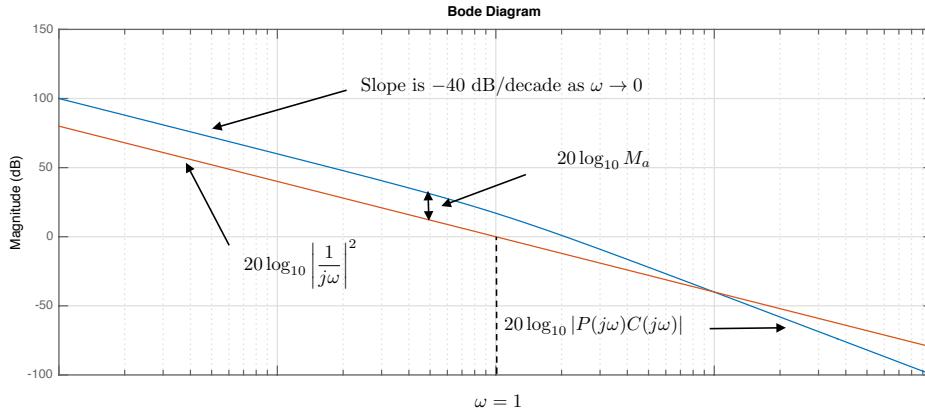


Figure 19.8: Bode plot of a type 2 system.

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 19.8, $20 \log_{10} M_a$ 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

$$\lim_{\omega \rightarrow 0} \left[20 \log_{10} |P(j\omega)C(j\omega)| - 20 \log_{10} \left| \frac{1}{j\omega} \right|^2 \right] = B_2 \text{ dB.}$$

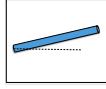
Then

$$M_a = 10^{B_2/20},$$

and

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{1}{M_a} A. \quad (19.12)$$

19.2 Design Study A. Single Link Robot Arm



Homework Problem A.21

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 PD control, and (3) the plant under PID control, using the control gains calculated in Homework A.15.

- (a) To what percent error can the closed loop system under PID control track the desired input if all of the frequency content of $\theta^d(t)$ is below $\omega_r = 0.4$ radians per second?
- (b) If the desired input is $\theta^d(t) = 5t$ for $t \geq 0$, what will be the steady state tracking error to this input when using PD and 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 PD control $P(s)C_{PD}(s)$, and the loop gain with PID control $P(s)C_{PID}(s)$ are shown in Figure 19.9.

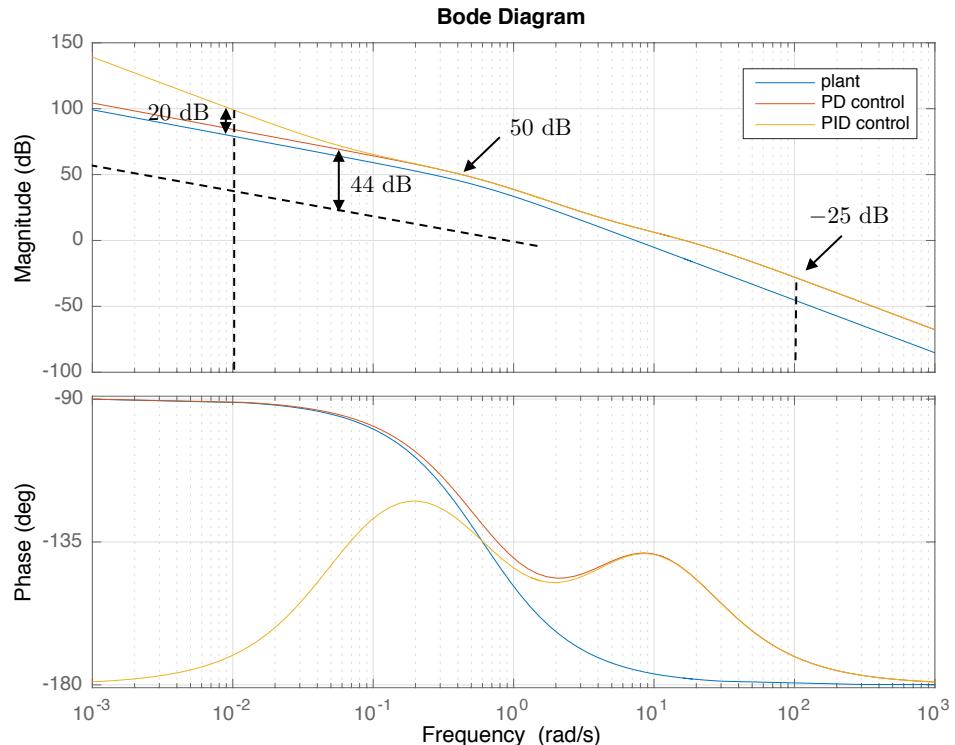


Figure 19.9: Bode plot for single link arm, plant only, under PD control, and under PID control.

From Figure 19.9 we see that below $\omega_r = 0.4$ rad/sec, the loop gain is above $B_r = 50$ dB. Therefore, from Equation (19.5) we have that

$$|e(t)| \leq \gamma_r |r(t)|,$$

where $\gamma_r = 10^{-50/20} = 0.0032$, which implies that the tracking error will be 0.32% of the magnitude of the input.

(b) Suppose now that the desired reference input is $\theta^d(t) = 5t$. 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 ramp with zero steady state error. Under PD control, there will be a finite error. Since the loop gain under PD control is $B_1 = 44$ dB above the $1/s$ line, from Equation (19.11) the steady state error satisfies

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{1}{M_v} A = 10^{-44/20} \cdot 5 = 0.0315.$$

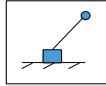
(c) If the input disturbance is below $\omega_{d_{in}} = 0.01$ rad/s, then the difference between the loop gain and the plant is $B_{d_{in}} = 20$ dB at $\omega_{d_{in}} = 0.01$ rad/s, therefore, from Equation (19.9) we see that

$$\gamma_{d_{in}} = 10^{-20/20} = 0.1,$$

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 19.9 that $B_n = 25$ dB. Therefore, $\gamma_n = 10^{-25/20} = 0.0562$ which implies that 5.6% of the noise will show up in the output signal.²⁰¹

19.3 Design Study B. Inverted Pendulum



Homework Problem B.21

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.15.

- (a) To what percent error can the closed loop system track the desired input if all of the frequency content of $\theta^d(t)$ 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 (1) the plant, (2) the plant under PD control, and (3) the plant under PID, using the control gains calculated in Homework B.15.

- (c) If the reference signal $y^d(t)$ is a ramp of magnitude 2 meters/sec, what is the tracking error under PD and 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 19.10.

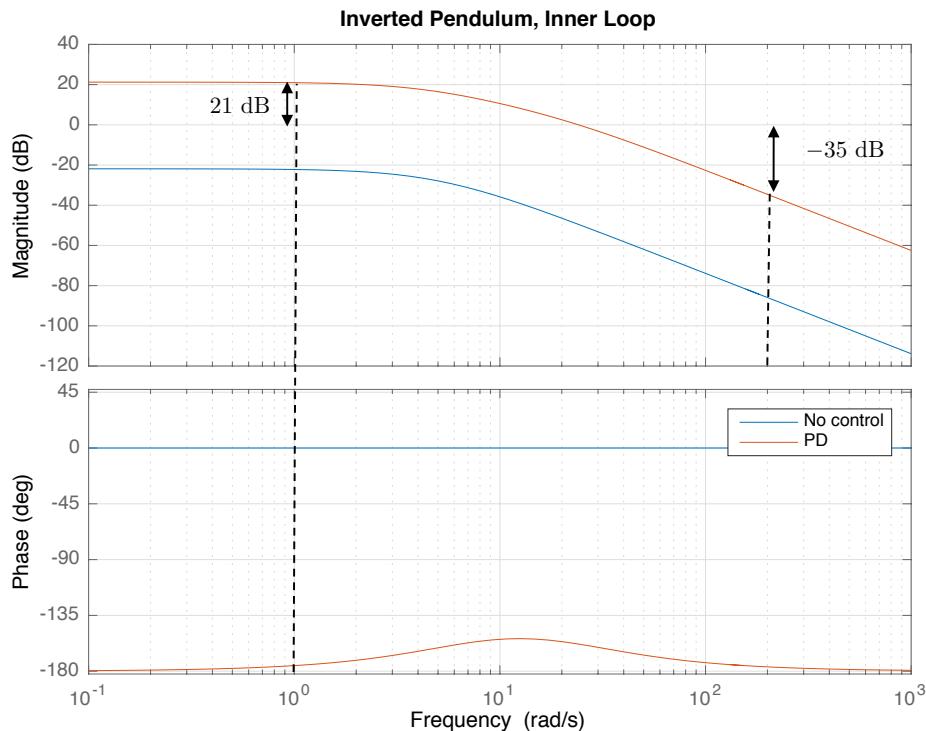


Figure 19.10: Bode plot for inner loop of the inverted pendulum, plant only, and under PD control.

From Figure 19.10 we see that below $\omega_r = 1.0$ rad/sec, the loop gain is above $B_r = 21$ dB. Therefore, from Equation (19.5) we have that

$$|e(t)| \leq \gamma_r |r(t)|,$$

where $\gamma_r = 10^{-21/20} = 0.0891$, which implies that the tracking error will be 8.91% of the magnitude of the input.

(b) For noise greater than $\omega_{no} = 200$ rad/sec, we see from Figure 19.10 that $B_n = 35$ dB. Therefore, $\gamma_n = 10^{-35/20} = 0.0178$ which implies that 1.78% of the noise will show up in the output signal.

(c) The Bode plot of the plant $P(s)$, and the loop gain with PD control $P(s)C_{PD}(s)$, and with PID control $P(s)C_{PID}(s)$ are shown in Figure 19.11.

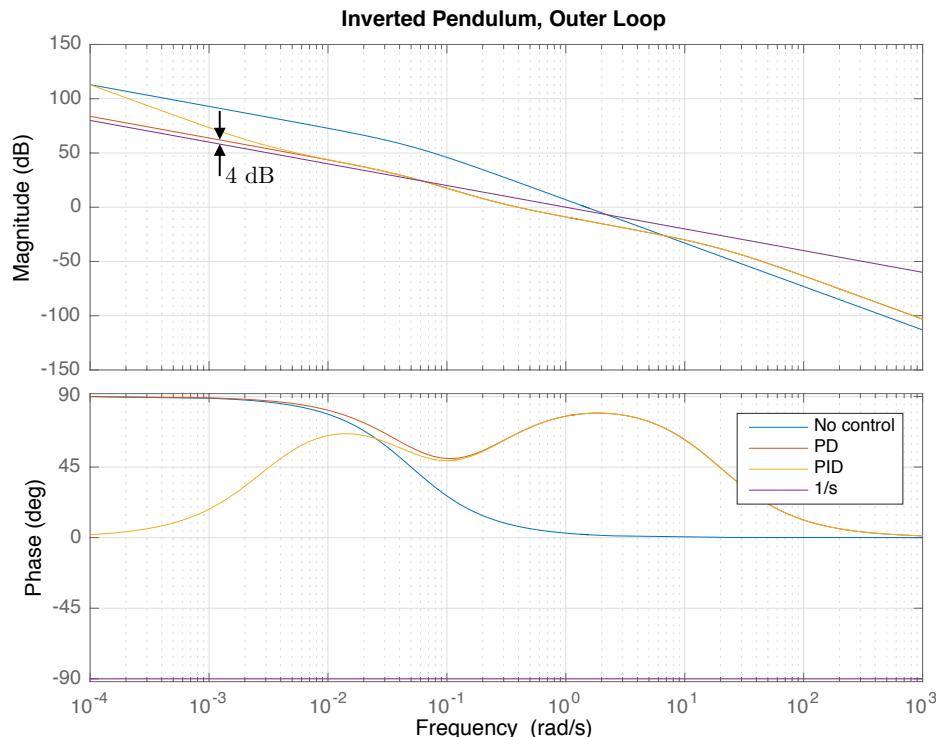
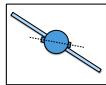


Figure 19.11: Bode plot for outer loop of the inverted pendulum, plant only, under PD control, and under PID control.

From Figure 19.11 it can be seen that under PID control the outer loop is type 2. Therefore, the tracking error will be zero. Under PD control the system is type 1 and the tracking error is related to how far the loop gain is above the $1/s$ line, which from Figure 19.11 it can be computed as $B_1 = 4$ dB. Therefore, from Equation (19.11) the steady state error satisfies

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{1}{M_v} A = 10^{-4/20} \cdot 2 = 1.2619.$$

19.4 Design Study C. Satellite Attitude Control



Homework Problem C.21

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.15.

- (a) To what percent error can the closed loop system under PD control track a step in $\theta^d(t)$ of 20 degrees?
- (b) If the input disturbance has frequency content below $\omega_{d_{in}} = 0.02$ 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 P control, and (3) the plant under PI control, using the control gains calculated in Homework C.15.

- (c) Under PI control, what is the percent tracking error of the outer loop for reference signals $\phi^d(t)$ with frequency content below $\omega_r = 0.01$ radians per second?
- (d) If all of the frequency content of the noise $n(t)$ is greater than $\omega_{no} = 1$ radians per second, what percentage of the noise shows up in the output signal ϕ , using PI 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 19.12.

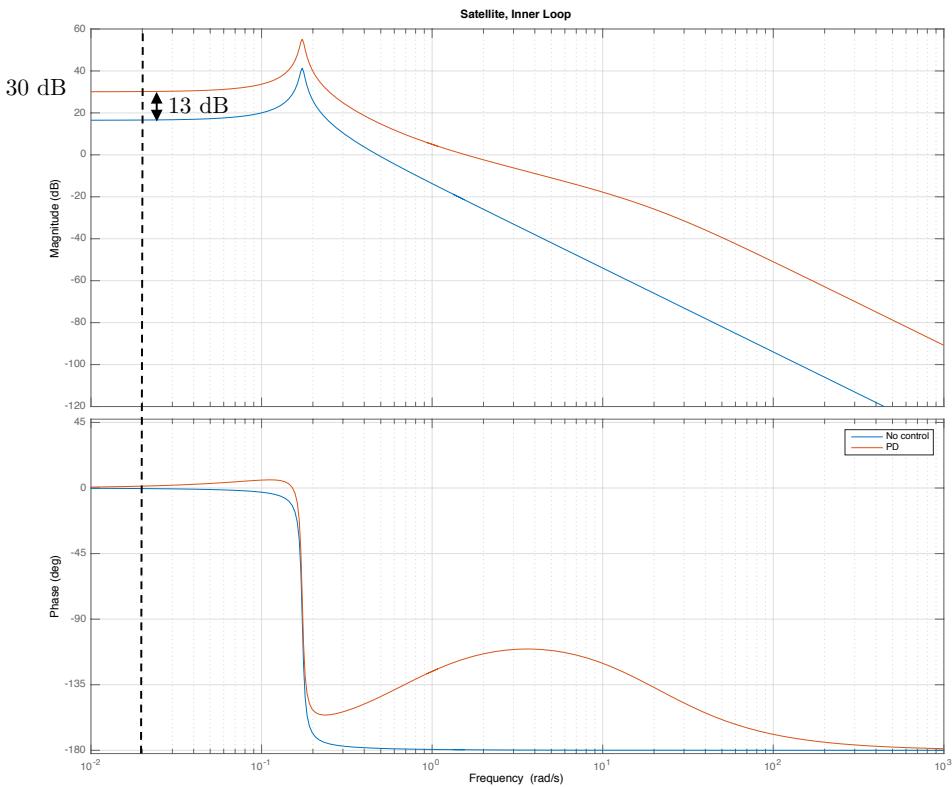


Figure 19.12: Bode plot for inner loop of the satellite attitude control, plant only, and under PD control.

From Figure 19.12 it can be seen that under PD control the inner loop is type 0 and that the loop gain as $\omega \rightarrow 0$ is $B_0 = 30$ dB. Therefore from Equation (19.10) the steady state error to a step of $\theta_d = 20$ degrees is

$$\lim_{t \rightarrow \infty} |e(t)| \leq \frac{1}{1 + M_p} A = \frac{1}{1 + 10^{30/20}} \frac{20\pi}{180} = 0.0107.$$

(b) If the input disturbance is below $\omega_{d_{in}} = 0.02$ rad/s, then the difference between the loop gain and the plant is $B_{d_{in}} = 13$ dB at $\omega_{d_{in}} = 0.02$ rad/s, therefore, from Equation (19.9) we see that

$$\gamma_{d_{in}} = 10^{-13/20} = 0.2239,$$

implying that 22% 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 19.13.

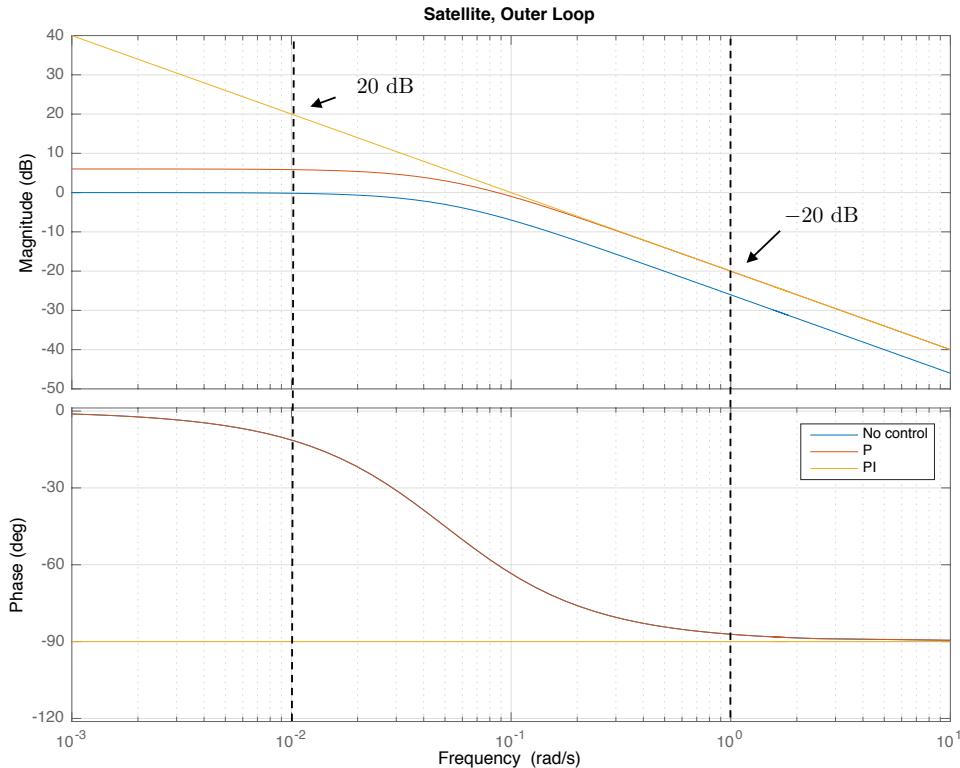


Figure 19.13: Bode plot for outer loop of the satellite attitude control, plant only, P control, and PI control.

From Figure 19.13 we see that below $\omega_r = 0.01$ rad/sec, the loop gain is above $B_r = 20$ dB. Therefore, from Equation (19.5) we have that

$$|e(t)| \leq \gamma_r |r(t)|,$$

where $\gamma_r = 10^{-20/20} = 0.1$, which implies that the tracking error will be 10% of the magnitude of the input.

(d) For noise greater than $\omega_{no} = 1$ rad/sec, we see from Figure 19.13 that $B_n = 20$ dB. Therefore, $\gamma_n = 10^{-10/20} = 0.1$ which implies that 10% of the noise will show up in the output signal.

Notes and References

Chapter 20

Stability and Robustness Margins

20.1 Theory

20.1.1 Phase and Gain Margins

In the previous chapter we did not talk about closed loop stability. To achieve the objectives discussed in Chapter 19, 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 20.1.

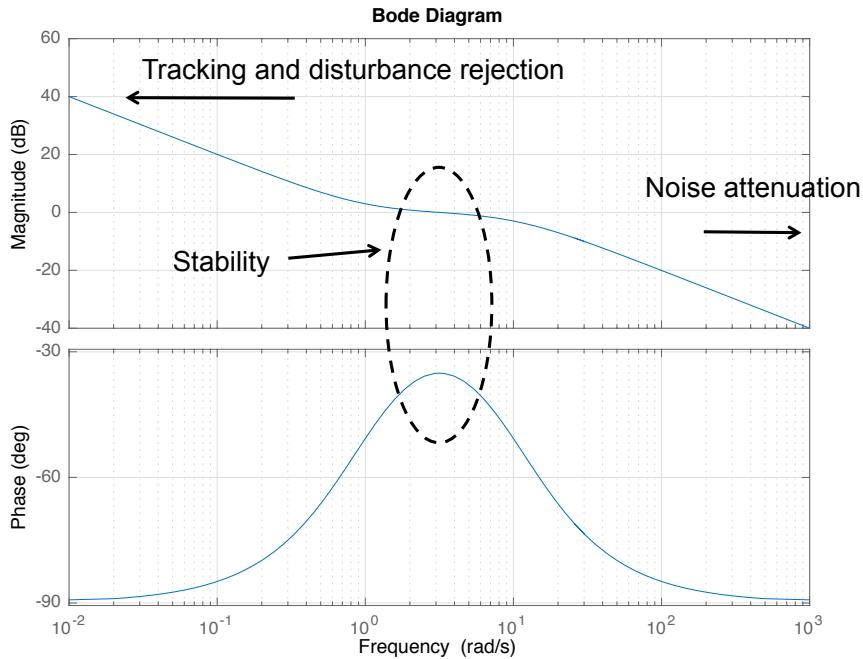


Figure 20.1: The shape of the magnitude plot primarily determines the tracking and disturbance rejection characteristics at low frequencies, and the noise attenuation at high frequencies. Stability is primarily determined by the phase at the crossover frequency.

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 (19.3). 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 20.2.

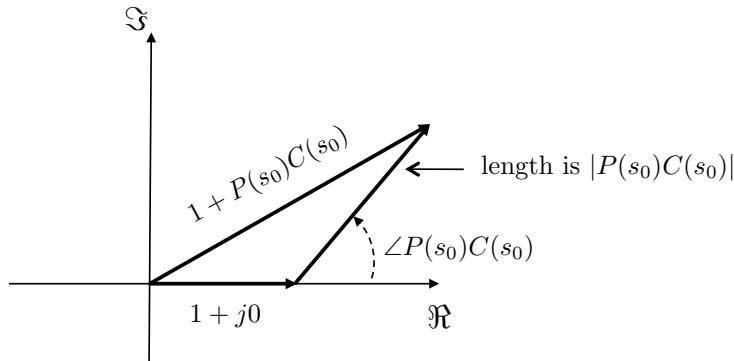


Figure 20.2: The plot of $1 + P(s_0)C(s_0)$ in the complex plane, is the sum of the complex number $1 + j0$ and $P(s_0)C(s_0)$.

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 20.3.

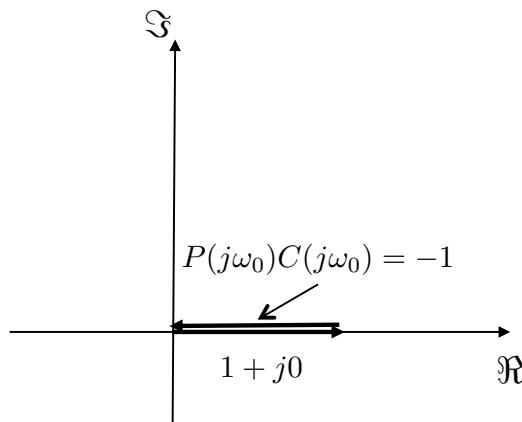


Figure 20.3: The plot of $1 + P(j\omega_0)C(j\omega_0)$ in the complex plane when $P(j\omega_0)C(j\omega_0) = -1$.

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 20.4 depicts the phase margin on the Bode plot.

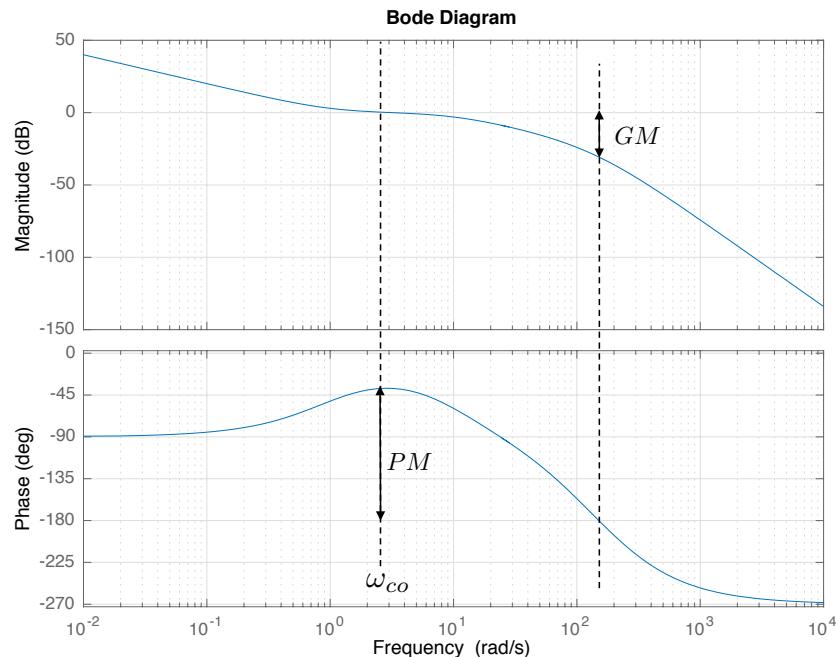


Figure 20.4: The phase margin PM is difference between the phase of $P(j\omega)C(j\omega)$ and -180 degrees at the crossover frequency ω_{co} . The gain margin GM is the magnitude of $P(j\omega)C(j\omega)$ when the phase crosses -180 degrees.

Another way to understand the phase margin is shown in Figure 20.5. The phase margin indicates the angular distance that $P(j\omega_{co})C(j\omega_{co})$ is from -1 .

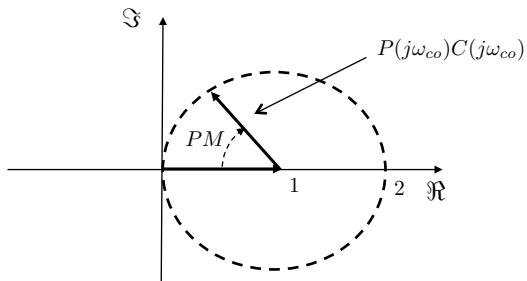


Figure 20.5: Alternative graphical view of the phase margin PM .

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 20.4. An alternative view is shown in Figure 20.6.

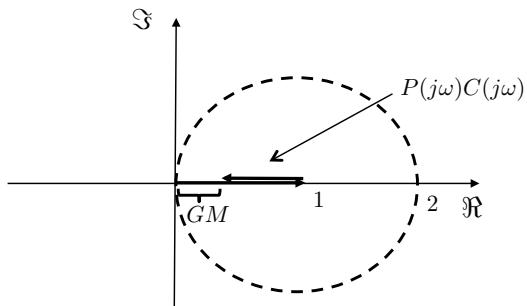


Figure 20.6: Alternative graphical view of the gain margin GM .

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 $GM = 10 \text{ dB}$ would be considered very good for most applications.

20.1.2 Open and Closed Loop Frequency Response

From Equation (19.1), 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 20.5, 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 20.7 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 $\zeta = 0.0707$, as shown in Figure 18.9. 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 a similar role for general systems that the natural frequency ω_n plays for second order systems. Figure 20.7 also shows the corresponding step response for the closed loop system. The step response looks similar to the step response for second order systems shown in Figure ??, where again, the step response for $PM = 60$ degrees in Figure 20.7 is roughly equivalent to the step response for $\zeta = 0.707$ in Figure ??.

RWB: Also note that the size of the control signal $u(t)$ is also determined by the crossover frequency. A large crossover frequency will result in a larger control signal $u(t)$. Therefore, saturation constraints on $u(t)$ will limit the size of ω_{co} . Show some plots to illustrate...

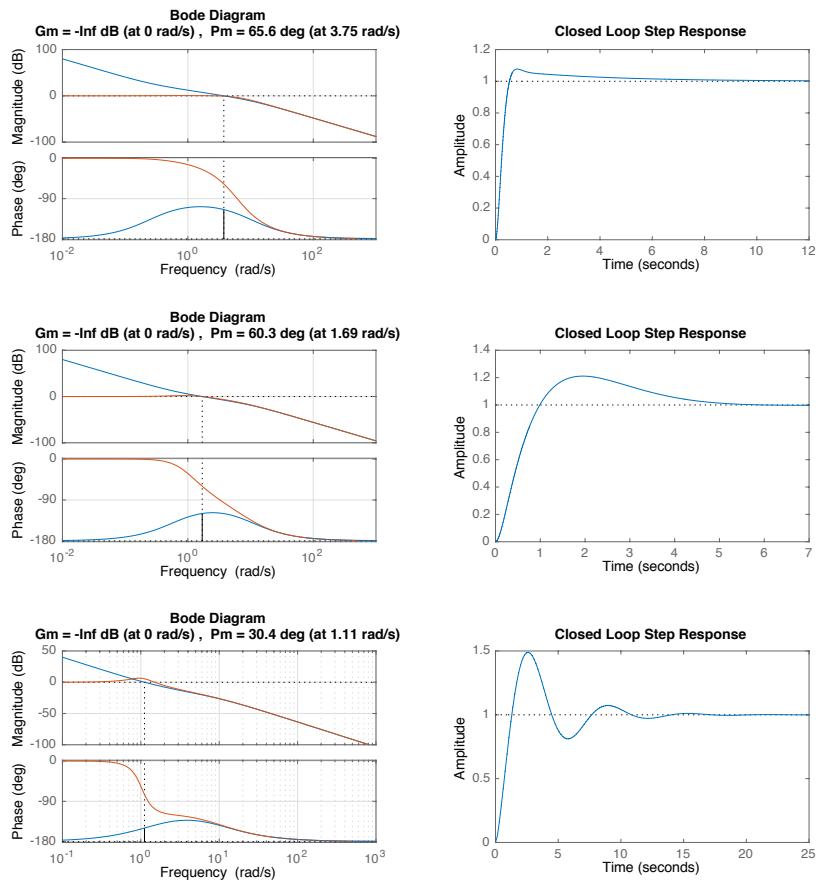


Figure 20.7: Closed loop Bode plot superimposed on the open loop Bode plot, together with the closed loop step response, for different phase margins.

20.1.3 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*

$$\angle G(j\omega_0) = \frac{1}{\pi} \int_{-\infty}^{\infty} \left(\frac{dM}{du} \right) W(u) du, \quad (20.1)$$

where

$$\begin{aligned} M &= 20 \log_{10} |G(j\omega)|, \\ u &= \log_{10} \left| \frac{\omega}{\omega_0} \right|, \\ W(u) &= \log_{10} \left(\coth \frac{|u|}{2} \right). \end{aligned}$$

The term $\frac{dM}{du}$ in Equation (20.1) is the slope of the magnitude of $G(j\omega)$ on the Bode plot. A plot of $W(u)$ is shown in Figure 20.8, 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 .

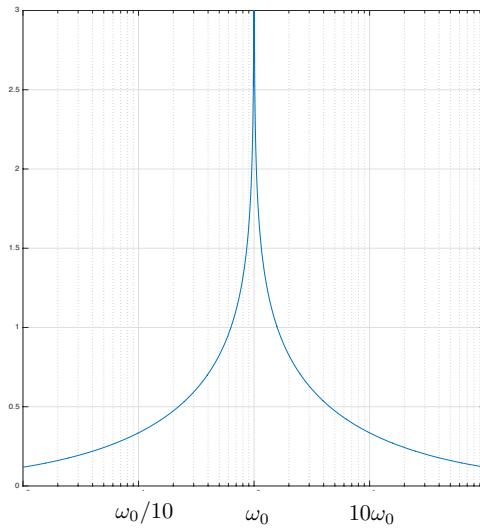


Figure 20.8: The weighting function $W(u)$ in the Bode gain-phase relationship heavily weights the slope around ω_0 .

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 (20.1) as follows:

- Slope of $20 \log_{10} |G(j\omega_0)| \approx +20$ dB/dec $\implies \angle G(j\omega_0) \approx +90$ deg,
- Slope of $20 \log_{10} |G(j\omega_0)| \approx 0$ dB/dec $\implies \angle G(j\omega_0) \approx 0$ deg,
- Slope of $20 \log_{10} |G(j\omega_0)| \approx -20$ dB/dec $\implies \angle G(j\omega_0) \approx -90$ deg,
- Slope of $20 \log_{10} |G(j\omega_0)| \approx -40$ dB/dec $\implies \angle G(j\omega_0) \approx -180$ deg,
- Slope of $20 \log_{10} |G(j\omega_0)| \approx -60$ dB/dec $\implies \angle G(j\omega_0) \approx -270$ 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 will never use Equation (20.1) 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 20.9. 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.

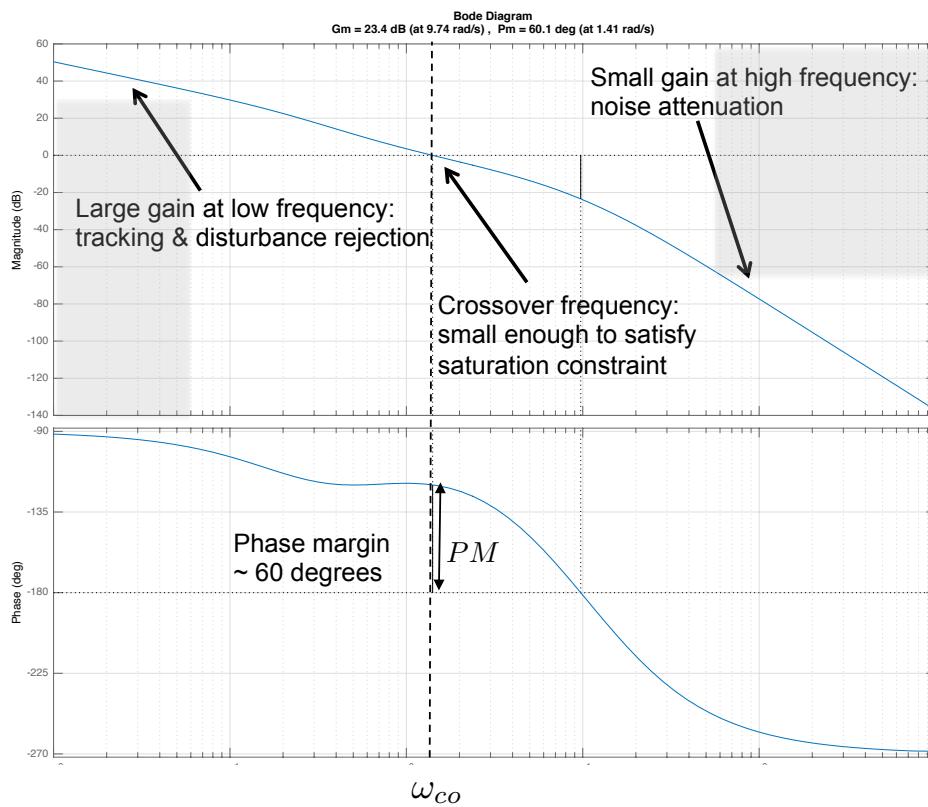
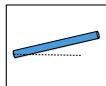


Figure 20.9: The ideal loopshape is large at low frequencies to achieve good tracking and disturbance rejection, is small at high frequencies to attenuate sensor noise, has a shallow slope at the crossover frequency to achieve a good phase margin, and has a small enough crossover frequency to satisfy the input saturation constraints.

20.2 Design Study A. Single Link Robot Arm



Homework Problem A.22

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 PD and 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.15.

Solution

The Matlab code used to generate the plots is shown below.

```

1 % transfer function for robot arm
2 Plant = tf([2/P.m/P.ell^2],[1, 2*P.b/P.m/P.ell^2, 0]);
3
4 % transfer function for PD and PID control
5 C_pd = tf([(P.kd_th+P.kp_th*P.tau),P.kp_th],[P.tau,1]);
6 C_pid = tf([(P.kd_th+P.kp_th*P.tau),...
7             (P.kp_th+P.ki_th*P.tau),P.ki_th],[P.tau,1,0]);
8
9 % margin and bode plot for PD control
10 figure(1), clf, margin(Plant*C_pd), grid on
11 hold on
12 bode(Plant*C_pd/(1+Plant*C_pd))
13 legend('Open Loop', 'Closed Loop')
14
15 % margin and bode plot for PID control
16 figure(2), clf, margin(Plant*C_pid), grid on
17 hold on
18 bode(Plant*C_pid/(1+Plant*C_pid))
19 legend('Open Loop', 'Closed Loop')
```

The transfer function for the plant is defined in Line 1. The transfer function for the PD controller is

$$C_{PD}(s) = k_P + \frac{k_D s}{\tau s + 1} = \frac{(\tau s + 1)k_P + k_D s}{\tau s + 1} = \frac{(k_D + \tau k_P)s + k_P}{\tau s + 1},$$

and is defined in Line 3. The transfer function for the PID controller is

$$\begin{aligned} C_{PID}(s) &= k_P + \frac{k_I}{s} + \frac{k_D s}{\tau s + 1} = \frac{s(\tau s + 1)k_P + (\tau s + 1)k_I + k_D s^2}{s(\tau s + 1)} \\ &= \frac{(k_D + \tau k_P)s^2 + (k_P + \tau k_I)s + k_I}{\tau s + 1}, \end{aligned}$$

and is defined in Line 3. 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 Lines 9 and 15, and the closed loop transfer functions are $PC/(1+PC)$ as defined in Lines 11 and 17. The results of this code are shown in Figure 20.10 and ??.

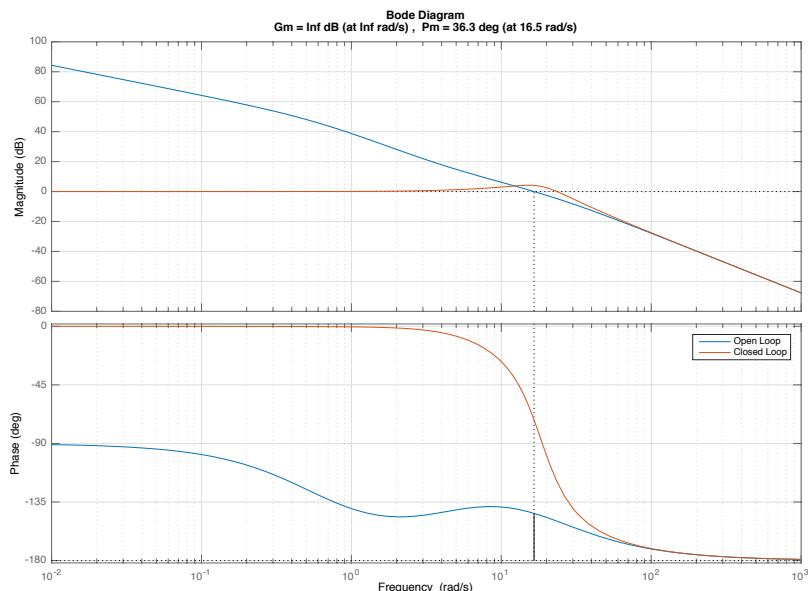


Figure 20.10: The `margin` plot of the open loop system and the `bode` plot of the closed loop system, of the single link robot arm under PD control.

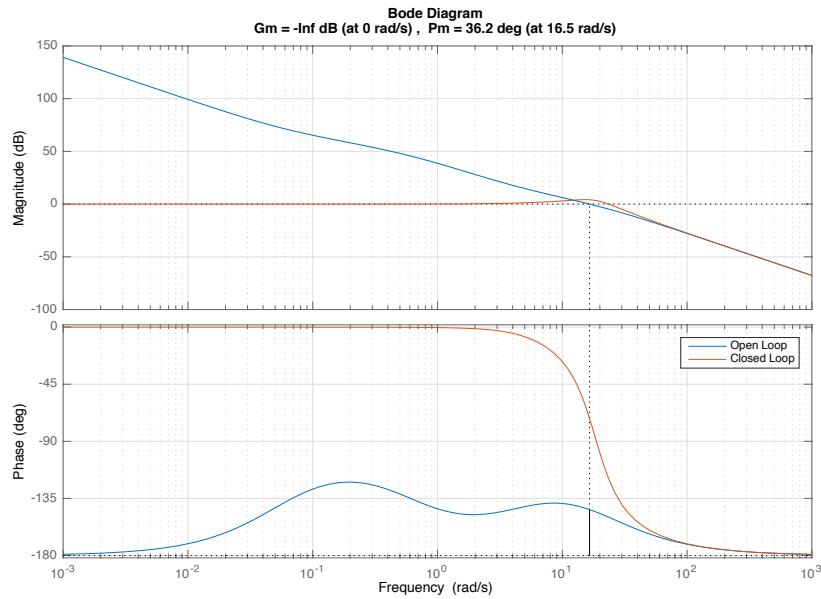
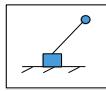


Figure 20.11: The margin plot of the open loop system and the bode plot of the closed loop system, of the single link robot arm under PID control.

As seen from Figure 20.10 the bandwidth for PD control is approximately 25 rad/sec, which is slightly larger than the cross over frequency of 16.5 rad/sec. The larger bandwidth is due to the small phase margin of $PM = 36.3$ degrees. Similarly, Figure 20.11 indicates that the bandwidth for PID control is approximately 30 rad/sec, which is slightly larger than the cross over frequency of 16.5 rad/sec. The larger bandwidth is due to the small phase margin of $PM = 36.2$ degrees.

20.3 Design Study B. Inverted Pendulum



Homework Problem B.22

For this problem, use the gains found in HW B.15.

- (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([-1/P.m2/P.ell],...
3            [1, 0, -(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_pd = tf([(P.kd_th+P.tau*P.kp_th), P.kp_th], [P.tau, 1]);
6 C_out_pid = tf([(P.kd_z+P.kp_z*P.tau),...
7                  (P.kp_z+P.ki_z*P.tau),P.ki_z],[P.tau,1,0]);
8
9 % margin and bode plots
10 figure(1), clf, margin(P_in*C_in_pd), grid on, hold on
11 bode(P_in*C_in_pd/(1+P_in*C_in_pd))
12 margin(P_out*C_out_pid)
13 bode(P_out*C_out_pid/(1+P_out*C_out_pid))
14 legend('Open Loop-Inner', 'Closed Loop-Inner',...
15        'Open Loop-Outer', 'Closed Loop-Outer')
```

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 10–14. The results of this code are shown in Figure 20.12.

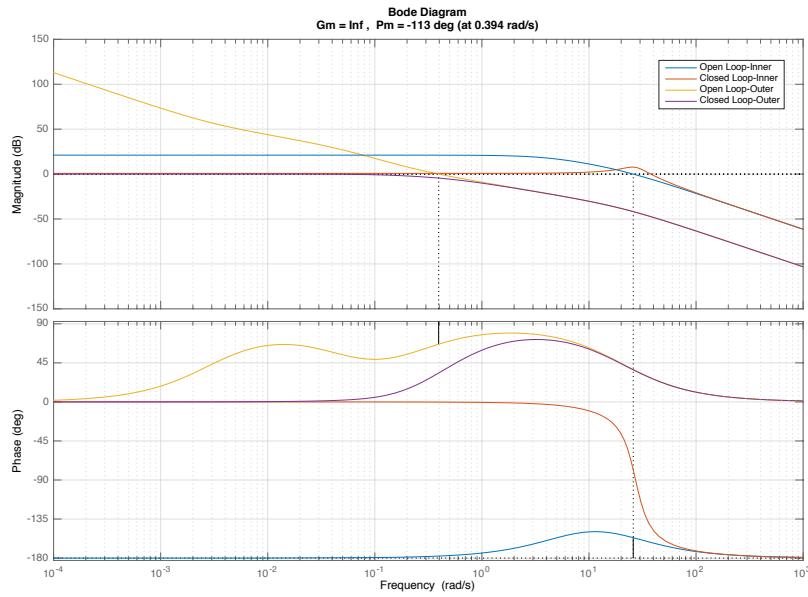
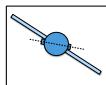


Figure 20.12: The `margin` and `bode` plots for the the open and closed loop systems of both the inner and outer loops of the inverted pendulum system.

As seen from Figure 20.12 the bandwidth of the inner loop is approximately 40 rad/sec, which is slightly larger than the cross over frequency of 26 rad/sec. The larger bandwidth is due to the small phase margin of $PM = 23$ degrees. Similarly, Figure 20.12 indicates that the bandwidth of the outer loop is approximately 0.24 rad/sec, which is slightly smaller than the cross over frequency of 0.39 rad/sec. The smaller bandwidth is due to the large phase margin of $PM = 113$ degrees.

The bandwidth separation between the inner and outer loop is almost two decades and successive loop closure is justified by the fact that the bode plot of the inner loop is approximately one far beyond the cross over frequency of the outer loop.

20.4 Design Study C. Satellite Attitude Control



Homework Problem C.22

For this problem, use the gains found in HW C.15.

- 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?
- 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?
- 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],[1,P.b/P.Jp]);
4 C_in_pd = tf([(P.kd_th+P.tau*P.kp_th),P.kp_th], [P.tau, 1]);
5 C_out_pi = tf([P.kp_phi, P.ki_phi],[1,0])
6
7 % margin and bode plots
8 figure(1), clf, margin(P_in*C_in_pd), grid on, hold on

```

```

9 bode(P_in*C_in_pd/(1+P_in*C_in_pd))
10 margin(P_out*C_out_pi)
11 bode(P_out*C_out_pi/(1+P_out*C_out_pi))
12 legend('Open Loop-Inner', 'Closed Loop-Inner',...
13           'Open Loop-Outer', 'Closed Loop-Outer')

```

The transfer functions for the inner and outer loop plants and controller are defined in Lines 2–5. For this problem we plot both the inner and outer loop frequency response on the same Bode plot, as implemented in Lines 8–12. The results of this code are shown in Figure 20.13.

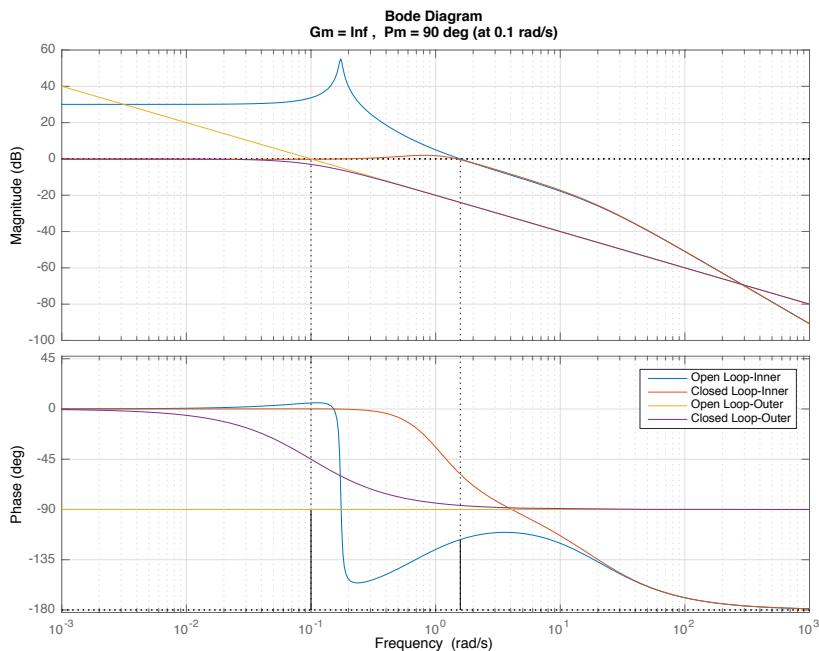


Figure 20.13: The `margin` and `bode` plots for the the open and closed loop systems of both the inner and outer loops of the satellite attitude control system.

As seen from Figure 20.13 the bandwidth of the inner loop is approximately the cross over frequency of 1.59 rad/sec, which is due to the near-ideal phase margin of $PM = 63$ degrees. Similarly, Figure 20.13 indicates that the bandwidth of the outer loop is slightly smaller than then the cross over fre-

quency of 0.1 rad/sec, with a phase margin of $PM = 90$ degrees.

The bandwidth separation between the inner and outer loop is over a decade and successive loop closure is justified by the fact that the bode plot of the inner loop is approximately one far beyond the cross over frequency of the outer loop.

Notes and References

Chapter 21

Compensator Design

21.1 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. The idea is that the controller $C(s)$ will be composed of several building blocks as

$$C(s) = C_1(s)C_2(s) \cdots C_q(s).$$

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. There are 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 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 only effect 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.

21.1.1 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 21.1 for different values of k_P .

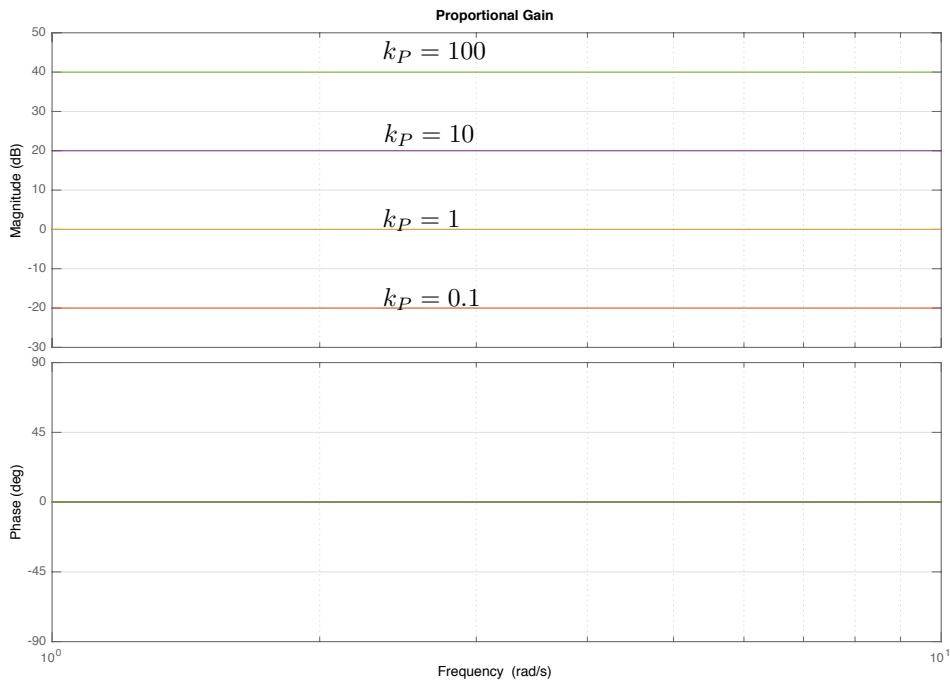


Figure 21.1: The Bode plot for proportional gain at different values of k_P . The proportional gain adds a constant to the loopshape at all frequencies, and does not affect the phase at any frequency.

21.1.2 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 21.2. 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 21.2 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 $\omega \ll \omega_{co}$.

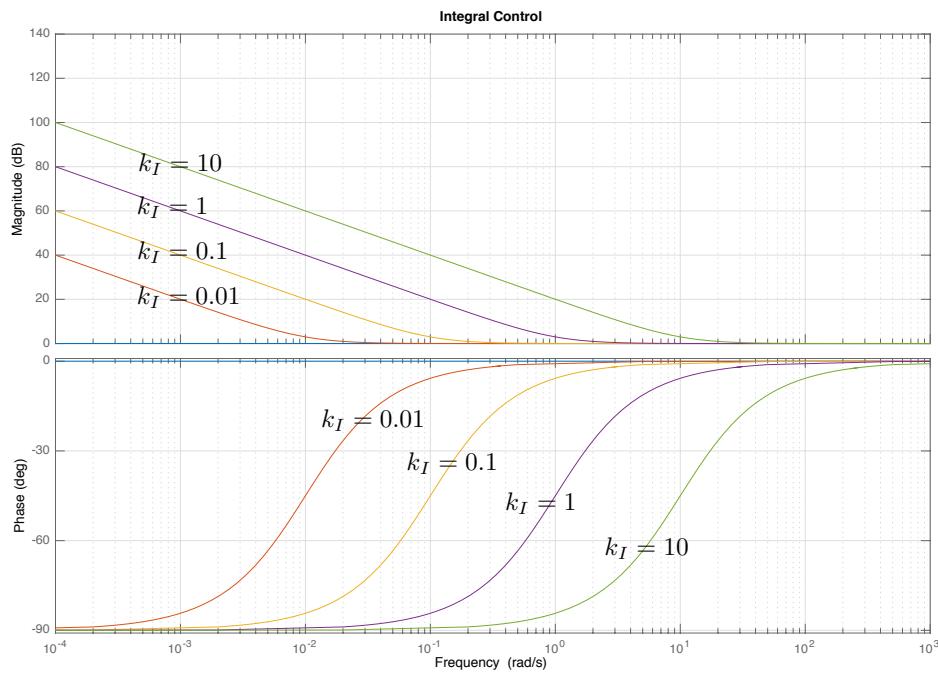


Figure 21.2: The Bode plot for integral control at different values of k_I . Integral control changes the system type by decreasing the slope of the loopshape by -20 dB/dec for frequencies below k_I , while leaving high frequencies unaffected.

21.1.3 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 21.3 for different values of p . It is evident from Figure 21.3 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.

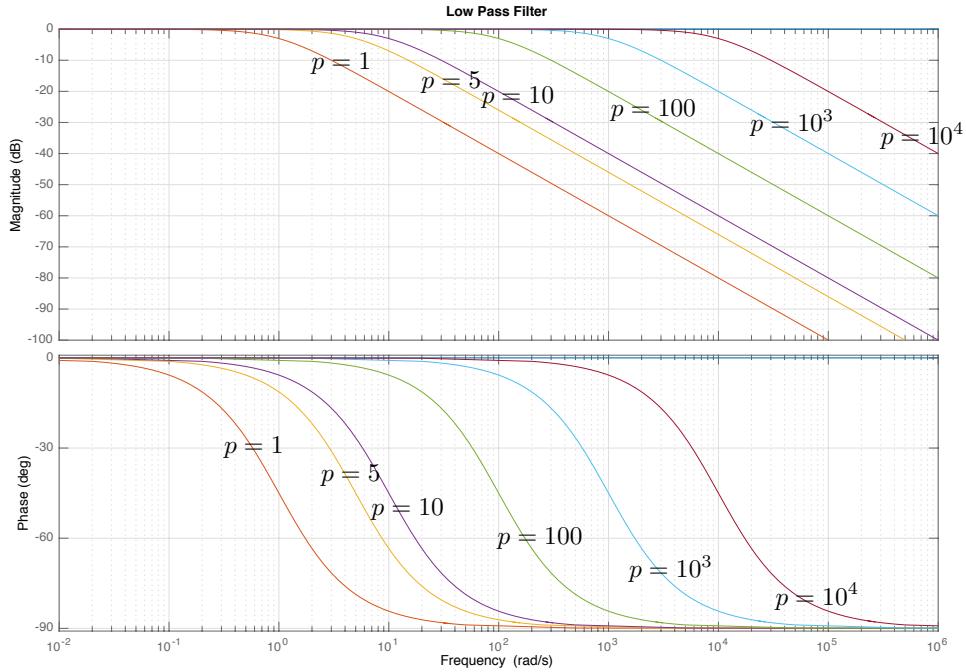


Figure 21.3: The Bode plot for a low pass filter at different values of the pole p . A Low pass filter decreasing the slope of the loopshape by -20 dB/dec for frequencies above p , while leaving low frequencies unaffected.

21.1.4 Building Block #4: Phase Lag Filter

The fourth basic building block for loopshaping design is a phase lag filter where

$$C_{Lag}(s) = \frac{s + z}{s + p},$$

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 21.4 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 to 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.

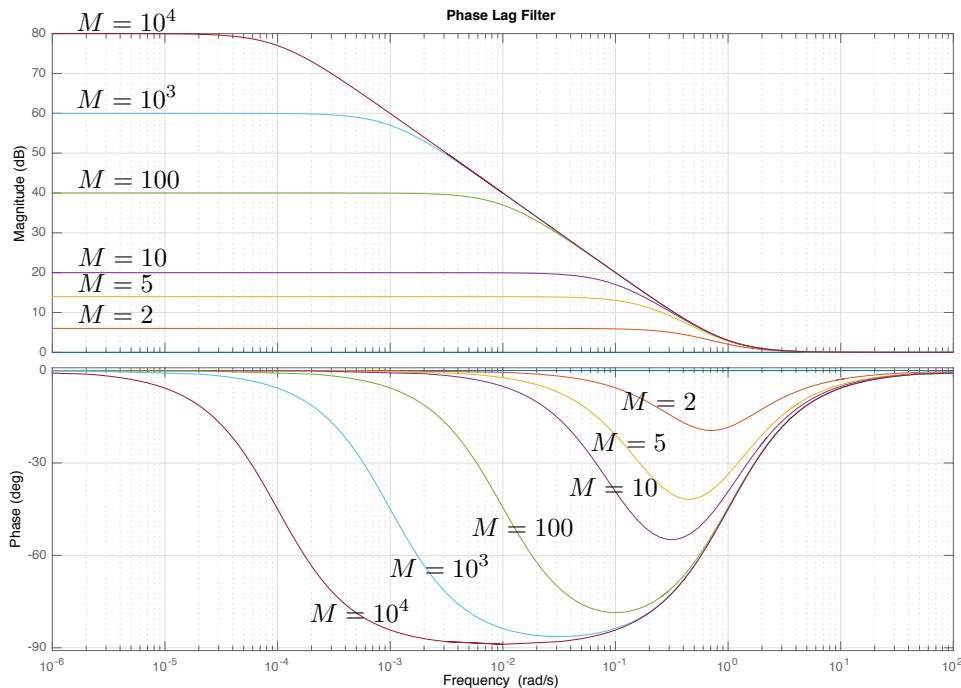


Figure 21.4: The Bode plot for a phase lag filter when the zeros is at $z = 1$ and the pole is at $p = z/M$, for different values of M . A phase lag filter increases the loop gain at low frequencies while leaving high frequencies unaffected.

21.1.5 Building Block #5: Phase Lead Filter

The final basic building block for loopshaping design that we will discuss is a phase lead filter where

$$C_{Lead}(s) = \left(\frac{p}{z}\right) \left(\frac{s+z}{s+p}\right),$$

where $p > z > 0$. Note that the phase lag filter is constructed so that

$$\lim_{s \rightarrow 0} \frac{p s + z}{z s + p} = 1.$$

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 21.5 when $z = 1$ and when $p = Mz$ for different values of pole-zero 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 21.5 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 21.5, 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.

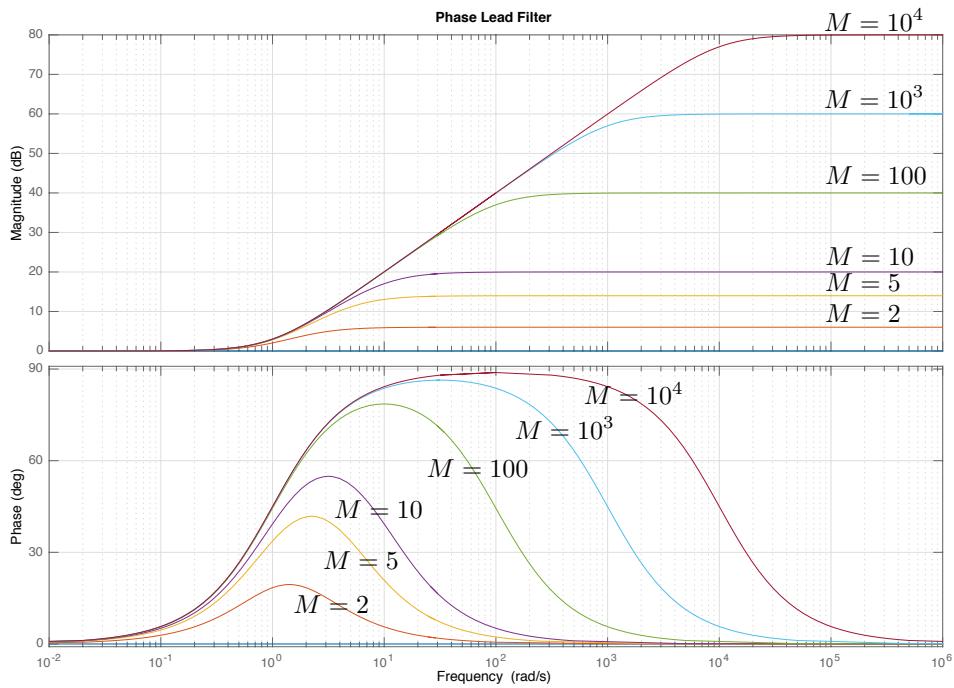


Figure 21.5: The Bode plot for a phase lead filter when the zeros is at $z = 1$ and the pole is at $p = Mz$, for different values of M . A phase lead filter increases the phase over frequencies between z and p . It also increases the loop gain a high frequencies.

21.1.6 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) Track 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 21.6. To track a ramp within $\gamma_1 = 0.03$ requires that the loop gain be above

$$B_1 = 20 \log_{10} \left| \frac{1}{0.03} \right| - 20 \log_{10} |\omega|$$

as $\omega \rightarrow 0$. Therefore, to satisfy this specification, we need to find C so that the Bode plot for PC gets above the green line on the top left of Figure 21.6.

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 21.6.

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 21.6.

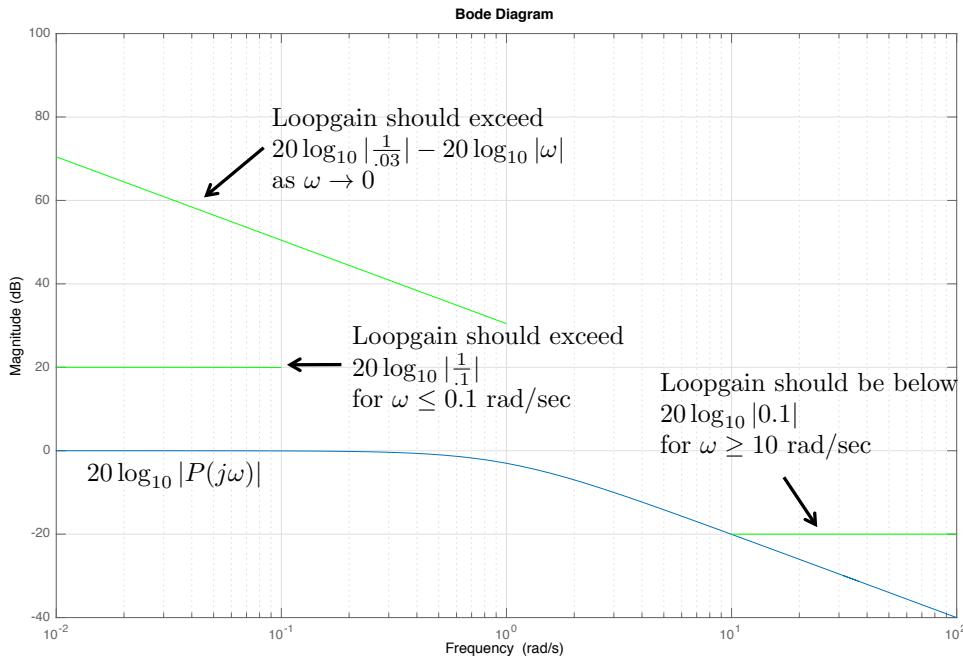


Figure 21.6: The Bode plot for the open loop plant in Example 1, together with the design specifications.

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 21.7 shows the loopgain of PC when

$$C(s) = C_P(s)C_{PI}(s) = (2) \left(\frac{s + 0.4}{s} \right).$$

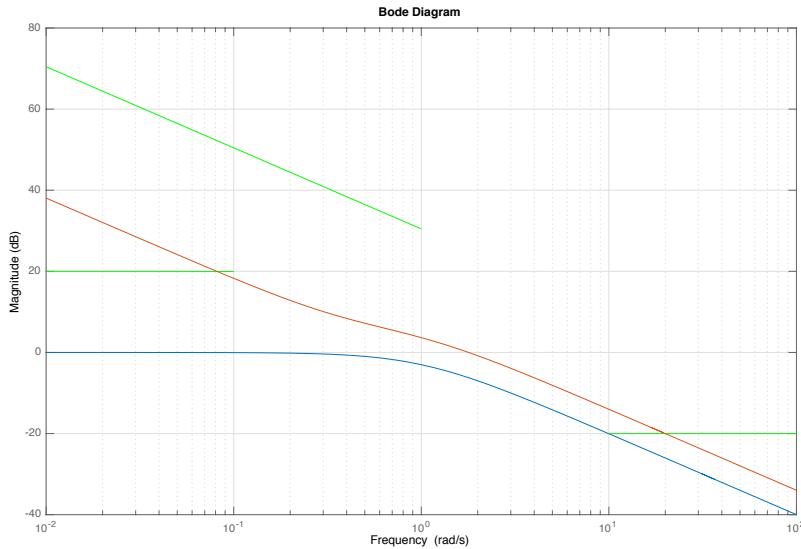


Figure 21.7: The Bode gain plot of P and PC when C includes proportional and integral control.

To increase the loop gain at low frequencies to get above the $1/s$ line as $\omega \rightarrow 0$, a phase lag filter can be added. Figure 21.8 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 21.8 we see that the closed loop system will satisfy the tracking and input disturbance specifications.

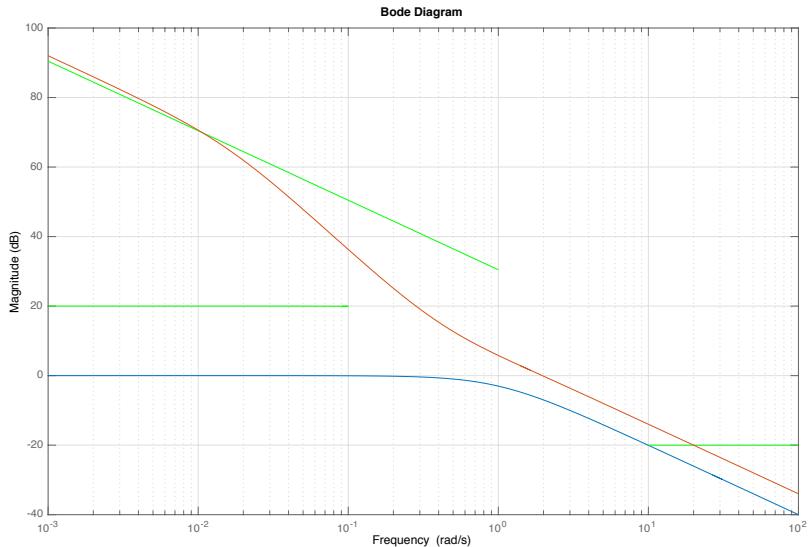


Figure 21.8: The Bode gain plot of P and PC when C includes proportional, integral, and lag filters.

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 21.9 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 21.9.

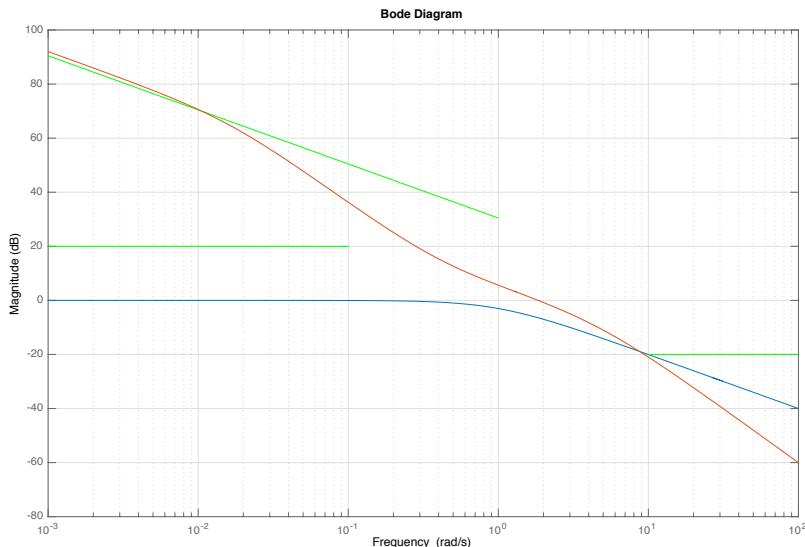


Figure 21.9: The Bode gain plot of P and PC when C includes proportional, integral, lag, and low pass filters.

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)), 'g')
17 % steady state tracking of ramp
18

```

```

19 gamma_1 = .03;
20 w = logspace(-4, 0);
21 plot(w,20*log10(1/gamma_1)-20*log10(w), 'g')
22
23 % Control Design
24 C = 1;
25 % proportional control
26 kp = 2;
27 C = C*kp;
28 % integral control
29 k_I = .4;
30 Integrator = tf([1,k_I], [1,0]);
31 C = C*Integrator;
32 % phase lag
33 z = .8;
34 p = z/50;
35 Lag = tf([1,z], [1,p]);
36 C = C*Lag;
37 % low pass filter
38 p = 5;
39 LPF = tf(p, [1,p]);
40 C = C*LPF;
41
42 bodemag(Plant*C) % update plot
43

```

21.1.7 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

$$\dot{z}_C = A_C z_C + B_C e \quad (21.1)$$

$$u = C_C z_C + D_C e \quad (21.2)$$

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

$$\begin{aligned}\dot{z}_C &= \begin{pmatrix} -5.016 & 0.08 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} z_C + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} e \\ u &= (10 \ 12 \ 3.2) z_C.\end{aligned}$$

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);
3     y      = in(2);
4     t      = in(3);
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 is initialized to the zero vector. The error signal is found in Line 11. The control dynamics (21.1) are integrated using N Euler steps in Lines 13–16, and the control output equation is computed in Line 18.

21.1.8 Prefilter Design

$$F(s) = \left(\frac{p_s p_e}{z^2} \right) \left(\frac{(s+z)^2}{(s+p_s)(s+p_e)} \right),$$

where $z = \sqrt{p_s p_e}$, the median frequency on the Bode plot.

If ω_s is the desired start frequency, and M is the separation between the two poles, then $p_s = \omega_s$, $p_e = M\omega_s$ and $z = \omega_s\sqrt{M}$.

Therefore

$$F(s) = \frac{s^2 + 2\sqrt{M}\omega_s s + M\omega_s^2}{s^2 + (M+1)\omega_s s + M\omega_s^2}.$$

Note that

$$\lim_{s \rightarrow 0} F(s) = \lim_{s \rightarrow \infty} F(s) = 1.$$

A notch filter for different values of M is shown in Figure 21.10.

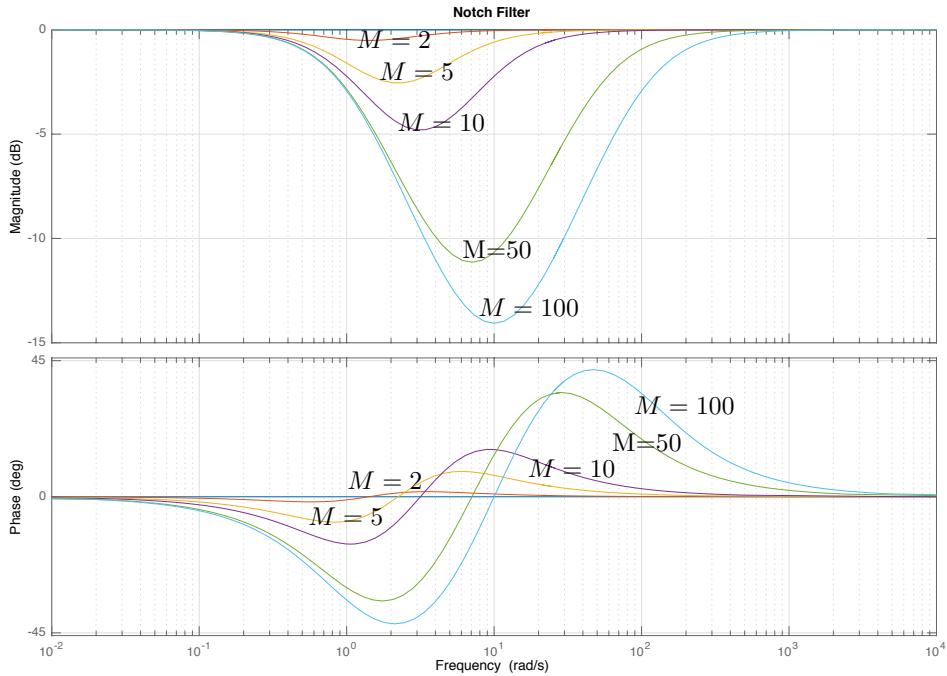


Figure 21.10: The Bode plot for a notch filter when $\omega_s = 1$, and for different values of M . A notch is often used as a prefilter on the reference command r .

The prefilter

$$R_f(s) = F(s)R(s)$$

is first converted to continuous time state space equations using, for example, control canonical form, to produce the equations

$$\dot{z}_F = A_F z_F + B_F r \quad (21.3)$$

$$r_f = C_F z_F + D_F r \quad (21.4)$$

where the reference $r(t)$ is the input to the state space equations, and the filtered reference signal $r_f(t)$ is the output.

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);

```

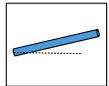
The prefilter is then implemented by integrating the differential equations between sample times. If T_s is the sample time, and $N = 10$ Euler steps are used between each sample, then Matlab code that implements the controller plus prefilter is given by

```

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    % prefilter the reference command
13    N = 10; % number of Euler integration steps per sample
14    for i=1:N, % solve differential equation for prefilter
15        z_F = z_F + P.Ts/N*( P.A_F*z_F + P.B_F*r );
16    end
17    % output equation for the prefilter
18    r_filtered = P.C_F*z_F + P.D_F*r;
19    % error signal uses filtered reference
20    error = r_filtered - y;
21    N = 10; % number of Euler integration steps per sample
22    for i=1:N, % solve differential equation for controller
23        z_C = z_C + P.Ts/N*( P.A_C*z_C + P.B_C*error );
24    end
25    % output equation for the controller
26    u = P.C_C*z_C + P.D_C*error;
27 end

```

21.2 Design Study A. Single Link Robot Arm



Homework Problem A.23

For this homework assignment we will use loopshaping to improve the PID controllers developed in HW A.15. Let $C_{pid}(s)$ be the PID controller designed

in HW A.15. 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 200 radians/sec.
 - (3) Phase margin that is approximately $PM = 60$ degrees.
- (b) Add zero mean Gaussian noise with variance $\sigma^2 = ??$ to the Simulink diagram developed in HW A.15.
- (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 21.11. To improve the tracking and disturbance rejection by a factor of 10 references 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 21.11. To improve noise attenuation by a factor of 10 for frequencies above $\omega_n = 200$ 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 21.11.

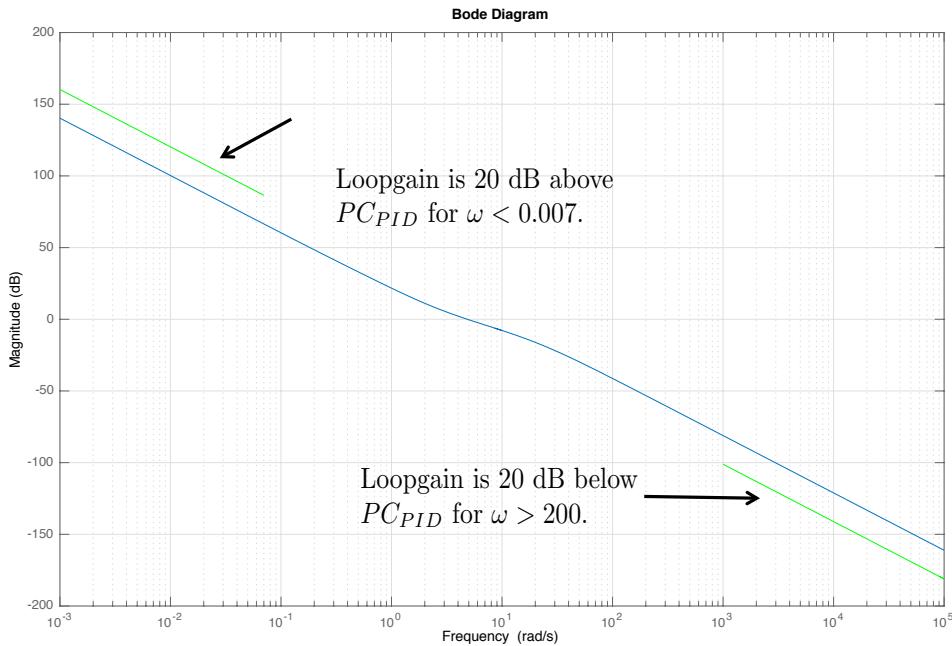


Figure 21.11: The Bode plot for the open loop plant in HW A.23, together with the design specifications.

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 21.8 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 21.12 we see that the closed loop system will satisfy the tracking and input disturbance specifications.

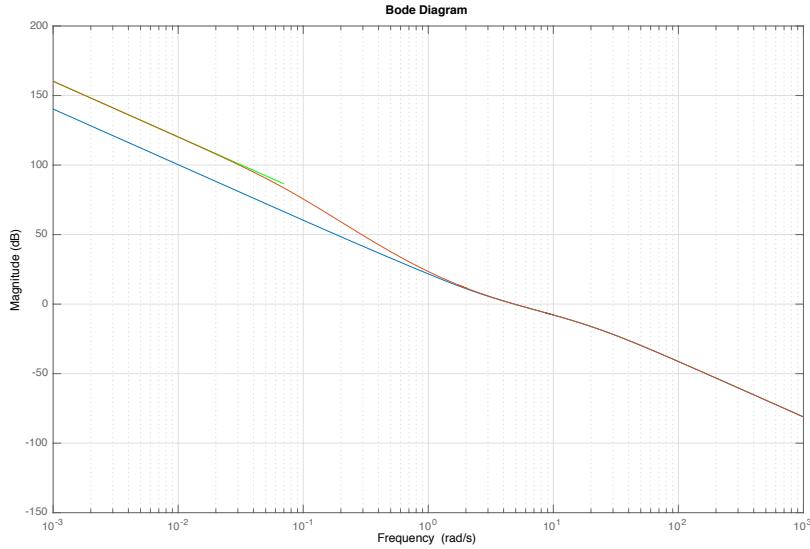


Figure 21.12: The Bode plot for HW A.23 where $C = C_{PID}C_{lag}$.

The phase margin after adding the lag filter is $PM = 41.2$ 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 21.13.

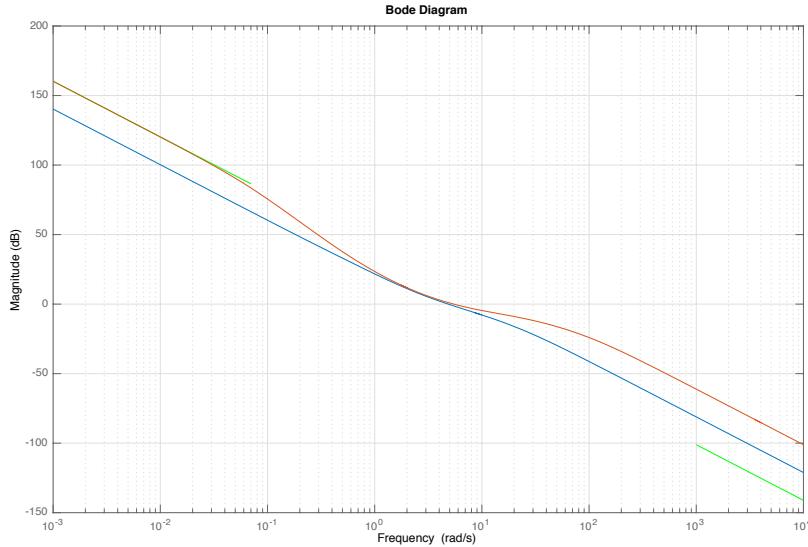


Figure 21.13: The Bode plot for HW A.23 where $C = C_{PID}C_{lag}C_{lead}$.

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 21.14, 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 loop-gain in Figure ???. The phase margin is $PM = 61$ 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.

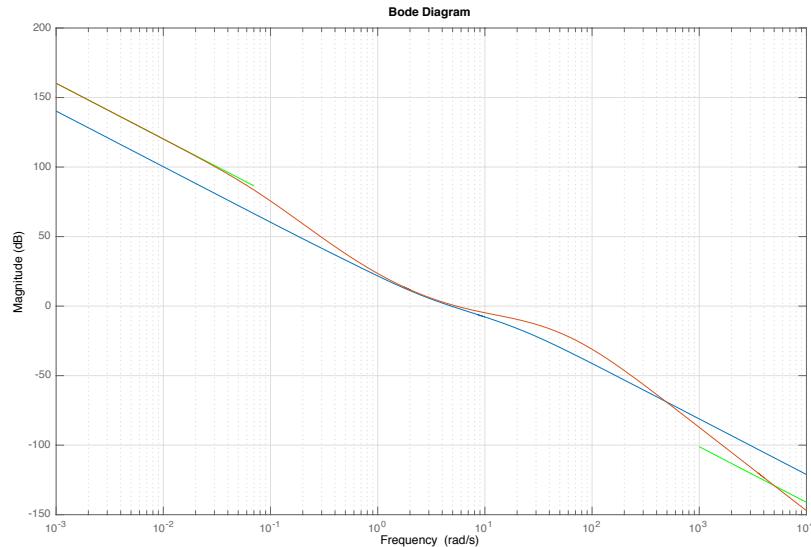


Figure 21.14: The Bode plot for HW A.23 where $C = C_{PID}C_{lag}C_{lead}C_{lpf}$.

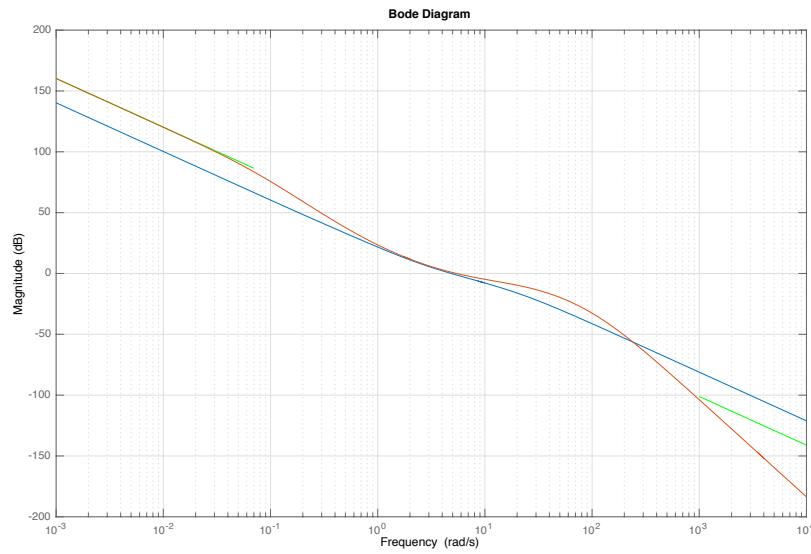


Figure 21.15: The Bode plot for HW A.23 where $C = C_{PID}C_{lag}C_{lead}C_{lpf}C_{lpf2}$.

Matlab code that implements this example is shown below.

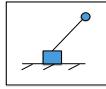
```

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)), 'g')
17
18 % steady state tracking of ramp
19 gamma_1 = .03;
20 w = logspace(-4, 0);
21 plot(w,20*log10(1/gamma_1)-20*log10(w), 'g')
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 p = z/50;
36 Lag = tf([1,z],[1,p]);
37 C = C*Lag;
38 % low pass filter
39 p = 5;
40 LPF = tf(p,[1,p]);
41 C = C*LPF;
42

```

```
43 bodemag(Plant*C) % update plot
```

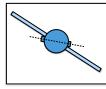
21.3 Design Study **B.** Inverted Pendulum



Homework Problem **B.23**

Solution

21.4 Design Study **C.** Satellite Attitude Control



Homework Problem **C.23**

Solution

Notes and References

Part VI

Design Studies

Design Problem

D. Mass Spring Damper

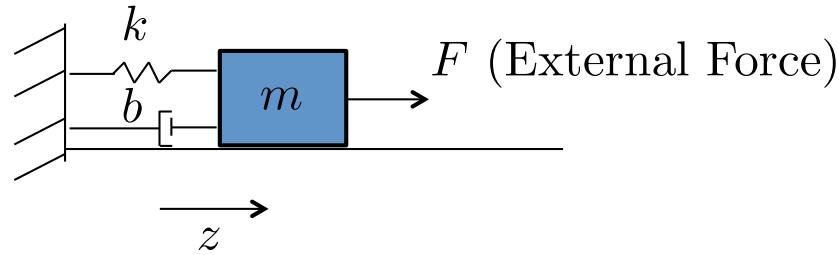


Figure .1: Mass Spring Damper

Figure .1 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 .1.

Assume the following physical constants: $m = 5 \text{ kg}$, $k = 3 \text{ Kg/s}^2$, $b = 0.5 \text{ Kg/s}$.

Homework D.1

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

Homework D.2

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.
- (d) Derive the equations of motion using the Euler-Lagrange equations.

Homework D.4

Modify the simulink model created in homework D.2 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 D.2.

Homework D.5

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

Homework D.6

- (a) Using the Laplace transform, convert from time domain to the s-domain.
- (b) Find the transfer function from the small deviation in force \tilde{F} to the small deviation in mass position \tilde{z} .
- (c) Draw the associated block diagram.

Homework D.7

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

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

Homework D.8

- (a) Given the open loop transfer function from force to position in problem D.6, find the open loop poles of the system, when the equilibrium position is $z_e = 0$.

- (b) Using PD control architecture shown in Figure ??, find the closed loop transfer function from z^c 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 $p_1 = -0.5$ and $p_2 = -0.2$.

Homework D.9

Using the gains found in Problem D.8, implement the PD control for the mass spring damper in Simulink and plot the step response. Turn in an image of the simulink block diagram, as well as the step response of the system.

Hint: Change the s-function that defines the dynamics so that the output is the configuration variable z . The Simulink block should look like that shown in Figure .2.

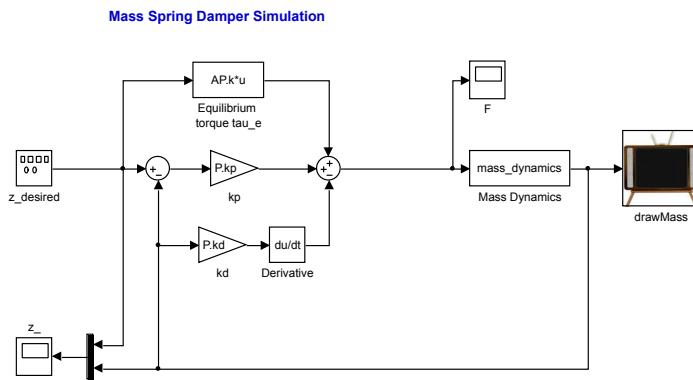


Figure .2: Simulink file for Homework D.9

Recall that the force in the transfer function model is the linearized

torque $\tilde{F} = F - F_e$. Therefore, the force applied to the mass is $F = F_e + \tilde{F}$. The equilibrium force must be added to the force computed by the PID controller as shown in Figure .2.

Homework D.10

- (a) For the mass spring damper, suppose that the design requirements are:

- Rise time: $t_r < 5$ seconds,
- Maximum percent overshoot: $M_p < 5\%$,
- Settling time: $t_s < 10$ seconds, within 1%.

Find the region in the s-plane where the poles would result in the design requirements being satisfied.

- (b) Select pole locations that satisfy the design requirements.
 (c) Using the PD architecture shown in Figure ??, find the proportional gain k_p and the derivative gain k_d that place the poles at the location selected in part (b).
 (d) Implement the PD controller in Simulink when $y^d(t)$ is a square wave of magnitude 0.5 meters and frequency 0.02 Hz. Verify that the step response satisfies the design specifications.

Homework D.12

Suppose that the size of the input force is limited to $F_{\max} = 2$ N and that the objective is to move the mass in steps from -0.5 meters to 0.5 meters.

- (a) Considering the equilibrium torque F_e , derive a bound \tilde{F}_{\max} on the linearized force $\tilde{\tau}$ such that $|\tilde{\tau}| \leq \tilde{\tau}_{\max}$ when the maximum possible equilibrium force is used.
 (b) Select the proportional gain k_P so that \tilde{F} just saturates when a step of size $A = 1$ meter is placed on z_d , i.e., a step command from $z^d = -0.5$ to $z^d = 0.5$.
 (c) Find the corresponding natural frequency and rise time, and find k_D that results in a damping ratio of $\zeta = 0.707$.

- (d) Modify the Simulink diagram developed in Problem D.10, to include a saturation block on the force F , and implement PD control with the revised gains.

Homework D.13

- (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.14

Adding an integrator to obtain PID control for the mass spring damper, put the characteristic equation in Evans 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. Implement the controller in Simulink and further tune k_I to get satisfactory response.

Homework D.15

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Change the parameter file `param.m` to specify actual parameters and the parameters known to the controller. Change the values of m , k , and b known to the controller by 5% to simulate imperfect knowledge of the plant.
- (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 desired angle θ^d .
- (c) Implement the PID controller designed in Problems D.14 using an m-function called `mass_ctrl.m`. Use the dirty derivative gain of

$\tau = 0.05$. Tune the integrator so that there is no steady state error.

Homework D.16

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.15.

- (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.12.
- (b) Add the state space matrices A , B , C , D derived in Homework D.7 to your param file.
- (e) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}) = n$.
- (c) Assuming that the parameters known to the controller are the true parameters, using the Matlab `place` command, 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^d 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.12. 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.17

- (a) Modify the state feedback solution developed in Homework D.16 to add an integrator with anti-windup to the feedback loop for z .
- (b) Change the parameters known to the controller by $\pm 5\%$.
- (c) Add a constant input disturbance of 0.25 Newtons to the input of the plant and verify that when the integrator gain is set to zero, there is a steady state tracking error.
- (d) Tune the integrator to get zero steady state tracking performance.

Homework D.18

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.17.

- (a) Modify the simulink diagram from HW D.17 so that the output of the s-function defining the dynamics is both y and x as shown in Figure ???. In addition, modify the Simulink diagram so that the controller outputs both u and \hat{x} , as also shown in Figure ??.
- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, within an input disturbance, and therefore remove the integrator. Modify your parameter file so that the parameters known to the controller are the actual plant parameters.
- (c) Verify that the state space system is observable by checking that $\text{rank}(\mathcal{O}) = n$.
- (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.

Homework D.19 (a) Modify the parameter file from HW D.18 to use inexact parameters, as in HW D.15 and HW D.17. Explicitly add an additional input disturbance of 0.25 Newtons. 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{z} 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.

Homework D.20

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.21

For the mass spring, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, (2) the plant under PD control, and (3) the plant under PID control, using the control gains calculated in Homework 11.

- (a) What is the tracking error to a unit step under PD control?
- (b) What is the tracking error to a unit ramp under PID control?
- (c) 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 PD 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 z ?

Homework D.22

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.15.

Homework D.23

Design Problem

E. Ball on Beam

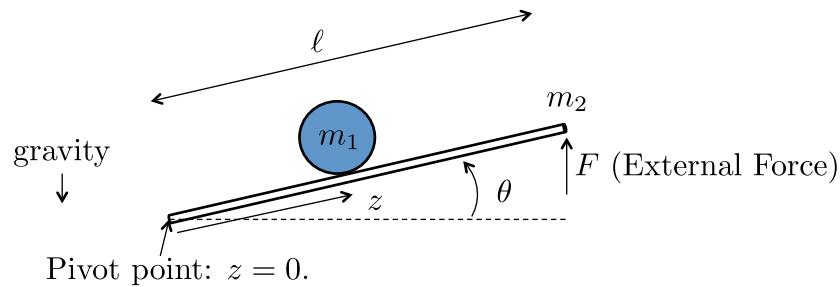


Figure .1: Ball on Beam Problem

Figure .1 shows the ball and beam 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 .1.

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 E.1

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.2

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.
- (d) Derive the equations of motion using the Euler-Lagrange equations.

Homework E.4

Modify the simulink model created in homework E.2 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 E.2.

Homework E.5

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

Homework E.6

- (a) Using the Laplace transform, convert from time domain to the s-domain.
- (b) Find the transfer functions that define the cascade connected from force F to the beam angle $\tilde{\theta}$ to the ball position \tilde{z} .
- (c) Find explicit expressions for the disturbances that have been ignored.
- (d) Draw the associated block diagram.

Homework E.7

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

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

Homework E.8

No assignment.

Homework E.9

No assignment.

Homework E.10

No assignment.

Homework E.11

- (a) For ball and beam, 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 $z_e = \frac{1}{s}\ell$. The input to the outer loop controller is the desired ball position z^d and the output of the controller is the desired beam angle $\tilde{\theta}^d$. The input to the inner loop controller is the desired beam angle $\tilde{\theta}^d$ 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 $t_{r_\theta} = 1$ 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 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 .2.

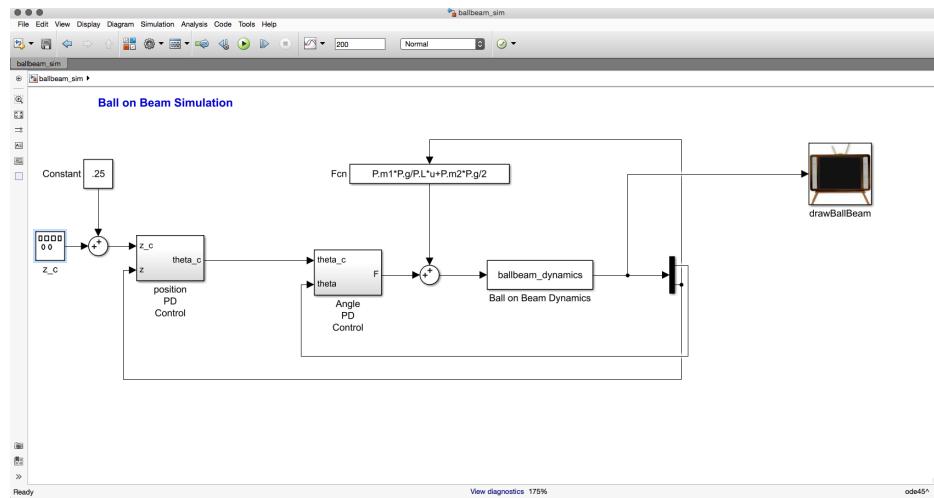


Figure .2: Block diagram for homework E.11.

Homework E.12

Suppose that the size of the input force on the beam is limited to $F_{\max} = 15$ N. The objective of this problem is to revise the successive loop closure design from Homework E.11 to maximize performance.

- (a) Considering the equilibrium force F_e , derive a bound F_{\max} on the linearized force \tilde{F} such that $|\tilde{F}| \leq F_{\max}$ when the maximum possible equilibrium force is used.
- (b) Select the proportional gain $k_{p\theta}$ so that \tilde{F} just saturates when a step of size A_{th} is placed on $\tilde{\theta}_d$.
- (c) Find the proportional gain $k_{D\theta}$ so that the damping ratio of the outer loop is $\zeta_\theta = 0.707$.
- (d) Find the DC gain $k_{DC\theta}$ of the inner loop.
- (e) Replacing the inner loop with its DC gain, find the proportional gain k_{Pz} of the outer loop so that $\tilde{\theta}^d$ just saturates at A_{th} when a step of size $A_z = 0.25$ meters is placed on \tilde{z}^d .
- (f) Find the derivative gain k_{Dz} so that the damping ratio of the outer loop is $\zeta_z = 0.707$.

- (g) Modify the Simulink diagram developed in HW E.11 to include a saturation block on the force F , and implement the successive loop closure design using the revised gains.

Homework E.13

- (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 $\dot{\theta}^d$ 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.14

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. Implement the controller in Simulink, change the value of m_1 to be 95% of its true value and compare the response with and without the integrator.

Homework E.15

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Change the parameter file `param.m` to specify actual parameters and the parameters known to the controller. Change the values for m_1 , m_2 , and ℓ known to the controller by 5% to simulate imperfect knowledge of the plant.
- (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.14 using an m-function called ballbeam_ctrl.m. 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 desired position z^d .

- (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.16

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.15.

- (a) Using the values for ω_{n_z} , ζ_z , ω_{n_θ} , and ζ_θ selected in Homework E.12, find the desired closed loop poles.
- (b) Add the state space matrices A , B , C , D derived in Homework E.7 to your param file.
- (c) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}) = n$.
- (d) Using the Matlab place command, 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^c to z is equal to one.
- (e) Assuming that the parameters known to the controller are the true parameters, 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.17

- (a) Modify the state feedback solution developed in Homework E.16 to add an integrator with anti-windup to the feedback loop for z .
- (b) Change the parameters known to the controller by $\pm 5\%$ and verify that there is a steady state tracking error when the integrator gain is zero.

- (c) Tune the integrator to get zero steady state tracking performance.

Homework E.18

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.17.

- (a) Modify the simulink diagram from HW E.17 so that the output of the s-function defining the dynamics is both y and x as shown in Figure ???. In addition, modify the Simulink diagram so that the controller outputs both u and \hat{x} , as also shown in Figure ??.
- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, within an input disturbance, and therefore remove the integrator. Modify your parameter file so that the parameters known to the controller are the actual plant parameters.
- (c) Verify that the state space system is observable by checking that $\text{rank}(\mathcal{O}) = n$.
- (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.

Homework E.19 (a) Modify the parameter file from HW E.18 to use inexact parameters, as in HW E.15 and HW E.17. Explicitly add an additional input disturbance of 0.5 Newtons. 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{z} 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.

Homework E.20

- (a) Draw by hand the Bode plot of the inner loop transfer function from force 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 $z(t)$ for the ball beam system. Use the Matlab `bode` command and compare your results.

Homework E.21

For the inner loop of the ballbeam 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.15.

- (a) To what percent error can the closed loop system under PD control track the desired input if all of the frequency content of $\theta^d(t)$ is below $\omega_r = 1.0$ radians per second?
- (b) If the frequency content of the input disturbance is all contained below $\omega_{din} = 0.8$ 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, (2) the plant under PD control, and (3) the plant under PID control, using the control gains calculated in Homework E.15.

- (d) If the frequency content of an output disturbance is contained below $\omega_{dout} = 0.1$ radian/sec, what percentage of the output disturbance will be contained in the output under both PD and PID control?
- (e) If the reference signal is $y^d(t) = 2 \sin(0.2t)$ what is the output error under PD control?

Homework E.22

For this problem, use the gains found in HW E.15.

- (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.23

Design Problem

F. Gantry Crane

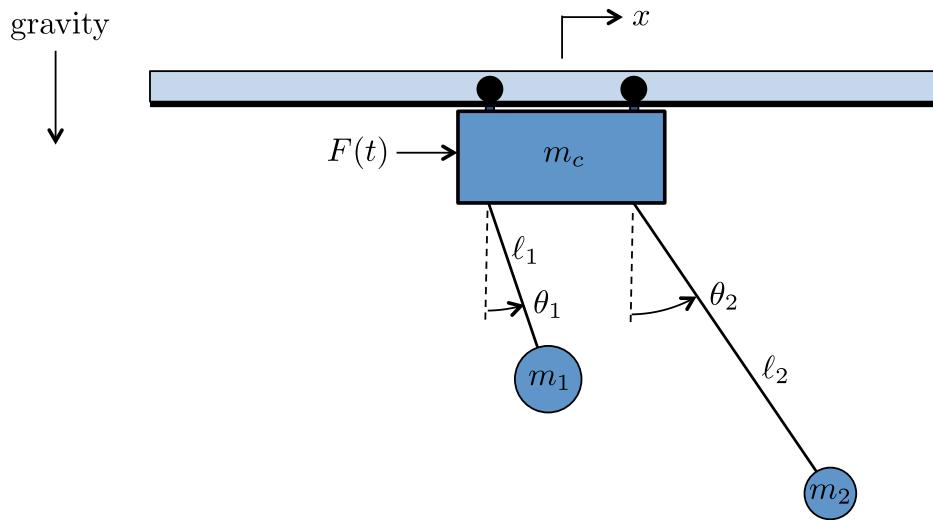


Figure .1: Gantry Crane

Figure .1 shows a gantry crane system with two slung loads. The crane car is suspended from a track and has mass m_c . The rolling friction on the track is proportional to the velocity of the car with friction constant b . The crane carries two loads. Load 1 has mass m_1 and is suspended by a cable of length ℓ_1 . Load 2 has mass m_2 and is suspended by a cable of length ℓ_2 . The crane car is propelled along the track by a force $F(t)$ and its position is given by x . The angles of the loads are given by θ_1 and θ_2 respectively.

Use the following physical parameters: $m_c = 1500$ kg, $m_1 = 500$ kg,

$m_2 = 200 \text{ kg}$, $b = 200 \text{ N-s/m}$, $\ell_1 = 10 \text{ m}$, $\ell_2 = 24 \text{ m}$, $g = 9.81 \text{ m/s}^2$.

Homework F.1 Using the configuration variable x , θ_1 , and θ_2 , write an expression for the kinetic energy of the system.

Homework F.2 Create a simulink animation of the gantry crane system. The inputs should be sliders for x , θ_1 and θ_2 . Turn in a screen capture of the animation.

Homework F.3

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

Homework F.4 Modify the simulink model created in homework F.2 by creating an s-function that implements the equations of motion. The input to the s-function should be a slider for F . The output should go to the animation developed in homework F.2.

Homework F.5

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

Homework F.6

- (a) Using the formula $A^{-1} = \frac{\text{adj}(A)}{\det(A)}$ where $\text{adj}(A)$ is the adjugate of A which equals transpose of the co-factors of A , and $\det(A)$ is the determinant of A , show that

$$M^{-1} = \begin{pmatrix} \frac{1}{m_c} & -\frac{1}{m_c \ell_1} & -\frac{1}{m_c \ell_2} \\ -\frac{1}{m_c \ell_1} & \frac{m_c + m_1}{m_c m_1 \ell_1^2} & \frac{1}{m_c \ell_1 \ell_2} \\ -\frac{1}{m_c \ell_2} & \frac{1}{m_c \ell_1 \ell_2} & \frac{m_c + m_2}{m_c m_2 \ell_2^2} \end{pmatrix}$$

- (b) Using M^{-1} solve for $(\ddot{x}, \ddot{\theta}_1, \ddot{\theta}_2)^\top$.
- (c) Using the Laplace transform, convert the equations of motion from time domain to the s-domain.

- (d) Solve each equation individually for $X(s)$, $\Theta_1(s)$, and $\Theta_2(s)$ ignoring the coupling between equations.
- (e) For frequency domain designs, we will treat the motion of the first mass m_1 , or in other words θ_1 as a disturbance on the system. Identify the transfer function from force $F(s)$ and the second angle $\Theta_2(s)$ and the transfer function from $\Theta_2(s)$ to the cart position $X(s)$. Label the other terms as disturbances.
- (f) Draw a block diagram of the open-loop system showing the transfer function cascade with F as the input and x as the output of the cascade, with disturbances entering at the appropriate locations.

Homework F.7 Defining the states as $x = (x, \theta_1, \theta_2, \dot{x}, \dot{\theta}_1, \dot{\theta}_2)^\top$, the input as $u = F$, and the measured output as $y = (x, \theta_1, \theta_2)^\top$, find the linear state space equations in the form

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

Homework F.8 Suppose that the size of the input force on the cart is limited to $F_{\max} = 4500$ N. Design separate PD controllers for the inner loop and the outer loop.

- (a) Using the transfer functions derived in problem F.6, and using a PD controller for the inner loop to regulate θ_2 to θ_2^d , and a PD controller for the outer loop to regulate z to z^d , draw the block diagram of the complete closed loop system.
- (b) Derive the actual transfer function of the inner loop from θ_2^d to θ_2 .
- (c) Select the proportional gain $k_{p\theta_2}$ so that F just saturates when a step of size A_{th} is placed on θ_2^d .
- (d) If the desired transfer function is given by

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

find the natural frequency $\omega_{n_{\theta_2}}$ and the derivative gain $k_{d\theta_2}$ so that the actual poles of the closed loop transfer function equals

the poles of the desired transfer function, where ζ_{θ_2} is a design parameter. Note that the DC gain of the actual transfer function is not equal to one.

- (e) Compute the DC gain k_{DC} .
- (f) Assuming that the inner loop is much faster than the outer loop, replace the inner loop with its DC gain k_{DC} , and draw the block diagram for the outer loop. Derive the actual transfer function from z^d to z . For this part of the problem you will need to use the transfer function $T_{z/\theta_2} = \frac{-\frac{m_2 g}{m_c}}{s(s+\frac{b}{m_c})}$ instead of $T_{z/\theta_2} = \frac{\frac{m_2 g}{m_c}}{s(s+\frac{b}{m_c})}$. The negative sign makes sense physically since achieving a positive motion in z requires a negative motion in θ_2 , but at this time I am not completely sure why the negative sign does not show up in the model.
- (g) Select the proportional gain k_{p_z} so that θ_2^d just saturates at A_{th} when a step of size $A_z = 0.25$ meters is placed on z^d .
- (h) If the desired transfer function is given by

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

find the natural frequency ω_{n_z} and the derivative gain k_{d_z} so that the actual transfer function equals the desired transfer function, where ζ_z is a design parameter.

- (i) What are the closed loop poles of the outer loop?

Homework F.9 Implement the PD controllers for the angle and position loops designed in Homework F.8 on the gantry crane system in Simulink. Note that the derivative of the positions ($\dot{\theta}_2$ and \dot{x}) are available for feedback as outputs of your dynamics block.

Homework F.10

- (a) With PD control for the inner loop, what is the system type of the inner loop? 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) Recall from Problem 6, that in deriving the transfer functions that we obtained the expression (setting $\Theta_1(s) = 0$)

$$\Theta_2(s) = \left(\frac{-a_1}{s^2 + a_2} \right) F(s) + \left(\frac{a_4 s}{s^2 + a_2} \right) Z(s),$$

where we considered

$$d(s) = \left(\frac{a_4 s}{s^2 + a_2} \right) Z(s)$$

as an output disturbance on the inner loop. What is the response of the inner loop to a step on $z(t)$ with and without an integrator in the controller?

- (c) With PD control for the outer loop, what is the system type of the outer loop? 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?
- (d) Find the response of the outer loop to a unit step input disturbance, both when the outer loop controller does not have an integrator and when it does.
- (e) With the integrator, put the characteristic equation of the outer loop in Evans's form and use the Matlab SISO tool to plot the root locus versus the integrator gain k_i .

Homework F.11 The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Change the parameter file param.m to specify actual parameters and the parameters known to the controller. Change the values for m_1 , m_2 , m_c , ℓ_1 , ℓ_2 , and b known to the controller by $\pm 5\%$ to simulate imperfect knowledge of the plant.
- (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 angles θ_1 , and θ_2 , as well as the desired position z^d .

- (c) Implement the nested PID loops designed in Problems F.8 and F.10 using an m-function called `gantry_ctrl.m`. Use the dirty derivative gain of $\tau = 0.05$.

Homework F.12 The objective of this problem is to implement state feedback controller using the full state. Start with the simulation files developed in Homework Problem 11.

- (a) Modify `gantry_dynamics.m` so that the outputs are $(z, \theta_1, \theta_2, x)^\top$, where $x = (z, \theta_1, \theta_2, \dot{z}, \dot{\theta}_1, \dot{\theta}_2)^\top$. In this problem we will use the full state x , but in the next problem we will again remove x as an output of the plant and reconstruct the state using an observer.
- (b) Rearrange the block diagram so that there is a single controller with the desired position z^d and the state x is an input to the controller.
- (c) There will be six closed loop poles, and in this problem we can select all six poles, or in other words, we will be controlling both angles. Using the values for ω_{n_z} , ζ_z , $\omega_{n_{\theta_2}}$, and ζ_{θ_2} derived in Problem 11 as starting points, find desired closed loop poles for all six states.
- (d) Add the state space matrices A , B , C , D derived in Homework 7 to your param file. For comparison, for perfect parameters, I get

$$A = \begin{pmatrix} 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 \\ 0 & 3.2700 & 1.3080 & -0.1333 & 0 & 0 \\ 0 & -13.0800 & -1.3080 & 0.1333 & 0 & 0 \\ 0 & -0.1363 & -0.4632 & 0.0056 & 0 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0.6667e-03 \\ -0.6667e-03 \\ -0.0278e-03 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

- (e) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}) = n$.
- (f) Using the Matlab place command, find the feedback gain K such that the eigenvalues of $(A - BK)$ are equal to desired closed loop poles. Use the control signal $u = -Kx - k_r z^d$ where z^d is the desired position of the cart.
- (g) Implement the state feedback scheme in Simulink and tune the closed loop poles to get good response.

Homework F.13

- (a) Modify the state feedback solution developed in Homework 15 to add an integrator with anti-windup to the position feedback.
- (b) Change the parameters known to the controller by $\pm 5\%$.
- (c) Add a constant input disturbance of 1000 Newtons to the input of the plant and verify that when the integrator gain is set to zero, there is a steady state tracking error.
- (c) Tune the integrator to get zero steady state tracking performance.

Hints:

- The integrator gain will be negative.
- To get fast integrator action, try turning off the integrator during transition, i.e., when $|y|$ is greater than some limit.

Homework F.14 The objective of this problem is to replace the state in Homework 16 with the estimated state \hat{x} produced by an observer.

- (a) Modify the simulink diagram from Homework 16 so that y is also an input to the controller, as well as x , and so that \hat{x} is also an output of the controller, where $y = (z, \theta_1, \theta_2)^\top$. Add a scope to plot the estimation error $x - \hat{x}$.
- (b) Verify that the state space system is observable by checking that $\text{rank}(\mathcal{O}) = n$.
- (c) In the control block, add an observer to estimate the state \hat{x} . Do not yet use \hat{x} in the state feedback equations. Run the controller

using the actual state x and tune the observer so that the estimation error is small. Make sure the $\text{eig}(A - BK - LC)$ are in the left half plane.

- (d) Replace x in the controller with the estimated state \hat{x} and tune the controller and observer if necessary to obtain good performance.
- (e) Modify the simulink diagram to add an input disturbance of 1000 Newtons. Add a state to the observer to estimate the input disturbance. Replace the integrator with an estimate of the disturbance and compare the performance.

Hints:

- *The performance of the system is highly dependent on the pole locations of the observer. I obtained good response by using the LQR command*

P.Lo = lqr(P.Ao', P.Co', diag([1,1,1,1,1,1,rho]), diag([1,1,1]))', where rho encodes the weight on the estimation error of the disturbance and needs to be a large number.

Homework F.15

- (a) Draw by hand the Bode plot of the inner loop transfer function from force F to angle $\tilde{\theta}_2$ 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 $z(t)$ for the ball beam system. Use the Matlab `bode` command and compare your results.

Homework F.16 For the inner loop of the gantry system, use the Matlab `bode` command to create a graph that simultaneously displays the Bode plots for (1) the plant, (2) the plant under PD control, and (3) the plant under PID control using the control gains calculated in Homework 11.

- (a) What percent of the input disturbance shows up in $\theta_2(t)$ if the frequency content of the input disturbance is below 0.3 radians per second?
- (b) What is the tracking error for a unit step in the command to $\theta_2(t)$ for PD control? How does this relate to the DC gain of the system?

- (c) If all of the frequency content of the noise $n(t)$ is greater than $\omega = 100$ radians per second, what percentage of the noise shows up in the output signal θ_2 ?

For the outer 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 11.

- (d) If the frequency content of an output disturbance is contained below 0.01 radian/sec, what percentage of the output disturbance will be contained in the output under PD control?
- (e) If the reference signal $z^d(t)$ is a ramp with slope 0.1 m/s, what is the steady state tracking error under PD control?

Homework F.17 For this homework assignment we will use loopshaping to improve the PD controllers developed in Homework V.9. Download the simulink files on the web that are associated with this homework. We will focus on the outer loop. When you open the Simulink file, notice the presence of the LTI system with parameter C.

- (a) At the Matlab prompt, type `>> C = 1` to set the additional compensator originally to one. Run the simulation to establish baseline performance.
- (b) Note in `param.m` the line `L=series(P_out,C_out_pd)`, where $L(s)$ is the loop gain for the out loop. Load the `sisotool` and enter L for the system plant under the tab System Data.
- (c) The objective will be to (1) improve disturbance rejection and tracking by a factor of 10 for input signals and disturbance below 0.001 radians/sec, (2) improve noise attention by a factor of 10 for frequencies above 20 radians/sec. In `sisotool` add constraints that reflect these objectives.
- (d) Add a phase lag filter to satisfy the disturbance rejection and tracking objective, and add one to two poles to meet the noise rejection objective. The phase margin in this case is interpreted different. The phase above 0 degrees corresponds to the normal phase margin. This should remain around 60 degrees. To understand why, you will need to understand the Nyquist criteria.

(e) Export $C(s)$, and run the combined controller in Simulink.

Turn the following graphics: (1) The step response with PID only. (3) The loopshape in `sisotool` showing constraints and the loop gain that satisfies the constraints. (4) The final step response.

Design Problem

G. Planar VTOL

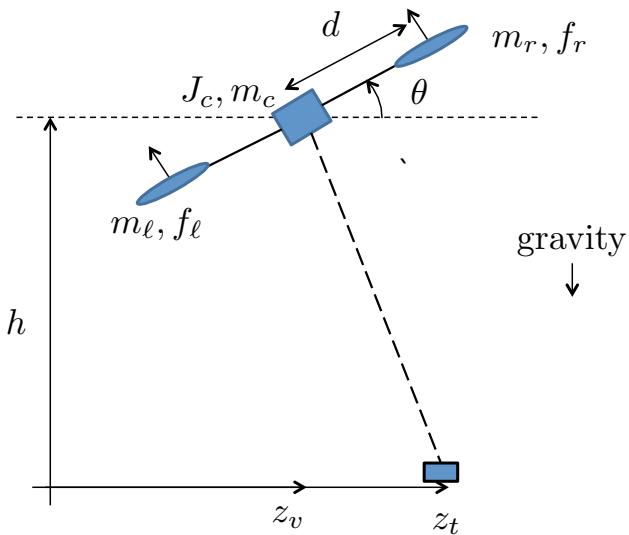


Figure .1: Planar vertical take-off and landing (VTOL)

There is currently significant interest in quadrotors and other multi-rotor systems due to their low cost and the ability to carry a small video camera. While most commercially available quadrotors are radio controlled (RC), there are many potential applications that will be enabled through autonomous operations. The first step in developing autonomous multi-rotor systems is the flight control algorithms. In this design study we will explore control design for a simplified 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 .1. 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 G.1

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

Homework G.2

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 G.3

- (a) Find the potential energy for the system.
- (b) Define the generalized coordinates.
- (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 G.4

Modify the simulink model created in homework G.2 by creating an s-function that implements the equations of motion. The input to the s-function should be a slider for the forces f_r and f_ℓ . The output

should go to the animation developed in homework G.2. The target is still maneuvered using a slider bar.

Homework G.5

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.

Homework G.6

- (a) Using the Laplace transform, convert the linearized equations of motion from time domain to the s-domain.
- (b) Find the transfer functions from $\tilde{F}(s)$ to $H(s)$, from $\tau(s)$ to $\Theta(s)$, and from $\Theta(s)$ to $Z(s)$. Write each transfer function in monic form, i.e., where the leading coefficient in the denominator is one.
- (c) Note that the system naturally divides into so-called longitudinal dynamics (up-down motion) with \tilde{F} as the input and H as the output, and lateral dynamics (side-to-side motion) with τ as the input and Z as the output, with Θ as an intermediate value. Draw a block diagram of the open-loop system longitudinal system, and the open loop lateral system.

Homework G.7

- (a) Defining the longitudinal states as $x_{lon} = (h, \dot{h})^\top$ and the longitudinal input as $u_{lon} = \tilde{F}$ and the longitudinal output as $y_{lon} = h$, find the linear state space equations in the form

$$\begin{aligned}\dot{x}_{lon} &= Ax_{lon} + Bu_{lon} \\ y_{lon} &= Cx_{lon} + Du_{lon}.\end{aligned}$$

- (b) Defining the lateral states as $x_{lat} = (z, \theta, \dot{z}, \dot{\theta})^\top$ and the lateral input as $u_{lat} = \tau$ and the lateral output as $y_{lat} = (z, \theta)^\top$, find the linear state space equations in the form

$$\begin{aligned}\dot{x}_{lat} &= Ax_{lat} + Bu_{lat} \\ y_{lat} &= Cx_{lat} + Du_{lat}.\end{aligned}$$

Homework G.8

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 F to altitude h found in problem D.6, find the open loop poles of the system.
- (b) Using PD control architecture shown in Figure ??, find the closed loop transfer function from h^c to 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 $p_2 = -0.3$.

Homework G.9

Using the gains found in Problem G.8, implement the PD control for altitude dynamics in Simulink and plot the step response from 10 m to 15 m. Turn in an image of the simulink block diagram, as well as the step response of the system.

Hint: Change the s-function that defines the dynamics so that the output is the configuration variables z, h, θ . The Simulink block should look like that shown in Figure .2.

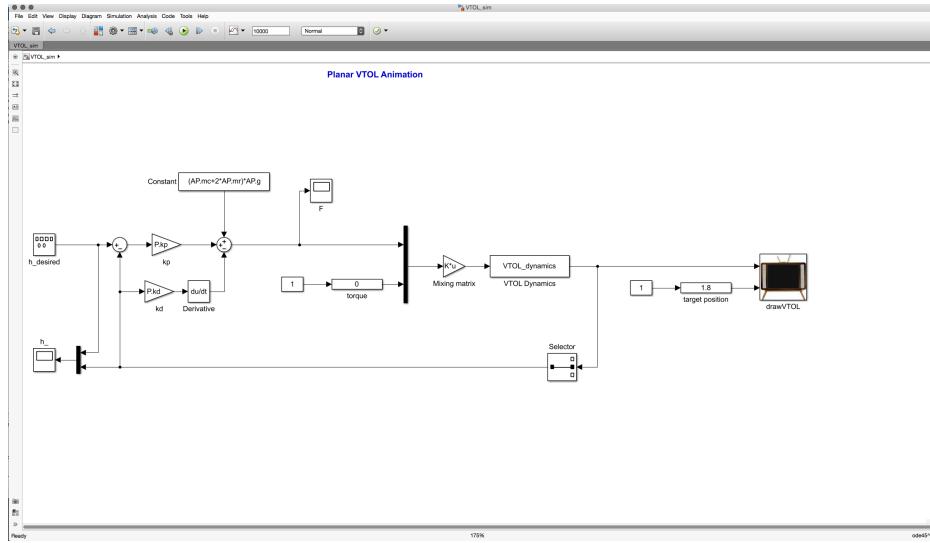


Figure .2: Simulink file for Homework G.9

Add a mixing block that converts force F and torque τ into right and left forces f_r and f_ℓ . For this problem set the torque to $\tau = 0$. Recall that the force in the transfer function model is the linearized torque $\tilde{F} = F - F_e$. Therefore, the force applied to the VTOL is $F = F_e + \tilde{F}$. The equilibrium force must be added to the force computed by the PID controller as shown in Figure .2.

Homework G.10

For this problem we will only consider the longitudinal (up-down) dynamics of the VTOL system.

- (a) Suppose that the design requirements for the altitude hold on the VTOL system are
- Rise time: $t_r < 8$ seconds,
 - Maximum percent overshoot: $M_p < 15\%$,
 - Settling time: $t_s < 20$ seconds, within 1%.

Find the region in the s-plane where the poles would result in the design requirements being satisfied.

- (b) Select pole locations that satisfy the design requirements and also minimize the distance to the origin in the complex plane. Min-

imizing the distance to the origin will result in minimal control action, i.e., $u(t)$ will be small in some sense.

- (c) Using the PD architecture shown in Figure ??, find the proportional gain k_p and the derivative gain k_d that place the poles at the location selected in part (b).
- (d) Implement the PD controller in Simulink when $h^d(t)$ is a square wave of magnitude 2.5 meters and frequency 0.02 Hz. Verify that the step response satisfies the design specifications.

Homework G.11

- (a) 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^d and the output of the controller is the desired pitch angle θ^d . The input to the inner loop controller is the desired pitch angle θ^d and the output is the torque τ on the VTOL center.
- (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 $t_{r_\theta} = 0.8$ 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 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. Run the lateral controller in conjunction with the controller designed in problem G.10. The block diagram for the system should look something like Figure .3 and .4.

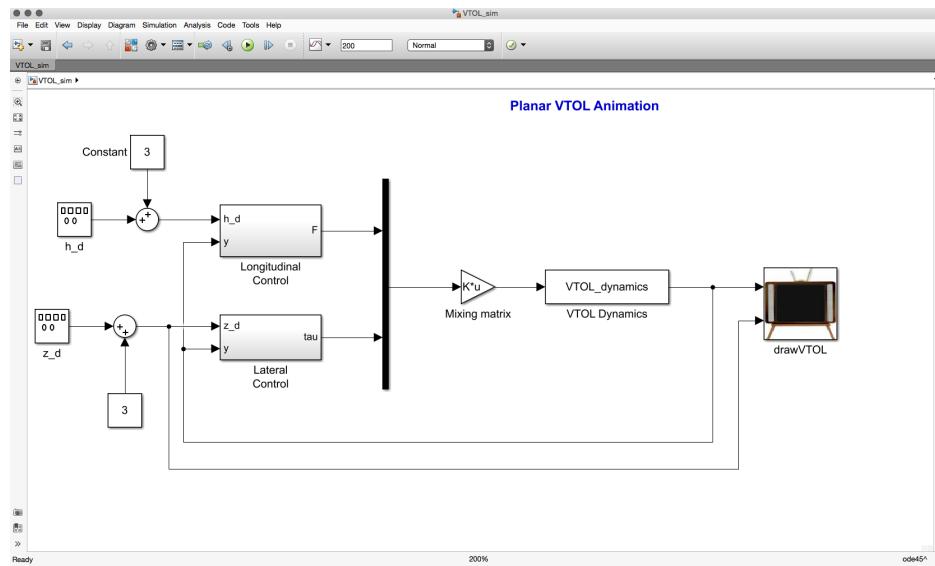


Figure .3: Block diagram for homework G.11.

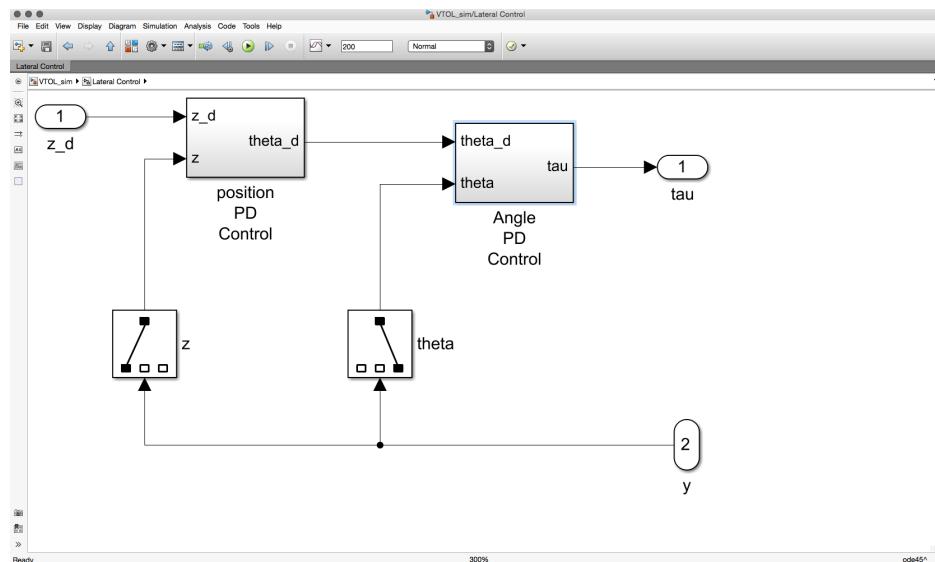


Figure .4: Block diagram for lateral controller in homework G.11.

Homework G.12

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. The objective of this problem is to revise the PD designs from HW G.10 and HW G.11.

- (a) Given the constraints on f_ℓ and f_r , and the equilibrium force required to maintain the altitude, find constraints on \tilde{F} and τ such that $\tilde{F} \leq \tilde{F}_{\max}$ and $|\tau| \leq \tau_{\max}$.
- (b) For the altitude loop, select the proportional gain k_{P_h} so that \tilde{F} just saturates when a step of size $A_h = 10$ meters is placed on h^d . Find the derivative gain so that the damping ratio of the altitude loop is $\zeta = 0.707$.
- (c) For the inner loop of the lateral autopilot, find the proportional gain k_{P_θ} so that the torque just saturates for a step of $\theta^d = A_\theta$. Find the derivative gain k_{D_θ} so that 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 with its DC gain, find the proportional gain k_{P_z} of the outer loop so that θ^d just saturates at A_{th} when a step of size $A_z = 10$ meters is placed on z^d . Find the derivative gain k_{D_z} so that the damping ratio of the outer loop is $\zeta_z = 0.707$.
- (f) Modify the Simulink diagram developed in HW G.10 and HW G.11 to include saturation blocks on the right and left forces f_ℓ and f_r , and implement the successive loop closure design using the revised gains, tuning the parameters A_θ and ζ_* as needed to obtain good performance.

Homework G.13

- (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 G.14

- (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 verses 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. Implement the controller in Simulink, change the value of m_c to be 95% of its true value and compare the response with and without the integrator.
- (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 verses 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. Implement the controller in Simulink, change the value of J_c to be 95% of its true value and compare the response with and without the integrator.

Homework G.15

The objective of this problem is to implement the PID controller using Matlab code using only measured outputs of the system.

- (a) Change the parameter file `param.m` to specify actual parameters and the parameters known to the controller. Change the values for m_c , J_c , d , and μ known to the controller by $\pm 5\%$ to simulate imperfect knowledge of the plant.
- (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 E.14 using an m-function called `VTOL_ctrl.m`. 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 desired position z^d , and the desired altitude h^d .

Homework G.16

The objective of this problem is to implement a state feedback controller for the VTOL system. Start with the simulation files developed in Homework G.15.

- (a) Using the values for ω_{n_h} , ζ_h , ω_{n_z} , ζ_z , ω_{n_θ} , and ζ_θ selected in Homework G.12, find the desired closed loop poles.
- (b) Add the state space matrices A , B , C , D derived in Homework G.7 to your param file.
- (c) Verify that the state space system is controllable by checking that $\text{rank}(\mathcal{C}) = n$.
- (d) Assuming that the parameters known to the controller are the true parameters, and using the Matlab `place` command, 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^d to h is equal to one, and the DC-gain from z^d 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 G.17

- (a) Modify the state feedback solution developed in Homework G.16 to add an integrator with anti-windup to the altitude feedback loop.
- (b) Change the parameters known to the controller by $\pm 5\%$.
- (c) Tune the integrator on the altitude loop to get zero steady state tracking performance.
- (d) Modify the state feedback solution developed in Homework G.16 to add an integrator with anti-windup on the lateral position.
- (e) Add a constant input disturbance of 0.1 Newtons to the input of the plant simulating wind and verify that when the integrator gain

is set to zero, there is a steady state tracking error. 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) / (AP.mc+2*AP.mr);
```

- (f) Tune the integrator on the position loop to get zero steady state tracking performance.

Homework G.18

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 G.17.

- (a) Modify the simulink diagram from HW G.17 so that the output of the s-function defining the dynamics is both y and x as shown in Figure ???. In addition, modify the Simulink diagram so that the controller outputs both u and \hat{x} , as also shown in Figure ???.
- (b) For the sake of understanding the function of the observer, for this problem we will use exact parameters, within an input disturbance, and therefore remove the integrator. Modify your parameter file so that the parameters known to the controller are the actual plant parameters.
- (c) For both the longitudinal and lateral controllers, verify that the state space system is observable by checking that $\text{rank}(\mathcal{O}) = n$.
- (d) For both the longitudinal and lateral controllers, 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.

Homework G.19 (a) Modify the parameter file from HW G.18 to use inexact parameters, as in HW G.15 and HW G.17. Explicitly add an additional input disturbance of ? Newtons to the longitudinal dynamics (simulating additional mass on the VTOL), and an input disturbance of ? Newtons to the lateral dynamics (simulating constant wind). 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{h} and \dot{z} are 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 both controllers, 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.

Homework G.20

- (a) Draw by hand the Bode plot of the altitude transfer function from force F to altitude 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 G.21

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, (2) the plant under PD control, and (3) the plant under PID control, using the control gains calculated in Homework G.15.

- (a) For both PD and PID control, 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 G.15.

- (c) If the frequency content of the input disturbance is all contained below $\omega_{d_{in}} = 2$ 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 G.15.

- (e) To what percent error can the closed loop system track the desired input if all of the frequency content of $z^d(t)$ is below $\omega_r = 0.1$ radians per second?
- (f) If the frequency content of an output disturbance is contained below $\omega_{d_{out}} = 1$ radian/sec, what percentage of the output disturbance will be contained in the output?

Homework G.22

For this problem, use the gains found in HW G.15.

- (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 G.23

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. Therefore, in this section we describe how to create animations in Matlab/Simulink.

1.1 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))

```

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 re-drawing 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 three detailed examples. In Section 1.2 we illustrate a 2-D animation of an inverted pendulum using the fill command. In Section 1.3 we illustrate a 3-D animation of a spacecraft using lines to produce a stick figure. In Section 1.4 we modify the spacecraft animation to use the vertices-faces data construction in Matlab.

1.2 Animation Example: Inverted Pendulum

Consider the image of the inverted pendulum shown in Figure A.1, where the configuration is completely specified by the position of the cart y , 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 the position of points that define the animation. For example, for the inverted pendulum in Figure A.1, the four corners of the base are

$$(y + w/2, g), (y + w/2, g + h), (y - w/2, g + h), \text{ and } (y - w/2, g),$$

and the two ends of the rod are given by

$$(y, g + h) \text{ and } (y + L \sin \theta, g + h + L \cos \theta).$$

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:

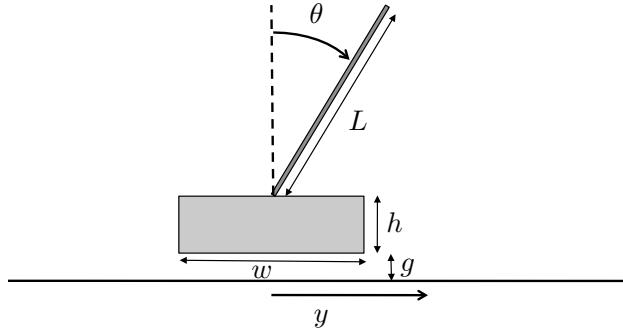


Figure A.1: Drawing for inverted pendulum. The first step in developing an animation is to draw a figure of the object to be animated and identify all of the physical parameters.

```

1   function handle = drawBase(y, width, height, gap, handle)
2       X = [y-width/2, y+width/2, y+width/2, y-width/2];
3       Y = [gap, gap, gap+height, gap+height];
4       if isempty(handle),
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(y, theta, L, gap, height, handle)
2       X = [y, y+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          y           = 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(y, width, height, gap, []);
26             rod_handle = drawRod(y, 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(y, width, height, gap, base_handle);
31             drawRod(y, theta, L, gap, height, rod_handle);
32         end

```

The routine `drawPendulum` is called from the Simulink file shown in Figure A.2, where there are three inputs: the position y , the angle θ , and the time t . Lines 3-5 rename the inputs to y , 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 18-31 is used to produce the animation. Lines 19–25 are called once at the beginning of the simulation, and draw the initial animation. Line 19 brings the figure 1 window to the front and clears it. Lines 20 and 21 draw the ground along which the pendulum will move. Line 23 calls

the `drawBase` routine with an empty handle as input, and returns the handle `base_handle` to the base. Line 24 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 29 and 30, the `drawBase` and `drawRod` routines are called with the figure handles as inputs.

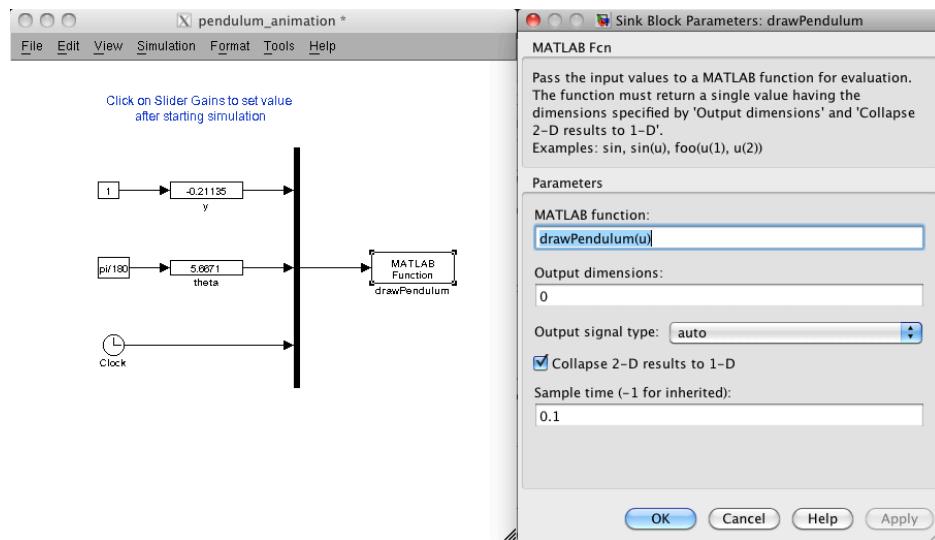


Figure A.2: Simulink file for debugging the pendulum simulation. There are three inputs to the Matlab m-file `drawPendulum`: the position y , the angle θ , and the time t . Slider gains for y and θ are used to verify the animation.

1.3 Animation Example: Spacecraft using Lines

The previous section described a simple 2-D animation. In this section we discuss a 3-D animation of a spacecraft with six degrees of freedom. Figure A.3 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.

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.3 and specify how coordinates

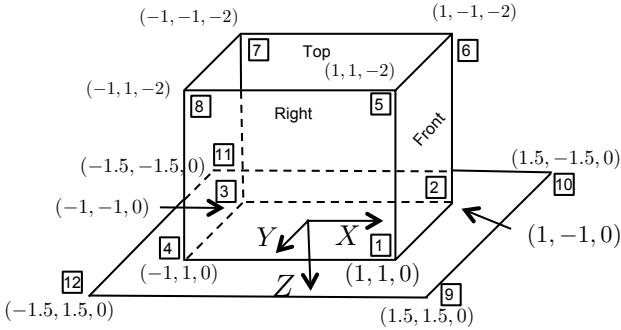


Figure A.3: Drawing used to create spacecraft animation. Standard aeronautics body axes are used, where the x -axis points out the front of the spacecraft, the y -axis points to the right, and the z -axis point out the bottom of the body.

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.3, 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
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
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.4.

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.5. The spacecraft points can be defined in the initializa-

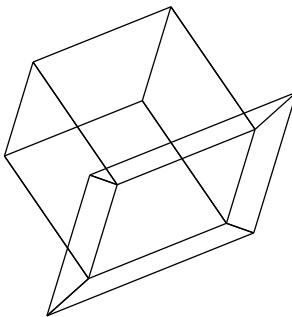


Figure A.4: Rendering of the spacecraft using lines and the `plot3` command.

tion window, as shown in Figure A.5, and passed to the `drawSpacecraft` m-file as a parameter.

1.4 Animation Example: Spacecraft Using Vertices and Faces

The stick-figure drawing shown in Figure A.4 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

```

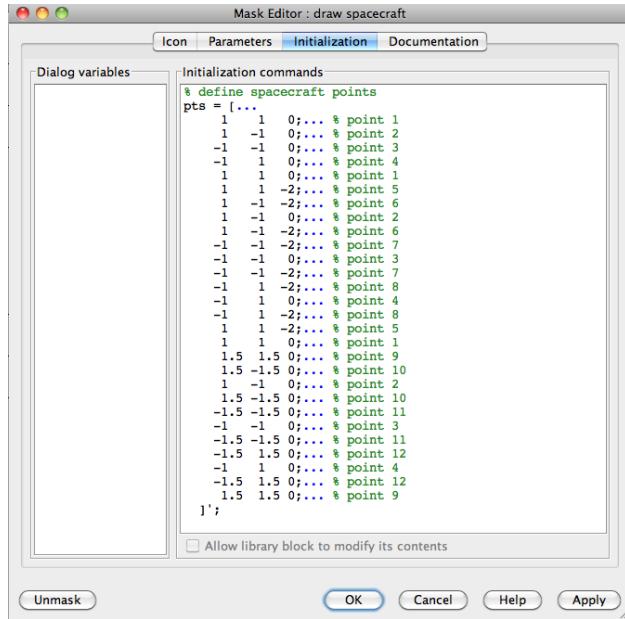


Figure A.5: The mask function in Simulink allows the spacecraft points to be initialized at the beginning of the simulation.

```

14      -1.5 -1.5   0;... % point 11
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.3 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 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.6. Additional examples using the vertex-face format can be found at the book website.

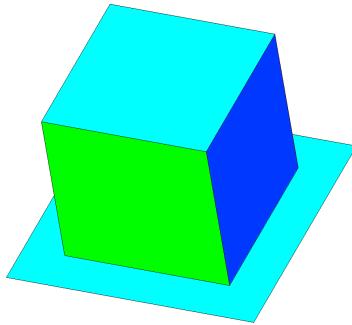
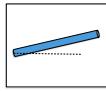


Figure A.6: Rendering of the spacecraft using vertices and faces.

1.5 Design Study A: Single Link Robot Arm



Homework Problem A.2

Create a simulink animation of the single link robot arm. The input should be a slider for θ .

Solution

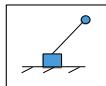
The m-file that implements the animation for the single link arm is listed below.

```
1 function drawArm(u)
2
3     % process inputs to function
4     theta    = u(1);
5     t        = u(2);
6
7     % drawing parameters
8     L = 1;
9     w = 0.3;
10
11    % define persistent variables
12    persistent arm_handle
13
14    % first time function is called, initialize plot
15    % and persistent vars
```

```
16 if t==0,
17     figure(1), clf
18     plot([0,L],[0,0],'k—'); % plot track
19     hold on
20     arm_handle = drawArm_(theta, L, w, []);
21     axis([-2*L, 2*L, -2*L, 2*L]);
22
23
24     % at every other time step, redraw base and rod
25 else
26     drawArm_(theta, L, w, arm_handle);
27 end
28 end
29
30
31 %
32 %=====%
33 % drawArm_
34 % draw the arm
35 % return handle if 3rd argument is empty, otherwise use
36 % 3rd arg as handle
37 %=====
38 %
39 function handle = drawArm_(theta, L, w, handle)
40
41     pts = [
42         0, -w/2; ...
43         L, -w/2; ...
44         L, w/2; ...
45         0, w/2; ...
46     ]';
47     pts = [cos(theta), -sin(theta); sin(theta), cos(theta)]*pts;
48     X = pts(1,:);
49     Y = pts(2,:);
50
51 if isempty(handle),
52     handle = fill(X,Y,'b');
53 else
54     set(handle, 'XData',X, 'YData',Y);
55     drawnow
56 end
57 end
```

For a complete solution to this problem, see the wiki associated with this book.

1.6 Design Study B. Pendulum on Cart



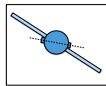
Homework Problem B.2

Create a simulink animation of the pendulum on a cart system. The inputs should be sliders for z and θ .

Solution

See the matlab code listed in the discussion above. For a complete solution to this problem, see the wiki associated with this book.

1.7 Design Study C. Satellite Attitude Control



Homework Problem C.2

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 single link arm is listed below.

```
1 function drawSatellite(u)
2
3     % process inputs to function
4     theta    = u(1);
5     phi      = u(2);
6     t        = u(3);
7
8     % drawing parameters
9     L = 1;
10    w = .3;
11
12    % define persistent variables
13    persistent base_handle
```

```
14 persistent panel_handle
15
16 % first time function is called, initialize plot
17 % and persistent vars
18 if t==0,
19     figure(1), clf
20     track_width=3;
21     plot([-2*L,2*L],[0,0],'k--'); % plot track
22     hold on
23     base_handle = drawBase(theta, w, []);
24     panel_handle = drawPanel(phi, w, L, []);
25     axis([-2*L, 2*L, -2*L, 2*L]);
26
27
28 % at every other time step, redraw base and rod
29 else
30     drawBase(theta, w, base_handle);
31     drawPanel(phi, w, L, panel_handle);
32 end
33 end
34
35
36 %
37 %=====%
38 % drawBase
39 % draw the base of the pendulum
40 % return handle if 3rd argument is empty, otherwise use
41 % 3rd arg as handle
42 %=====
43 %
44 function handle = drawBase(theta, w, handle)
45
46 % define points on base (without rotation)
47 pts = [...
48     w/2, -w/2; ...
49     w/2, -w/6; ...
50     w/2+w/6, -w/6; ...
51     w/2+w/6, w/6; ...
52     w/2, w/6; ...
53     w/2, w/2; ...
54     -w/2, w/2; ...
55     -w/2, w/6; ...
56     -w/2-w/6, w/6; ...
57     -w/2-w/6, -w/6; ...
58     -w/2, -w/6; ...
```

```
59      -w/2, -w/2;...
60      ]';
61 % define rotation matrix
62 R = [cos(theta), sin(theta); -sin(theta), cos(theta)];
63 % rotate points
64 pts = R*pts;
65 % break into X and Y components
66 X = pts(1,:);
67 Y = pts(2,:);

68 if isempty(handle),
69     handle = fill(X,Y,'b');
70 else
71     set(handle,'XData',X,'YData',Y);
72     drawnow
73 end
74 end
75 end

76 %
77 %=====
78 % drawPanel
79 % draw the solar panel
80 % return handle if 3rd argument is empty, otherwise use
81 % 3rd arg as handle
82 %=====
83 %
84 function handle = drawPanel(phi, w, L, handle)

85 % define points on base (without rotation)
86 pts = [...
87     -L, -w/6;...
88     L, -w/6;...
89     L, w/6;...
90     -L, w/6;...
91 ];
92 % define rotation matrix
93 R = [cos(phi), sin(phi); -sin(phi), cos(phi)];
94 % rotate points
95 pts = R*pts;
96 % break into X and Y components
97 X = pts(1,:);
98 Y = pts(2,:);

101 if isempty(handle),
102     handle = fill(X, Y, 'g');
```

```
104     else
105         set(handle, 'XData', X, 'YData', Y);
106         drawnow
107     end
108 end
```

For a complete solution to this problem, see the wiki associated with this book.

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

$$\dot{x}_c = f(t, x_c, x_d, u); \quad x_c(0) = x_{c0} \quad (\text{B.1})$$

$$x_d[k+1] = g(t, x_c, x_d, u); \quad x_d[0] = x_{d0} \quad (\text{B.2})$$

$$y = h(t, x_c, x_d, u), \quad (\text{B.3})$$

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 two different methods: a level-1 m-file s-function and a C-file s-function. The C-file s-function is compiled into C-code and executes much faster than m-file s-functions.

2.1 Example: Second-order Differential Equation

In this section we will show how to implement a system specified by the standard second-order transfer function

$$Y(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} U(s) \quad (\text{B.4})$$

using both a level-1 m-file s-function and a C-file s-function. The first step in either case is to represent Equation (B.4) in state space form. Using control canonical form [?], we have

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -2\zeta\omega_n & -\omega_n^2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u \quad (\text{B.5})$$

$$y = (0 \quad \omega_n^2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \quad (\text{B.6})$$

2.1.1 Level-1 M-file S-function

The code listing for an m-file s-function that implements the system described by Equations (B.5) and (B.6) is shown below. Line 1 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 `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 2–11. The block setup and the definition of the initial conditions are shown in Lines 13–27. The number of continuous states, discrete states, outputs, and inputs are defined in Lines 16–19, respectively. The direct feedthrough term on Line 20 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 24. The sample times are defined on Line 27. 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 30–32, and the output function $h(t, x, u)$ is defined in Lines 35–36. A Simulink file that calls this m-file s-function is included on the book website.

```

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 %=====
```

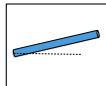
```

36     function y=mdlOutputs(t,x,u,wn)
37         y = wn^2*x(2);

```

2.2 Design Study A. Single Link Robot Arm

]



Homework Problem A.4

Modify the simulink model created in homework A.2 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.2.

Solution

The s-function is listed below.

```

1 function [sys,x0,str,ts,simStateCompliance] = ...
2                               arm_dynamics(t,x,u,flag,AP)
3 switch flag,
4
5 %%%%%%%%
6 % Initialization %
7 %%%%%%%%
8 case 0,
9     [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(AP);
10
11 %%%%%%%
12 % Derivatives %
13 %%%%%%
14 case 1,
15     sys=mdlDerivatives(t,x,u,AP);
16
17 %%%%%%
18 % Update %
19 %%%%%%
20 case 2,

```

```

21     sys=mdlUpdate(t,x,u);
22
23 %%%%%%
24 % Outputs %
25 %%%%%%
26 case 3,
27     sys=mdlOutputs(t,x,u,AP);
28
29 %%%%%%%%%%%%%%
30 % GetTimeOfNextVarHit %
31 %%%%%%%%%%%%%%
32 case 4,
33     sys=mdlGetTimeOfNextVarHit(t,x,u);
34
35 %%%%%%
36 % Terminate %
37 %%%%%%
38 case 9,
39     sys=mdlTerminate(t,x,u);
40
41 %%%%%%
42 % Unexpected flags %
43 %%%%%%
44 otherwise
45     DASTudio.error('Simulink:blocks:unhandledFlag',...
46                     num2str(flag));
47
48 end
49
50 % end sfuntmpl
51
52 %
53 %=====
54 % mdlInitializeSizes
55 % Return the sizes, initial conditions, and sample times
56 % for the S-function.
57 %=====
58 %
59 function [sys,x0,str,ts,simStateCompliance]...
60             =mdlInitializeSizes(AP)
61
62 sizes = simsizes;
63
64 sizes.NumContStates = 2;
65 sizes.NumDiscStates = 0;

```

```
66 sizes.NumOutputs      = 1;
67 sizes.NumInputs       = 1;
68 sizes.DirFeedthrough = 0;
69 sizes.NumSampleTimes = 1;
70
71 sys = simsizes(sizes);
72
73 %
74 % initialize the initial conditions
75 %
76 x0  = [AP.theta0; AP.thetadot0];
77
78 %
79 % str is always an empty matrix
80 %
81 str = [];
82
83 %
84 % initialize the array of sample times
85 %
86 ts  = [0 0];
87
88 simStateCompliance = 'UnknownSimState';
89
90 % end mdlInitializeSizes
91
92 %
93 %=====
94 % mdlDerivatives
95 % Return the derivatives for the continuous states.
96 %=====
97 %
98 function sys=mdlDerivatives(t,x,u,AP)
99     theta      = x(1);
100    thetadot  = x(2);
101    tau        = u(1);
102
103    thetaddot = (3/AP.m/AP.ell^2)*...
104                  (tau - AP.b*thetadot ...
105                  - AP.m*AP.g*AP.ell/2*cos(theta));
106
107 sys = [thetadot; thetaddot];
108
109 % end mdlDerivatives
110
```

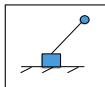
```

111 %
112 %
113 %=====
114 % mdlOutputs
115 % Return the block outputs.
116 %=====
117 %
118 function sys=mdlOutputs(t,x,u,P)
119
120 sys = x(1);
121
122 % end mdlOutputs

```

For a complete solution to this problem, see the wiki associated with this book.

2.3 Design Study B. Inverted Pendulum



Homework Problem B.4

Modify the simulink model created in homework B.2 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.2.

Solution

The s-function is listed below.

```

1
2 function [sys,x0,str,ts,simStateCompliance]...
3 %===== = pendulum_dynamics(t,x,u,flag,AP)
4 switch flag,
5
6 %%%%%%
7 % Initialization %
8 %%%%%%
9 case 0,
10 [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(AP);
11

```

```

12 %%%%%%
13 % Derivatives %
14 %%%%%%
15 case 1,
16     sys=mdlDerivatives(t,x,u,AP);
17
18 %%%%%%
19 % Update %
20 %%%%%%
21 case 2,
22     sys=mdlUpdate(t,x,u);
23
24 %%%%%%
25 % Outputs %
26 %%%%%%
27 case 3,
28     sys=mdlOutputs(t,x,u);
29
30 %%%%%%
31 % GetTimeOfNextVarHit %
32 %%%%%%
33 case 4,
34     sys=mdlGetTimeOfNextVarHit(t,x,u);
35
36 %%%%%%
37 % Terminate %
38 %%%%%%
39 case 9,
40     sys=mdlTerminate(t,x,u);
41
42 %%%%%%
43 % Unexpected flags %
44 %%%%%%
45 otherwise
46     DASTUDIO.error('Simulink:blocks:unhandledFlag',...
47                         num2str(flag));
48
49 end
50
51 % end sfunmpl
52
53 %
54 %=====
55 % mdlInitializeSizes
56 % Return the sizes, initial conditions, and sample times

```

```

57 % for the S-function.
58 %=====
59 %
60 function [sys,x0,str,ts,simStateCompliance]...
61 %mdlInitializeSizes(AP)
62
63 sizes = simsizes;
64
65 sizes.NumContStates = 4;
66 sizes.NumDiscStates = 0;
67 sizes.NumOutputs = 2;
68 sizes.NumInputs = 1;
69 sizes.DirFeedthrough = 0;
70 sizes.NumSampleTimes = 1;
71
72 sys = simsizes(sizes);
73
74 %
75 % initialize the initial conditions
76 %
77 x0 = [AP.z0; AP.theta0; AP.zdot0; AP.thetadot0];
78
79 %
80 % str is always an empty matrix
81 %
82 str = [];
83
84 %
85 % initialize the array of sample times
86 %
87 ts = [0 0];
88
89 simStateCompliance = 'UnknownSimState';
90
91 % end mdlInitializeSizes
92
93 %
94 %=====
95 % mdlDerivatives
96 % Return the derivatives for the continuous states.
97 %=====
98 %
99 function sys=mdlDerivatives(t,x,u,AP)
100 z = x(1);
101 theta = x(2);

```

```

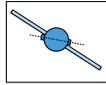
102     zdot      = x(3);
103     thetadot = x(4);
104     F        = u(1);
105
106     M = [ ...
107         AP.m1+AP.m2,           AP.m1*AP.ell*cos(theta); ...
108         AP.m1*AP.ell*cos(theta), AP.m1*AP.ell^2; ...
109     ];
110     c = [ ...
111         AP.m1*AP.ell*thetadot^2*sin(theta) + F - AP.b*zdot; ...
112         AP.m1*AP.g*AP.ell*sin(theta); ...
113     ];
114     tmp = inv(M)*c;
115     zddot    = tmp(1);
116     thetaddot = tmp(2);
117
118 sys = [zdot; thetadot; zddot; thetaddot];
119
120 % end mdlDerivatives
121 %
122 %
123 %=====
124 % mdlOutputs
125 % Return the block outputs.
126 %=====
127 %
128 function sys=mdlOutputs(t,x,u,AP)
129     z      = x(1);
130     theta = x(2);
131     zdot   = x(3);
132     thetadot = x(4);
133 sys = [z; theta];
134
135 % end mdlOutputs

```

For a complete solution to this problem, see the wiki associated with this book.

Solution

2.4 Design Study C. Satellite Attitude Control



Homework Problem C.4

Modify the simulink model created in homework C.2 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 C.2.

Solution

The s-function is listed below.

```

1 function [sys,x0,str,ts,simStateCompliance]...
2                               = satellite_dynamics(t,x,u,flag,AP)
3 switch flag,
4
5 %%%%%%
6 % Initialization %
7 %%%%%%
8 case 0,
9     [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(AP);
10
11 %%%%%%
12 % Derivatives %
13 %%%%%%
14 case 1,
15     sys=mdlDerivatives(t,x,u,AP);
16
17 %%%%%%
18 % Update %
19 %%%%%%
20 case 2,
21     sys=mdlUpdate(t,x,u);
22
23 %%%%%%
24 % Outputs %
25 %%%%%%
26 case 3,
```

```

27     sys=mdlOutputs(t,x,u,AP);
28
29 %%%%%%%%%%%%%%
30 % GetTimeOfNextVarHit %
31 %%%%%%%%%%%%%%
32 case 4,
33     sys=mdlGetTimeOfNextVarHit(t,x,u);
34
35 %%%%%%%%%%%%%%
36 % Terminate %
37 %%%%%%%%%%%%%%
38 case 9,
39     sys=mdlTerminate(t,x,u);
40
41 %%%%%%%%%%%%%%
42 % Unexpected flags %
43 %%%%%%%%%%%%%%
44 otherwise
45     DASTudio.error('Simulink:blocks:unhandledFlag',...
46                     num2str(flag));
47 end
48
49
50 % end sfunmpl
51
52 %
53 =====
54 % mdlInitializeSizes
55 % Return the sizes, initial conditions, and sample times
56 % for the S-function.
57 =====
58 %
59 function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(AP)
60
61 sizes = simsizes;
62
63 sizes.NumContStates = 4;
64 sizes.NumDiscStates = 0;
65 sizes.NumOutputs = 2;
66 sizes.NumInputs = 1;
67 sizes.DirFeedthrough = 0;
68 sizes.NumSampleTimes = 1;
69
70 sys = simsizes(sizes);
71

```

```

72 %
73 % initial conditions
74 %
75 x0 = [AP.theta0; AP.phi0; AP.thetadot0; AP.phidot0];
76
77 %
78 % str is always an empty matrix
79 %
80 str = [];
81
82 %
83 % initialize the array of sample times
84 %
85 ts = [0 0];
86
87 simStateCompliance = 'UnknownSimState';
88
89 % end mdlInitializeSizes
90
91 %
92 %=====
93 % mdlDerivatives
94 % Return the derivatives for the continuous states.
95 %=====
96 %
97 function sys=mdlDerivatives(t,x,u,AP)
98     theta      = x(1);
99     phi        = x(2);
100    thetadot   = x(3);
101    phidot     = x(4);
102    tau        = u(1);
103
104    M = [
105        AP.Js, 0; 0, AP.Jp;
106    ];
107    c = [
108        tau - AP.b*(thetadot-phidot)-AP.k*(theta-phi);
109        -AP.b*(phidot-thetadot);
110    ];
111
112    tmp = inv(M)*c;
113    thetaddot = tmp(1);
114    phiddot   = tmp(2);
115
116    sys = [thetadot; phidot; thetaddot; phiddot];

```

```
117 % end mdlDerivatives
118
119
120
121 %
122 %=====
123 % mdlOutputs
124 % Return the block outputs.
125 %=====
126 %
127 function sys=mdlOutputs(t,x,u,AP)
128
129 sys = x(1:2);
130
131 % end mdlOutputs
```

For a complete solution to this problem, see the wiki associated with this book.

Appendix C

Simple Derivation of the Euler Lagrange Equations

RWB: Insert simple derivation of Euler-Lagrange equation. Something like the following:

American Journal of Physics, Vol. 72, No. 4, pp. 510–513, April 2004
©2004 American Association of Physics Teachers. All rights reserved.

Deriving Lagrange's equations using elementary calculus

Jozef Hanc^{a)}

Technical University, Vysokoskolska 4, 042 00 Kosice, Slovakia

Edwin F. Taylor^{b)}

Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Slavomir Tuleja^{c)}

Gymnázium arm. gen. L. Svobodu, Komenskeho 4, 066 51 Humenne, Slovakia

Received: 30 December 2002; accepted: 20 June 2003

We derive Lagrange's equations of motion from the principle of least action using elementary calculus rather than the calculus of variations. We also demonstrate the conditions under which energy and momentum are constants of the motion. © 2004 American Association of Physics Teachers.

Contents

- [I. INTRODUCTION](#)
- [II. DIFFERENTIAL APPROXIMATION TO THE PRINCIPLE OF LEAST ACTION](#)
- [III. DERIVATION OF LAGRANGE'S EQUATION](#)
- [IV. MOMENTUM AND ENERGY AS CONSTANTS OF THE MOTION](#)
 - [A. Momentum](#)
 - [B. Energy](#)
- [V. SUMMARY](#)
- [ACKNOWLEDGMENT](#)
- [APPENDIX: EXTENSION TO MULTIPLE DEGREES OF FREEDOM](#)

- [REFERENCES](#)
- [FIGURES](#)
- [FOOTNOTES](#)

I. INTRODUCTION

The equations of motion¹ of a mechanical system can be derived by two different mathematical methods—vectorial and analytical. Traditionally, introductory mechanics begins with Newton's laws of motion which relate the force, momentum, and acceleration vectors. But we frequently need to describe systems, for example, systems subject to constraints without friction, for which the use of vector forces is cumbersome. Analytical mechanics in the form of the Lagrange equations provides an alternative and very powerful tool for obtaining the equations of motion. Lagrange's equations employ a single scalar function, and there are no annoying vector components or associated trigonometric manipulations. Moreover, the analytical approach using Lagrange's equations provides other capabilities² that allow us to analyze a wider range of systems than Newton's second law.

The derivation of Lagrange's equations in advanced mechanics texts³ typically applies the calculus of variations to the principle of least action. The calculus of variation belongs to important branches of mathematics, but is not widely taught or used at the college level. Students often encounter the variational calculus first in an advanced mechanics class, where they struggle to apply a new mathematical procedure to a new physical concept. This paper provides a derivation of Lagrange's equations from the principle of least action using elementary calculus,⁴ which may be employed as an alternative to (or a preview of) the more advanced variational calculus derivation.

In Sec. II we develop the mathematical background for deriving Lagrange's equations from elementary calculus. Section III gives the derivation of the equations of motion for a single particle. Section IV extends our approach to demonstrate that the energy and momentum are constants of the motion. The Appendix expands Lagrange's equations to multiparticle systems and adds angular momentum as an example of generalized momentum.

II. DIFFERENTIAL APPROXIMATION TO THE PRINCIPLE OF LEAST ACTION

A particle moves along the x axis with potential energy $V(x)$ which is time independent. For this special case the Lagrange function or Lagrangian L has the form:⁵

$$L(x, v) = T - V = \frac{1}{2}mv^2 - V(x). \quad (1)$$

The action S along a world line is defined as

$$S = \int_{\text{along the world line}} L(x, v) dt. \quad (2)$$

The principle of least action requires that between a fixed initial event and a fixed final event the particle follow a world line such that the action S is a minimum.

The action S is an additive scalar quantity, and is the sum of contributions $L\Delta t$ from each segment along the entire world line between two events fixed in space and time. Because S is additive, it follows that the principle of least action must hold for each individual infinitesimal segment of the world line.⁶ This property allows us to pass from the integral equation for the principle of least action, Eq. (2), to Lagrange's differential equation, valid anywhere along the world line. It also allows us to use elementary calculus in this derivation.

We approximate a small section of the world line by two straight-line segments connected in the middle (Fig. 1) and make the following approximations: The average position coordinate in the Lagrangian along a segment is at the midpoint of the segment.⁷ The average velocity of the particle is equal to its displacement across the segment divided by the time interval of the segment. These approximations applied to segment A in Fig. 1 yield the average Lagrangian L_A and action S_A contributed by this segment:

$$L_A \equiv L\left(\frac{x_1 + x_2}{2}, \frac{x_2 - x_1}{\Delta t}\right), \quad (3a)$$

$$S_A \approx L_A \Delta t = L\left(\frac{x_1 + x_2}{2}, \frac{x_2 - x_1}{\Delta t}\right) \Delta t, \quad (3b)$$

with similar expressions for L_B and S_B along segment B .

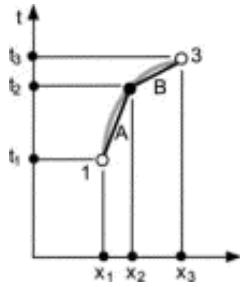


Figure 1.

III. DERIVATION OF LAGRANGE'S EQUATION

We employ the approximations of Sec. II to derive Lagrange's equations for the special case introduced there. As shown in Fig. 2, we fix events 1 and 3 and vary the x coordinate of the intermediate event to minimize the action between the outer two events.

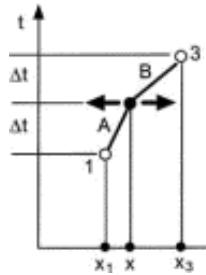


Figure 2.

For simplicity, but without loss of generality, we choose the time increment Δt to be the same for each segment, which also equals the time between the midpoints of the two segments. The average positions and velocities along segments A and B are

$$x_A = \frac{x_1 + x}{2}, \quad v_A = \frac{x - x_1}{\Delta t}, \quad (4a)$$

$$x_B = \frac{x + x_3}{2}, \quad v_B = \frac{x_3 - x}{\Delta t}. \quad (4b)$$

The expressions in Eq. (4) are all functions of the single variable x . For later use we take the derivatives of Eq. (4) with respect to x :

$$\frac{dx_A}{dx} = \frac{1}{2}, \quad \frac{dv_A}{dx} = +\frac{1}{\Delta t}, \quad (5a)$$

$$\frac{dx_B}{dx} = \frac{1}{2}, \quad \frac{dv_B}{dx} = -\frac{1}{\Delta t}. \quad (5b)$$

Let L_A and L_B be the values of the Lagrangian on segments A and B , respectively, using Eq. (4), and label the summed action across these two segments as S_{AB} :

$$S_{AB} = L_A \Delta t + L_B \Delta t. \quad (6)$$

The principle of least action requires that the coordinates of the middle event x be chosen to yield the smallest value of the action between the fixed events 1 and 3. If we set the derivative of S_{AB} with respect to x equal to zero⁸ and use the chain rule, we obtain

$$\frac{dS_{AB}}{dx} = 0 = \frac{\partial L_A}{\partial x_A} \frac{dx_A}{dx} \Delta t + \frac{\partial L_A}{\partial v_A} \frac{dv_A}{dx} \Delta t + \frac{\partial L_B}{\partial x_B} \frac{dx_B}{dx} \Delta t + \frac{\partial L_B}{\partial v_B} \frac{dv_B}{dx} \Delta t.$$

We substitute Eq. (5) into Eq. (7), divide through by Δt , and regroup the terms to obtain

$$\frac{1}{2} \left(\frac{\partial L_A}{\partial x_A} + \frac{\partial L_B}{\partial x_B} \right) - \frac{1}{\Delta t} \left(\frac{\partial L_B}{\partial v_B} - \frac{\partial L_A}{\partial v_A} \right) = 0. \quad (8)$$

To first order, the first term in Eq. (8) is the average value of $\partial L/\partial x$ on the two segments A and B . In the limit $\Delta t \rightarrow 0$, this term approaches the value of the partial derivative at x . In the same limit, the second term in Eq. (8) becomes the time derivative of the partial derivative of the Lagrangian with respect to velocity $d(\partial L/\partial v)/dt$. Therefore in the limit $\Delta t \rightarrow 0$, Eq. (8) becomes the Lagrange equation in x :

$$\frac{\partial L}{\partial x} - \frac{d}{dt} \left(\frac{\partial L}{\partial v} \right) = 0. \quad (9)$$

We did not specify the location of segments A and B along the world line. The additive property of the action implies that Eq. (9) is valid for every adjacent pair of segments.

An essentially identical derivation applies to any particle with one degree of freedom in any potential. For example, the single angle φ tracks the motion of a simple pendulum, so its equation of motion follows from Eq. (9) by replacing x with φ without the need to take vector components.

IV. MOMENTUM AND ENERGY AS CONSTANTS OF THE MOTION

A. Momentum

We consider the case in which the Lagrangian does not depend explicitly on the x coordinate of the particle (for example, the potential is zero or independent of position). Because it does not appear in the Lagrangian, the x coordinate is "ignorable" or "cyclic." In this case a simple and well-known conclusion from Lagrange's equation leads to the momentum as a conserved quantity, that is, a constant of motion. Here we provide an outline of the derivation.

For a Lagrangian that is only a function of the velocity, $L = L(v)$, Lagrange's equation (9) tells us that the time derivative of $\partial L / \partial v$ is zero. From Eq. (1), we find that $\partial L / \partial v = mv$, which implies that the x momentum, $p = mv$, is a constant of the motion.

This usual consideration can be supplemented or replaced by our approach. If we repeat the derivation in Sec. III with $L = L(v)$ (perhaps as a student exercise to reinforce understanding of the previous derivation), we obtain from the principle of least action

$$\frac{dS_{AB}}{dx} = 0 = \frac{\partial L_A}{\partial v_A} \frac{dv_A}{dx} \Delta t + \frac{\partial L_B}{\partial v_B} \frac{dv_B}{dx} \Delta t. \quad (10)$$

We substitute Eq. (5) into Eq. (10) and rearrange the terms to find:

$$\frac{\partial L_A}{\partial v_A} = \frac{\partial L_B}{\partial v_B}$$

or

$$p_A = p_B. \quad (11)$$

Again we can use the arbitrary location of segments A and B along the world line to conclude that the momentum p is a constant of the motion everywhere on the world line.

B. Energy

Standard texts⁹ obtain conservation of energy by examining the time derivative of a Lagrangian that does not depend explicitly on time. As pointed out in Ref. 9, this lack of dependence of the Lagrangian implies the homogeneity of time: temporal translation has no influence on the form of the Lagrangian. Thus conservation of energy is closely connected to the symmetry properties of nature.¹⁰ As we will see, our elementary calculus approach offers an alternative way¹¹ to derive energy conservation.

Consider a particle in a time-independent potential $V(x)$. Now we vary the time of the middle event (Fig. 3), rather than its position, requiring that this time be chosen to minimize the action.

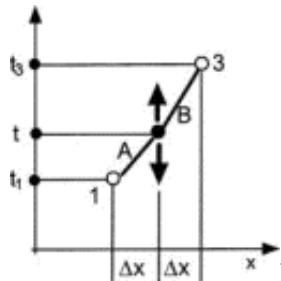


Figure 3.

For simplicity, we choose the x increments to be equal, with the value Δx . We keep the spatial coordinates of all three events fixed while varying the time coordinate of the middle event and obtain

$$v_A = \frac{\Delta x}{t - t_1}, \quad v_B = \frac{\Delta x}{t_3 - t}. \quad (12)$$

These expressions are functions of the single variable t , with respect to which we take the derivatives

$$\frac{dv_A}{dt} = -\frac{\Delta x}{(t - t_1)^2} = -\frac{v_A}{t - t_1}, \quad (13a)$$

and

$$\frac{dv_B}{dt} = \frac{\Delta x}{(t_3 - t)^2} = \frac{v_B}{t_3 - t}. \quad (13b)$$

Despite the form of Eq. (13), the derivatives of velocities are *not* accelerations, because the x separations are held constant while the time is varied.

As before [see Eq.(6)],

$$S_{AB} = L_A(t - t_1) + L_B(t_3 - t). \quad (14)$$

Note that students sometimes misinterpret the time differences in parentheses in Eq. (14) as arguments of L .

We find the value of the time t for the action to be a minimum by setting the derivative of S_{AB} equal to zero:

$$\frac{dS_{AB}}{dt} = 0 = \frac{\partial L_A}{\partial v_A} \frac{dv_A}{dt} (t - t_1) + L_A + \frac{\partial L_B}{\partial v_B} \frac{dv_B}{dt} (t - t_3)$$

If we substitute Eq. (13) into Eq. (15) and rearrange the result, we find

$$\frac{\partial L_A}{\partial v_A} v_A - L_A = \frac{\partial L_B}{\partial v_B} v_B - L_B. \quad (16)$$

Because the action is additive, Eq. (16) is valid for every segment of the world line and identifies the function $v \frac{\partial L}{\partial v} - L$ as a constant of the motion. By substituting Eq. (1) for the Lagrangian into $v \frac{\partial L}{\partial v} - L$ and carrying out the partial derivatives, we can show that the constant of the motion corresponds to the total energy $E = T + V$.

V. SUMMARY

Our derivation and the extension to multiple degrees of freedom in the Appendix allow the introduction of Lagrange's equations and its connection to the principle of least action without the apparatus of the calculus of variations. The derivations also may be employed as a preview of Lagrangian mechanics before its more formal derivation using variational calculus.

One of us (ST) has successfully employed these derivations and the resulting Lagrange equations with a small group of talented high school students. They used the equations to solve problems presented in the Physics Olympiad. The excitement and enthusiasm of these students leads us to hope that others will undertake trials with larger numbers and a greater variety of students.

ACKNOWLEDGMENT

The authors would like to express thanks to an anonymous referee for his or her valuable criticisms and suggestions, which improved this paper.

APPENDIX: EXTENSION TO MULTIPLE DEGREES OF FREEDOM

We discuss Lagrange's equations for a system with multiple degrees of freedom, without pausing to discuss the usual conditions assumed in the derivations, because these can be found in standard advanced mechanics texts.³

Consider a mechanical system described by the following Lagrangian:

$$L = L(q_1, q_2, \dots, q_s, \dot{q}_1, \dot{q}_2, \dots, \dot{q}_s, t), \quad (17)$$

where the q are independent generalized coordinates and the dot over q indicates a derivative with respect to time. The subscript s indicates the number of degrees of freedom of the system. Note that we have generalized to a Lagrangian that is an explicit function of time t . The specification of all the values of all the generalized coordinates q_i in Eq. (17) defines a configuration of the system. The action S summarizes the evolution of the system as a whole from an initial configuration to a final configuration, along what might be called a world line through multidimensional space–time. Symbolically we write:

$$S = \int_{\text{initial configuration}}^{\text{to final configuration}} L(q_1, q_2 \dots q_s, \dot{q}_1, \dot{q}_2 \dots \dot{q}_s)$$

The generalized principle of least action requires that the value of S be a minimum for the actual evolution of the system symbolized in Eq. (18). We make an argument similar to that in Sec. III for the one-dimensional motion of a particle in a potential. If the principle of least action holds for the entire world line through the intermediate configurations of L in Eq. (18), it also holds for an infinitesimal change in configuration anywhere on this world line.

Let the system pass through three infinitesimally close configurations in the ordered sequence 1, 2, 3 such that all generalized coordinates remain fixed except for a single coordinate q at configuration 2. Then the increment of the action from configuration 1 to configuration 3 can be considered to be a function of the single variable q . As a consequence, for each of the s degrees of freedom, we can make an argument formally identical to that carried out from Eq. (3) through Eq. (9). Repeated s times, once for each generalized coordinate q_i , this derivation leads to s scalar Lagrange equations that describe the motion of the system:

$$\frac{\partial L}{\partial q_i} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) = 0 \quad (i = 1, 2, 3, \dots, s). \quad (19)$$

The inclusion of time explicitly in the Lagrangian (17) does not affect these derivations, because the time coordinate is held fixed in each equation.

Suppose that the Lagrangian (17) is not a function of a given coordinate q_k . An argument similar to that in Sec. IV A tells us that the corresponding generalized

momentum $\partial L/\partial \dot{q}_k$ is a constant of the motion. As a simple example of such a generalized momentum, we consider the angular momentum of a particle in a central potential. If we use polar coordinates r, θ to describe the motion of a single particle in the plane, then the Lagrangian has the form $L = T - V = m(\dot{r}^2 + r^2\dot{\theta}^2)/2 - V(r)$, and the angular momentum of the system is represented by $\partial L/\partial \dot{\theta}$.

If the Lagrangian (17) is not an explicit function of time, then a derivation formally equivalent to that in Sec. IVB (with time as the single variable) shows

that the function $(\sum_i \dot{q}_i \partial L / \partial \dot{q}_i) - L$, sometimes called¹² the energy function h , is a constant of the motion of the system, which in the simple cases we cover¹³ can be interpreted as the total energy E of the system.

If the Lagrangian (17) depends explicitly on time, then this derivation yields the equation $dh/dt = -\partial L/\partial t$.

REFERENCES

[Citation links](#) [e.g., [Phys. Rev. D 40, 2172 \(1989\)](#)] go to online journal abstracts. Other links (see [Reference Information](#)) are available with your current login. Navigation of links may be more efficient using a [second browser window](#).

1. We take "equations of motion" to mean relations between the accelerations, velocities, and coordinates of a mechanical system. See L. D. Landau and E. M. Lifshitz, *Mechanics* (Butterworth-Heinemann, Oxford, 1976), Chap. 1, Sec. 1. [first citation in article](#)
2. Besides its expression in scalar quantities (such as kinetic and potential energy), Lagrangian quantities lead to the reduction of dimensionality of a problem, employ the invariance of the equations under point transformations, and lead directly to constants of the motion using Noether's theorem. More detailed explanation of these features, with a comparison of analytical mechanics to vectorial mechanics, can be found in Cornelius Lanczos, *The Variational Principles of Mechanics* (Dover, New York, 1986), pp. xxi–xxix. [first citation in article](#)

3. Chapter 1 in Ref. 1 and Chap. V in Ref. 2; Gerald J. Sussman and Jack Wisdom, *Structure and Interpretation of Classical Mechanics* (MIT, Cambridge, 2001), Chap. 1; Herbert Goldstein, Charles Poole, and John Safko, *Classical Mechanics* (Addison–Wesley, Reading, MA, 2002), 3rd ed., Chap. 2. An alternative method derives Lagrange's equations from D'Alambert principle; see Goldstein, Sec. 1.4. [first citation in article](#)
4. Our derivation is a modification of the finite difference technique employed by Euler in his path-breaking 1744 work, "The method of finding plane curves that show some property of maximum and minimum." Complete references and a description of Euler's original treatment can be found in Herman H. Goldstine, *A History of the Calculus of Variations from the 17th Through the 19th Century* (Springer-Verlag, New York, 1980), Chap. 2. Cornelius Lanczos (Ref. 2, pp. 49–54) presents an abbreviated version of Euler's original derivation using contemporary mathematical notation. [first citation in article](#)
5. R. P. Feynman, R. B. Leighton, and M. Sands, *The Feynman Lectures on Physics* (Addison–Wesley, Reading, MA, 1964), Vol. 2, Chap. 19. [first citation in article](#)
6. See Ref. 5, p. 19-8 or in more detail, J. Hanc, S. Tuleja, and M. Hancova, "Simple derivation of Newtonian mechanics from the principle of least action," *Am. J. Phys.* **71** (4), 386–391 (2003). [\[ISI\]](#) [first citation in article](#)
7. There is no particular reason to use the midpoint of the segment in the Lagrangian of Eq. (2). In Riemann integrals we can use any point on the given segment. For example, all our results will be the same if we used the coordinates of either end of each segment instead of the coordinates of the midpoint. The repositioning of this point can be the basis of an exercise to test student understanding of the derivations given here. [first citation in article](#)
8. A zero value of the derivative most often leads to the world line of minimum action. It is possible also to have a zero derivative at an inflection point or saddle point in the action (or the multidimensional equivalent in configuration space). So the most general term for our basic law is the principle of stationary action. The conditions that guarantee the existence of a minimum can be found in I. M. Gelfand and S. V. Fomin, *Calculus of Variations* (Prentice–Hall, Englewood Cliffs, NJ, 1963). [first citation in article](#)
9. Reference 1, Chap. 2 and Ref. 3, Goldstein et al., Sec. 2.7. [first citation in article](#)

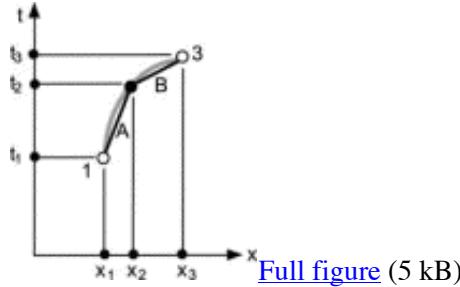
10. The most fundamental justification of conservation laws comes from symmetry properties of nature as described by Noether's theorem. Hence energy conservation can be derived from the invariance of the action by temporal translation and conservation of momentum from invariance under space translation. See N. C. Bobillo-Ares, "Noether's theorem in discrete classical mechanics," [Am. J. Phys. 56 \(2\), 174–177 \(1988\)](#) or C. M. Giordano and A. R. Plastino, "Noether's theorem, rotating potentials, and Jacobi's integral of motion," [ibid. 66 \(11\), 989–995 \(1998\)](#). [first citation in article](#)
11. Our approach also can be related to symmetries and Noether's theorem, which is the main subject of J. Hanc, S. Tuleja, and M. Hancova, "Symmetries and conservation laws: Consequences of Noether's theorem," Am. J. Phys. (to be published). [first citation in article](#)
12. Reference 3, Goldstein et al., Sec. 2.7. [first citation in article](#)
13. For the case of generalized coordinates, the energy function h is generally not the same as the total energy. The conditions for conservation of the energy function h are distinct from those that identify h as the total energy. For a detailed discussion see Ref. 12. Pedagogically useful comments on a particular example can be found in A. S. de Castro, "Exploring a rheonomic system," [Eur. J. Phys. 21, 23–26 \(2000\) \[Inspec\]](#) and C. Ferrario and A. Passerini, "Comment on Exploring a rheonomic system," [ibid. 22, L11–L14 \(2001\)](#). [\[Inspec\] \[ISI\]](#) [first citation in article](#)

CITING ARTICLES

This list contains links to other [online articles](#) that cite the article currently being viewed.

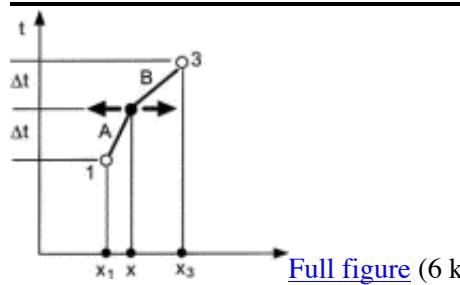
1. Hamilton's principle: Why is the integrated difference of the kinetic and potential energy minimized?
Alberto G. Rojo, [Am. J. Phys. 73, 831 \(2005\)](#)

FIGURES



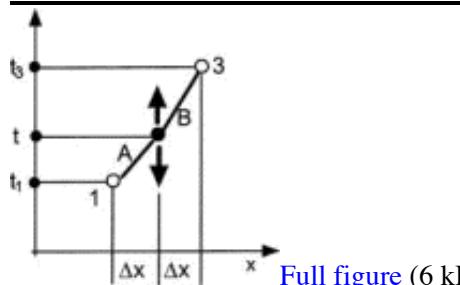
[Full figure \(5 kB\)](#)

Fig. 1. An infinitesimal section of the world line approximated by two straight line segments. [First citation in article](#)



[Full figure \(6 kB\)](#)

Fig. 2. Derivation of Lagrange's equations from the principle of least action. Points 1 and 3 are on the true world line. The world line between them is approximated by two straight line segments (as in Fig. 1). The arrows show that the x coordinate of the middle event is varied. All other coordinates are fixed. [First citation in article](#)



[Full figure \(6 kB\)](#)

Fig. 3. A derivation showing that the energy is a constant of the motion. Points 1 and 3 are on the true world line, which is approximated by two straight line

segments (as in Figs. 1 and 2). The arrows show that the t coordinate of the middle event is varied. All other coordinates are fixed. [First citation in article](#)

FOOTNOTES

^aElectronic mail: jozef.hanc@tuke.sk

^bElectronic mail: eftaylor@mit.edu; <http://www.eftaylor.com>

^cElectronic mail: tuleja@stonline.sk

Up: [Issue Table of Contents](#)

Go to: [Previous Article](#) | [Next Article](#)

Other formats: [HTML \(smaller files\)](#) | [PDF \(83 kB\)](#)

Appendix D

Linear Algebra Review

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,

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 \\ 6 \end{pmatrix} = \begin{pmatrix} 17 \\ 39 \end{pmatrix}.$$

Multiplying two matrices also requires matching dimensions:

$$\begin{aligned} \underbrace{\begin{matrix} A \\ m \times n \end{matrix}}_{\text{rows}} \underbrace{\begin{matrix} B \\ n \times p \end{matrix}}_{\text{columns}} &= \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} b_{11} & \dots & b_{1n} \\ \vdots & & \vdots \\ b_{m1} & \dots & b_{mn} \end{pmatrix} \\ &= \begin{pmatrix} \sum_{i=1}^n a_{1i}b_{i1} & \dots & \sum_{i=1}^n a_{1i}b_{ip} \\ \vdots & & \vdots \\ \sum_{i=1}^n a_{mi}b_{i1} & \dots & \sum_{i=1}^n a_{mi}b_{ip} \end{pmatrix} = \underbrace{\begin{matrix} C \\ m \times p \end{matrix}}_{\text{rows}}. \end{aligned}$$

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

$$AB \neq BA.$$

4.1 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

$$\begin{aligned} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^\top &= (x_1 \ x_2 \ x_3), \\ \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}^\top &= \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}. \end{aligned}$$

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 $A_s = 1/2(A + A^\top)$ is symmetric, and $A_s s = 1/2(A - A^\top)$ is skew-symmetric.

The following properties are easily verified by direct manipulation:

$$\begin{aligned}(AB)^\top &= B^\top A^\top \\ (A + B)^\top &= A^\top + B^\top.\end{aligned}$$

4.2 Trace

The trace operator is defined for squared matrices. If $A \in \mathbb{R}^{n \times n}$, then

$$\text{tr}(A) = \sum_{i=1}^n a_{ii}.$$

For example

$$\text{tr} \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = 1 + 4 = 5.$$

4.3 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 :

$$\text{adj}(A) = \begin{pmatrix} \gamma_{11} & \cdots & \gamma_{n1} \\ \vdots & & \vdots \\ \gamma_{1n} & \cdots & \gamma_{nn} \end{pmatrix}.$$

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 diagonal matrix. Let

$$E = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

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,

$$\det \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 0 & 0 & 0 & 8 & 9 \\ 0 & 0 & 0 & 9 & 10 \end{pmatrix} = \det \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{pmatrix} \cdot \det \begin{pmatrix} 8 & 9 \\ 9 & 10 \end{pmatrix}.$$

4.4 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

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

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 we have

$$\begin{aligned} \begin{pmatrix} A & B \\ C & D \end{pmatrix} &= \begin{pmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1} \end{pmatrix} \\ &= \begin{pmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{pmatrix}, \end{aligned}$$

from which we get the famous matrix inversion lemma

$$(A - BD^{-1}C)^{-1} = A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1}.$$

When $C = 0$ we get the simple formula

$$\begin{pmatrix} A & B \\ 0 & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}BD^{-1} \\ 0 & D^{-1} \end{pmatrix}.$$

4.5 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 eigenvalue 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 .

To find the eigenvalue 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 .

4.6 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

$$c_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + c_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

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

$$\begin{aligned} \text{rank} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} &= 2 \\ \text{rank} \begin{pmatrix} 1 & 3 \\ 0 & 0 \end{pmatrix} &= 1. \end{aligned}$$

4.7 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,

$$A = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 0 & 6 & 7 & 8 \\ 0 & 9 & 10 & 11 \\ 0 & 0 & 0 & 12 \end{pmatrix}$$

is an upper triangular matrix.

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.