Control Theory: A Brief Introduction

Brendon Anderson[1] Simon Rufer[1]

August 13, 2018

[1]Bruin Racing | Baja SAE at University of California, Los Angeles

Contents

1 Terminology **1**

2 Dynamical Systems **5**
2.1 Motivation . . . 5
2.2 Linear, Time-Invariant Systems . . . 5
2.3 Laplace Transform . . . 9
2.4 Unit Impulse Response . . . 10
2.5 State-Space Representations . . . 11
2.6 Poles and Stability . . . 13
2.7 Frequency Response . . . 15

3 Feedback Control **19**
3.1 Motivation . . . 19
3.2 Time Domain Specifications . . . 19
3.3 Introduction to Controller Design . . . 20
3.4 Root Locus Plots . . . 25
3.5 Common Controllers . . . 30
3.5.1 P Controller . . . 30
3.5.2 PD Controller . . . 31
3.5.3 PI Controller . . . 33
3.5.4 PID Controller . . . 34
3.5.5 Notch Filter . . . 38
3.6 Computer Design of Feedback Controllers . . . 39

4 Additional Topics in Control Theory **47**

Appendix A: Rules for Sketching Root Locus Plots **48**

1 Terminology

1. *System*: An entity or connection of entities whose behavior is of interest. May be physical or not.
2. *State*: A degree of freedom; a quantity that may change through time whose value describes the current behavior or configuration of a system.
3. *Dynamical System*: A system whose states change through time.

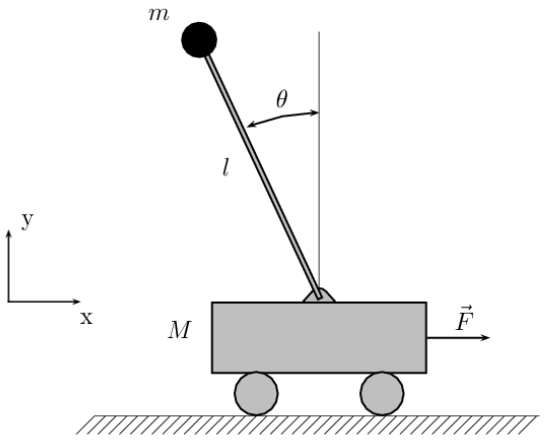

Figure 1: The inverted pendulum is an example of a dynamical system. The angle θ of the pendulum is a state of the system. The angular velocity, $\dot{\theta}$, might also be taken as a state of the system.

4. *Steady-State Dynamics*: Dynamics of a system that persist indefinitely.
5. *Transient Dynamics*: Dynamics of a system that do not persist indefinitely.

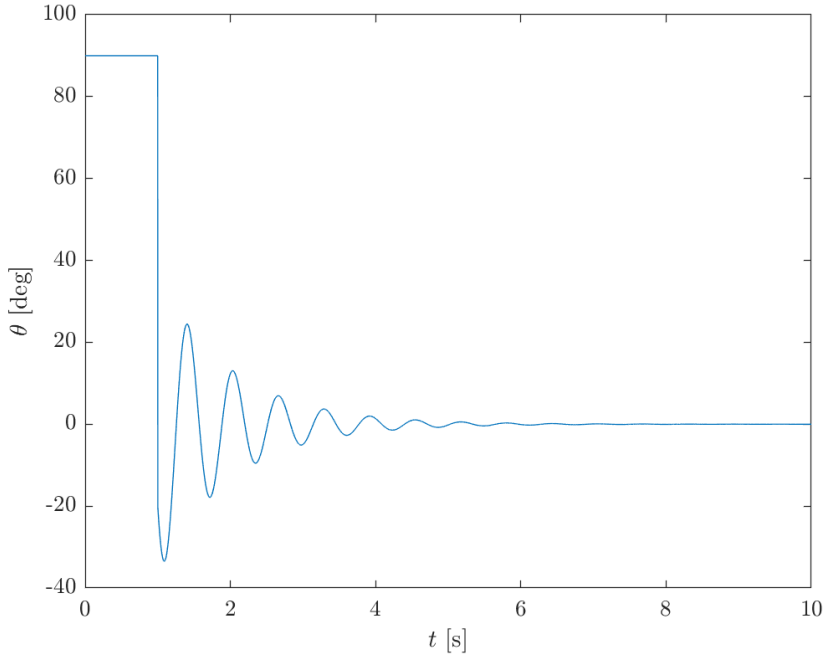

Figure 2: Plot of inverted pendulum angle versus time. Note that the pendulum is initially resting at $\theta_0 = 90°$. The pendulum is then driven to its upright position by either a motor or translational force onto the cart. The dynamics are transient for $t \in [1\,\text{s}, 7\,\text{s}]$ and steady-state for $t > 7\,\text{s}$. This plot represents the classical "swing up" control problem.

6. *Signal*: A time-dependent function. Typically signals are mathematical functions which describe the behavior of a state through time.

   EXAMPLE 1: The signal describing pendulum angle in the swing up of Figure 2 is $\theta(t) = 100e^{-t}\sin(10t),\ t \geq 1\,\text{s}$, where $\theta(t)$ is in units of degrees. △

7. *Input*: A signal which enters or drives a system.

   EXAMPLE 2: The input to the inverted pendulum might be either a torque, $\tau(t)$, imposed onto the base of the pendulum arm by a motor, or a translational force, $F(t)$, imposed onto the cart at the bottom of the pendulum. △

8. *Output*: A signal leaving a system. Typically this is something we measure with a sensor. Often times, the output of a system is one of its states.

   EXAMPLE 3: The output of the inverted pendulum might be either the angle, $\theta(t)$, or the angular velocity, $\dot{\theta}(t)$. △

9. *Controller*: An entity that manipulates signals. One way of conceptualizing controllers is as mathematical constructs or algorithms that are implemented via computers that influence the way a dynamical system moves through time. This concept will become more clear later on.

10. *Actuator*: An entity that generates input signals.

    Example 4: A DC motor which rotates the pendulum arm is an actuator. △

11. *Sensor*: An entity that measures output signals.

    Example 5: A rotary encoder is a sensor that may be mounted at the base of the pendulum arm in order to sense angular position. △

12. *Control System*: A combination of systems, actuators, sensors, signals, and controllers. Typically, a control system is designed in order to enhance or manipulate the behavior of some "original" system whose uncontrolled behavior is undesirable.

    Example 6: The inverted pendulum, without an actuator, sensor, and controller, would simply fall over due to its inherent instabilities! Therefore, a control system must be designed to maintain the pendulum at its upright position (or some other desired configuration). △

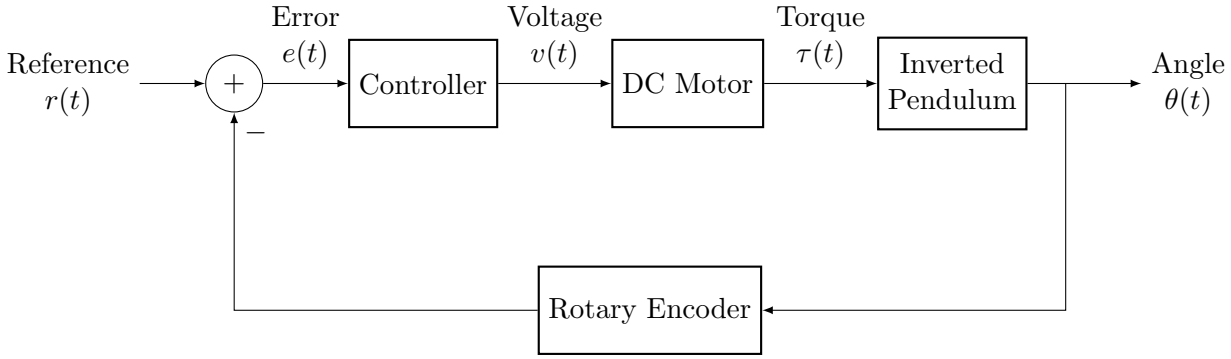

Figure 3: Control system for inverted pendulum. The DC motor is an actuator, the inverted pendulum is the open-loop system, and the rotary encoder is a sensor. The input is a reference signal, which is the desired angle we would like the pendulum to achieve (note that this may be time varying). The angle, $\theta(t)$, is only output. It is measured by the sensor then fed back to a computer which computes the error signal between the reference input and the output angle. The controller then determines which voltage to send to the actuator based on current and past values of $e(t)$.

PROBLEM 1: This problem is intended to help understand what a control system does. The control system of interest is that shown in Figure 3. This control system has the following (fictitious) properties:

- The computer updates every second; it reads the angle, $\theta(t)$, once a second using the encoder.
- The computer only stores the current and previous two values of the angle, θ, as measured by the encoder. In other words, if the computer reads the encoder value every second, then at $t = 5\,\text{s}$, the computer "knows" $\theta(t),\ t \in \{3\,\text{s}, 4\,\text{s}, 5\,\text{s}\}$.
- The controller supplies a constant voltage to the DC motor over each 1 s interval. However, the voltage may change between intervals.
- After a 1 s interval, the angle of the pendulum, $\theta(t)$, changes by one degree per volt applied to the DC motor.
- The error at time t is defined as $e(t) = r(t) - \theta(t)$.
- The controller computes the voltage, $v(t)$, by adding the current error divided by three to the previous error divided by nine with the error before that divided by ten. The controller then floors the voltage to the nearest integer before sending the signal to the DC motor. In other words, at $t = 5\,\text{s}$, the controller sends the following voltage to the DC motor: $v(5) = \lfloor \frac{e(5)}{3} + \frac{e(4)}{9} + \frac{e(3)}{10} \rfloor$. For $t < 0$, the errors are taken as zero.

Using a reference signal of $r(t) = 0^\circ,\ \forall t$, determine how long it takes for the pendulum to swing up from $\theta_0 = 90^\circ$ to within a degree from the reference point.

2 Dynamical Systems

2.1 Motivation

In PROBLEM 1, the controller was defined by the control law,

$$v(t) = \lfloor \frac{e(t)}{3} + \frac{e(t-1)}{9} + \frac{e(t-2)}{10} \rfloor, \quad t \in \mathbb{Z}.$$

A few natural questions should arise after working through the problem. For instance, how would the system dynamics change if the voltage wasn't floored? Is there a better control law that produces the desired swing up within less time? How would one design this control algorithm if it wasn't given? These questions form some of the underlying motivation behind the study of control theory, and although there are many different control strategies (optimal control, model predictive control, robust control, etc.), a common feature is the need for a mathematical model of the open-loop system to be controlled. The study of these models is encompassed within the field of dynamical systems. In other words, before designing and analyzing control algorithms, one must understand how to model and analyze the behavior of dynamical systems.

Dynamical systems are mathematically modeled by differential equations. A differential equation is an equation containing a function (signal) and its derivatives. Examples of differential equations include

$$\dot{x}(t) + x(t) = 0,$$
$$\ddot{e}(t) + e^2(t) = x(t).$$

Differential equations describe how the states of a system change through time. For example, applying Newton's second law to a simple mass-spring-damper system gives

$$\sum F = ma \implies m\ddot{x}(t) + c\dot{x}(t) + kx(t) = f(t),$$

where m is the mass, c is the viscous damping coefficient, k is the spring constant, and $f(t)$ is an external force acting on the mass. Similarly, differential equations can be used to describe the dynamics of an electrical circuit. For example, applying Kirchhoff's voltage law to a series RLC circuit gives

$$\oint v = 0 \implies LC\ddot{v}_C(t) + RC\dot{v}_C(t) + v_C(t) = v_i(t),$$

where R is the resistance, L is the inductance, C is the capacitance, $v_C(t)$ is the voltage across the capacitor, and $v_i(t)$ is the input voltage.

Obviously differential equations are important for modeling system dynamics, so how does one solve differential equations? Guess and check!

2.2 Linear, Time-Invariant Systems

One very special class of systems is called linear, time-invariant systems (LTI systems). LTI systems are described mathematically by ordinary, linear differential equations with constant coefficients. These equations take the form

$$a_n x^{(n)}(t) + a_{n-1}x^{(n-1)}(t) + \cdots + a_2\ddot{x}(t) + a_1\dot{x}(t) + a_0 x(t) = f(t), \quad a_k \in \mathbb{C}, \tag{1}$$

where $x^{(k)}(t)$ denotes the k^{th} derivative of $x(t)$ and a_k is constant through time for $k \in \{1, 2, \ldots, n\}$.

EXAMPLE 7: The following differential equation models a LTI system with an excitation term:

$$3x^{(4)}(t) - 2\ddot{x}(t) + x(t) = 2e^{-t}\sin(5t).$$

$\triangle$

Many real systems can be accurately modeled as linear and time-invariant. LTI systems are very easy to work with, as their mathematical analysis is well established. If a system is nonlinear, it may be approximated by a linear model. This process is called linearization.

EXAMPLE 8: An inverted pendulum is a nonlinear, time-invariant system. It is modeled using Newton's second law as follows:

$$\sum M = I\alpha \implies mgl\sin(\theta(t)) = ml^2\ddot{\theta}(t),$$
$$l\ddot{\theta}(t) - g\sin(\theta(t)) = 0.$$

Note that this differential equation is nonlinear due to the $\sin(\theta(t))$ term. However, for $\theta \approx 0$, $\sin\theta \approx \theta$, and therefore the model can be approximated as linear around the point $\theta = 0$;

$$l\ddot{\theta}(t) - g\theta(t) = 0, \quad \theta \approx 0.$$

$\triangle$

LTI systems have complex exponential eigenfunctions (solutions). These eigenfunctions are found by substituting complex exponentials into the differential equation, then solving the resulting characteristic equation.

EXAMPLE 9: Take the linearized non-inverted pendulum governed by the following model:

$$\ddot{\theta}(t) + \theta(t) = 0.$$

Guess the solution $\theta(t) = ce^{st},\ c, s \in \mathbb{C}$. Plugging this guess back into the differential equation gives

$$cs^2e^{st} + ce^{st} = 0,$$
$$(s^2+1)ce^{st} = 0,$$
$$s^2 + 1 = 0,$$
$$s = \pm\sqrt{-1} := \pm j,$$
$$\implies \theta(t) = c_1e^{jt} + c_2e^{-jt}, \quad c_1, c_2 \in \mathbb{C}.$$

$\triangle$

When there is an external force applied to the system, it shows up in the differential equation as a nonhomogeneous term. The total solution to the linear ODE is the superposition of the homogeneous (unforced) solution with a particular (forced) solution.

EXAMPLE 10: Take $f(t) = e^{-t},\ t \geq 0$, to be the forcing signal applied to a linearized non-inverted pendulum;

$$\ddot{\theta}(t) + \theta(t) = e^{-t}.$$

Guess the solution to be of the form given by $f(t)$; $\theta(t) = ce^{-t},\ c \in \mathbb{C}$. Plugging this guess back into the nonhomogeneous differential equation gives

$$\begin{aligned} ce^{-t} + ce^{-t} &= e^{-t}, \\ 2c &= 1, \\ \implies \theta(t) &= \frac{1}{2}e^{-t}. \end{aligned}$$

Define the total response to be Θ. This total response contains both transient and steady-state components, and is given by

$$\Theta(t) = \underbrace{\frac{1}{2}e^{-t}}_{\text{transient}} + \underbrace{c_1 e^{jt} + c_2 e^{-jt}}_{\text{steady-state}}.$$

Note that the transient component corresponds to the forcing input, whereas the steady-state components correspond to the unforced behavior of the system. This is not always the case. When a system has damping (energy losses), it typically has transient unforced behavior. Furthermore, if the forcing term were to take the form of a persistent signal, such as $f(t) = \sin(t)$, the forced response would become a steady-state component of the total response. In any case, it is useful to analyze the terms of a system's total response by relating them to forced or unforced behavior. The constants c_1 and c_2 can be found by applying initial conditions. For example, if the pendulum started at rest ($\dot{\theta}(0) = 0$) at the position $\theta(0) = \frac{1}{2}$, then the total response gives

$$\Theta(0) = \frac{1}{2}e^0 + c_1 e^0 + c_2 e^0 = \frac{1}{2} \implies c_1 + c_2 = 0.$$

Taking a derivative of the total response and applying the condition $\dot{\theta}(0) = 0$ gives

$$\dot{\Theta}(0) = -\frac{1}{2}e^0 + jc_1 e^0 - jc_2 e^0 = 0 \implies c_1 - c_2 = \frac{1}{j2}.$$

Solving this system of equations yields

$$\begin{aligned} c_1 &= \frac{1}{j4}, \\ c_2 &= -\frac{1}{j4}. \end{aligned}$$

Hence, the dynamics of the non-inverted pendulum have successfully been solved for. △

Two very important signals are defined as follows:

1. *Unit Step Function*:

$$\mu(t) \coloneqq \begin{cases} 0, & t < 0, \\ 1, & t \geq 0. \end{cases} \tag{2}$$

2. *Unit Impulse Function*:

$$\delta(t) \coloneqq \begin{cases} 0, & t \neq 0, \\ \infty, & t = 0. \end{cases} \tag{3}$$

More rigorously, the unit impulse function is defined to have unit area as follows:

$$\delta(t) \coloneqq \lim_{\Delta \to 0} \begin{cases} 0, & t \notin [-\frac{\Delta}{2}, \frac{\Delta}{2}], \\ \frac{1}{\Delta}, & t \in [-\frac{\Delta}{2}, \frac{\Delta}{2}], \end{cases} \implies \int_a^b \delta(t)dt = 1,$$

where $a < 0$ and $b > 0$. Note that the unit step and unit impulse are related; $\frac{d}{dt}\mu(t) = \delta(t)$. These functions are discontinuous at the time origin. In the case that a step or impulse signal is applied as an input to a system, the differential equation becomes more difficult to solve.

Example 11: Take $f(t) = \delta(t)$ to be the forcing signal applied to a linearized non-inverted pendulum;

$$\ddot{\theta}(t) + \theta(t) = \delta(t).$$

This might represent a pendulum which is hit by a hammer to set it in motion. Let the pendulum be initially at rest; $\theta(0^-) = \dot{\theta}(0^-) = 0$. The notation $t = 0^-$ indicates the time just before the impulsive input is applied. The time $t = 0^+$ indicates the time just after the impulse drops back to zero. To solve for $\theta(t)$, the energy imparted to the system by the impulse may be accounted for by solving for new initial conditions, $\theta(0^+)$ and $\dot{\theta}(0^+)$, then solving the equivalent unforced differential equation that governs the system's motion once the impulse returns to zero. These new initial conditions are found by integrating the differential equation across the discontinuity, as follows:

$$\int_{0^-}^{0^+} \ddot{\theta}(t)dt + \int_{0^-}^{0^+} \theta(t)dt = \int_{0^-}^{0^+} \delta(t)dt.$$

In order to continue, it must be introduced that the integral of a unit step function is a unit ramp function; $r(t) \coloneqq \int \mu(t)dt = \begin{cases} 0, & t < 0, \\ t, & t \geq 0. \end{cases}$ Note that $r(0^-) = r(0^+)$, and therefore the unit ramp function is continuous, but its derivatives are not. This kind of continuous function is denoted as $\mathcal{C}^0$. A function whose first derivative is continuous, but all higher order derivatives are not is denoted as $\mathcal{C}^1$. Therefore, since the integral of a unit step function is continuous, but the step is not, then $\mu(t)$ is $\mathcal{C}^{-1}$. Furthermore, since the double integral of a unit impulse is a unit ramp, which is $\mathcal{C}^0$ continuous, the impulse itself is $\mathcal{C}^{-2}$. Since the input to the system of interest is $\mathcal{C}^{-2}$, and derivatives of $\theta(t)$ must be of a lower class of differentiability than $\theta(t)$ itself (unless $\theta(t)$ is $\mathcal{C}^\infty$), then $\ddot{\theta}(t)$ must also be $\mathcal{C}^{-2}$. Therefore, $\dot{\theta}(t)$ is $\mathcal{C}^{-1}$ and $\theta(t)$ is $\mathcal{C}^0$. Since $\theta(t)$ is continuous, $\theta(0^-) = \theta(0^+) = 0$. Furthermore, $\int \theta(t)dt$ is $\mathcal{C}^1$, which means that the antiderivative of $\theta(t)$ is continuous across the origin. This allows for the completion of integrating the differential equation across the discontinuity:

$$\dot{\theta}(0^+) - \dot{\theta}(0^-) + 0 = 1 \implies \dot{\theta}(0^+) = 1.$$

Hence, the differential equation reduces to

$$\ddot{\theta}(t) + \theta(t) = 0, \quad t > 0,$$

where $\theta(0^+) = 0$ and $\dot{\theta}(0^+) = 1$. Solving this equation using complex exponential eigenfunctions (as shown in earlier examples) gives

$$\theta(t) = \sin(t), \quad t > 0.$$

$\triangle$

Clearly, analyzing system dynamics in the presence of discontinuous signals is cumbersome and quite complicated. Fortunately, the Laplace transform allows for simpler solutions by transforming LTI ODEs into algebraic equations.

2.3 Laplace Transform

The Laplace transform utilizes the complex exponential eigenfunction as a kernel in an integral transform;

$$F(s) = \mathcal{L}\{f(t)\} \coloneqq \int_{0^-}^{\infty} f(t)e^{-st}dt. \tag{4}$$

As seen by the definition, the Laplace transform eliminates the time variable and introduces the complex variable $s \in \mathbb{C}$. This operation may be thought of as transforming a function from the time domain into the complex frequency (Laplace) domain.

EXAMPLE 12: Let $f(t) = e^{-at}$. Then the Laplace transform is

$$F(s) = \int_{0^-}^{\infty} e^{-at}e^{-st}dt = \int_{0^-}^{\infty} e^{-(s+a)t}dt = -\frac{1}{s+a}\left(e^{-\infty} - e^{0}\right) = \frac{1}{s+a}.$$

△

The Laplace transforms of common functions can be found in tables online or in texts on differential equations or dynamical systems. Perhaps the most important Laplace transform is that of derivatives:

$$\begin{aligned}
\mathcal{L}\{\dot{f}(t)\} &= s\mathcal{L}\{f(t)\} - f(0^-),\\
\mathcal{L}\{\ddot{f}(t)\} &= s^2\mathcal{L}\{f(t)\} - sf(0^-) - \dot{f}(0^-),\\
&\vdots\\
\mathcal{L}\{f^{(n)}(t)\} &= s^n\mathcal{L}\{f(t)\} - s^{n-1}f(0^-) - s^{n-2}\dot{f}(0^-) - \cdots - sf^{(n-2)}(0^-) - f^{(n-1)}(0^-).
\end{aligned} \tag{5}$$

This transform can be used to turn LTI differential equations into algebraic equations.

EXAMPLE 13: Returning to the linearized non-inverted pendulum problem with a unit impulse input, Laplace transforms may be employed to solve the problem in a much easier manner. Looking into a Laplace transform table, it is found that $\mathcal{L}\{\delta(t)\} = 1$. Therefore, applying the Laplace transform to the differential equation and using the Laplace transform's linearity,

$$\begin{gathered}
\mathcal{L}\{\ddot{\theta}(t)\} + \mathcal{L}\{\theta(t)\} = \mathcal{L}\{\delta(t)\},\\
s^2\Theta(s) - s\theta(0^-) - \dot{\theta}(0^-) + \Theta(s) = 1,\\
s^2\Theta(s) - s\cdot 0 - 0 + \Theta(s) = 1,\\
(s^2+1)\Theta(s) = 1,\\
\Theta(s) = \frac{1}{s^2+1},\\
\implies \theta(t) = \mathcal{L}^{-1}\left\{\frac{1}{s^2+1}\right\}.
\end{gathered}$$

Here, $\mathcal{L}^{-1}\{\cdot\}$ represents the inverse Laplace transform, which transforms a function from the frequency domain back into the time domain. Referencing a Laplace table once more, it is seen that $\mathcal{L}\{\sin(at)\} = \frac{a}{s^2+a^2}$. In this case, $a = 1$, and therefore

$$\theta(t) = \sin(t), \quad t > 0.$$

Note that this is the same result found using direct integration across the discontinuity. △

It is useful to transform the differential equations governing a system's dynamics into the Laplace domain, since it allows for the easy characterization of input-output relations.

EXAMPLE 14: Let $f(t)$ represent an arbitrary forcing signal which acts as the input to a linearized, non-inverted pendulum. Taking the initial conditions to be zero, the differential equation is transformed as follows:

$$\ddot{\theta}(t) + \theta(t) = f(t) \implies s^2\Theta(s) + \Theta(s) = F(s) \implies \Theta(s) = \frac{1}{s^2+1}F(s).$$

Denote $H(s) = \frac{1}{s^2+1}$. Therefore, $\Theta(s) = H(s)F(s) \implies \theta(t) = \mathcal{L}^{-1}\{H(s)F(s)\}$. In other words, the output of the system, $\Theta(s)$, is simply the product of $H(s)$ with the input signal, $F(s)$. The function $H(s)$ is called the transfer function of the system. Note that the transfer function is completely characterized by the properties of the system itself, and is independent of the input or output signals. △

Every LTI system has a transfer function, and the output of the system (in the Laplace domain) is equal to the transfer function multiplied by the input signal. In other words, a LTI system's transfer function completely defines the input-output relationship of the system.

EXAMPLE 15: A mass-spring-damper system is governed by the following ODE:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = f(t).$$

The transfer function of the system is found using the Laplace transform with zero initial conditions.

$$ms^2X(s) + csX(s) + kX(s) = F(s) \implies X(s) = \frac{1}{ms^2+cs+k}F(s).$$

Therefore, this system's transfer function is

$$H(s) = \frac{1}{ms^2+cs+k}.$$

Note that the transfer function is entirely defined in terms of the system's physical parameters, m, c, and k. △

2.4 Unit Impulse Response

Recall that $\mathcal{L}\{\delta(t)\} = 1$. Therefore, if the input to a LTI system is a (possibly scaled) unit impulse, $f(t) = \delta(t)$, then the output of the system (in the Laplace domain) is simply equal to the transfer function of the system;

$$X(s) = H(s)F(s) = H(s)\mathcal{L}\{\delta(t)\} = H(s) \implies x(t) = h(t).$$

The signal $h(t)$ is termed the "unit impulse response" of the system. In other words, the transfer function of a system equals the Laplace transform of the system's unit impulse response. This implication is extremely important to the study of dynamical systems, and it may be interpreted in a variety of ways. First, this statement shows that if the transfer function of a system is known, then the system's response to a unit impulse input may be found by simply transforming the transfer function to the time domain. On the other hand, if an experiment is performed in

which an impulse is input to a system and the output signal, $x(t) = h(t)$ is measured, then the system's transfer function may be found by transforming the impulse response into the Laplace domain. Since the transfer function can be used to find the system's response to any arbitrary input, an impulse response experiment suffices to completely characterize the system's dynamics in the presence of any arbitrary input signal.

The process of creating a mathematical model of a system using experiments or simulations is called system identification. Obviously, imparting an impulse to a system and recording the impulse response is one method of system identification, since it gives the transfer function of the system. However, many times it is not possible to impart an impulsive input that closely resembles the idealistic function, $\delta(t)$. It is often easier to generate step inputs in a physical setting. Recall, however, that the unit step and unit impulse are related: $\delta(t) = \frac{d}{dt}\mu(t)$. Using the Laplace transform gives

$$1 = \mathcal{L}\{\delta(t)\} = \mathcal{L}\left\{\frac{d}{dt}\mu(t)\right\} = s\mathcal{L}\{\mu(t)\} \implies \mathcal{L}\{\mu(t)\} = \frac{1}{s}.$$

Hence, if the input to a system is $f(t) = \mu(t)$, the output in the Laplace domain is

$$X(s) = H(s)\mathcal{L}\{\mu(t)\} \implies X(s) = \frac{1}{s}H(s).$$

In other words, applying a step input to a system, then recording its response, $x(t)$, also works as a method of system identification, since the transfer function $H(s)$ may be found by transforming the step response to the Laplace domain then multiplying by s.

2.5 State-Space Representations

Instead of working in the Laplace domain, LTI dynamical systems may be analyzed in the time domain by converting the differential equations of motion into a system of first-order equations. This results in a first-order matrix-vector differential equation.

EXAMPLE 16: A mass-spring-damper system is governed by the following ODE:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = f(t).$$

Let $\mathbf{z}(t) = \begin{bmatrix} z_1(t) & z_2(t) \end{bmatrix}^T$, where $z_1(t) = x(t)$ and $z_2(t) = \dot{x}(t)$. Then $\dot{z}_1(t) = \dot{x}(t) = z_2(t)$ and $\dot{z}_2(t) = \ddot{x}(t) = -\frac{k}{m}x(t) - \frac{c}{m}\dot{x}(t) + \frac{1}{m}f(t) = -\frac{k}{m}z_1(t) - \frac{c}{m}z_2(t) + \frac{1}{m}f(t)$. Therefore,

$$\dot{\mathbf{z}}(t) = \begin{bmatrix} z_2(t) \\ -\frac{k}{m}z_1(t) - \frac{c}{m}z_2(t) + \frac{1}{m}f(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} f(t).$$

Suppose the output signal of interest is taken to be the mass position; $y(t) = x(t)$. Then

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix}.$$

The state-space description of the system is therefore governed by the following two equations:

$$\begin{aligned} \dot{\mathbf{z}}(t) &= \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \mathbf{z}(t) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} f(t), \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{z}(t). \end{aligned}$$

△

A LTI system described by an n^{th}-order differential equation may always be converted into a system of n first-order differential equations, and therefore a state-space representation of the system may always be found. These state-space equations take the form

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t), \tag{6}$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t). \tag{7}$$

Here, the vectors $\mathbf{u}(t)$, $\mathbf{x}(t)$, and $\mathbf{y}(t)$ represent the input signal, state vector, and output signal, respectively. The matrices A, B, C, and D are termed the state matrix, input matrix, output matrix, and feedthrough matrix, respectively. The first equation is called the state equation, and the second equation is termed the output equation. Typically the feedthrough matrix is $D = 0$, unless the input $\mathbf{u}(t)$ directly influences the output $\mathbf{y}(t)$. Taking the Laplace transform of the state equation gives

$$\begin{aligned} s\mathbf{X}(s) &= A\mathbf{X}(s) + B\mathbf{U}(s), \\ (sI - A)\mathbf{X}(s) &= B\mathbf{U}(s), \\ \mathbf{X}(s) &= (sI - A)^{-1}B\mathbf{U}(s). \end{aligned}$$

Taking the Laplace transform of the output equation and substituting $\mathbf{X}(s)$ yields

$$\begin{aligned} \mathbf{Y}(s) &= C\mathbf{X}(s) + D\mathbf{U}(s), \\ &= C(sI - A)^{-1}B\mathbf{U}(s) + D\mathbf{U}(s), \\ \mathbf{Y}(s) &= (C(sI - A)^{-1}B + D)\mathbf{U}(s). \end{aligned}$$

Hence, the transfer function of the system is

$$H(s) = C(sI - A)^{-1}B + D. \tag{8}$$

Hence, if a system's state-space model is known, then its unique transfer function may be determined by the four coefficient matrices of the state-space equations.

Example 17: Taking $m = c = k = 1$, a mass-spring-damper system has the following equation of motion:

$$\ddot{x}(t) + \dot{x}(t) + x(t) = f(t).$$

Note that, by direct Laplace transform, the transfer function is found to be

$$s^2X(s) + sX(s) + X(s) = F(s) \implies H(s) = \frac{X(s)}{F(s)} = \frac{1}{s^2 + s + 1}.$$

The state-space description of the system, taking the states to be $z_1(t) = x(t)$ and $z_2(t) = \dot{x}(t)$, becomes

$$\begin{aligned} \mathbf{z}(t) &= \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \mathbf{z}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} f(t), \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{z}(t). \end{aligned}$$

Note that the output is taken to be $y(t) = x(t)$. Using the state-space transfer function equation gives

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

△

2.6 Poles and Stability

Since LTI systems have complex exponential eigenfunctions (solutions), their natural responses take the form

$$x(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} + \cdots + c_n e^{\lambda_n t}, \quad \lambda_k \in \mathbb{C}.$$

Each λ_k is determined by solving the system's characteristic equation. These roots of the characteristic polynomial are called the poles of the system. Note that some of the terms might be multiplied by powers of t in the case that the system's characteristic equation has repeated roots. Regardless, the exponential factor of each term dominates the decay or growth of the term. The complex poles take the form $\lambda_k = \sigma_k + j\omega_k,\ \sigma_k, \omega_k \in \mathbb{R}$. Therefore,

$$e^{\lambda_k t} = e^{(\sigma_k + j\omega_k)t} = e^{\sigma_k t} e^{j\omega_k t} = e^{\sigma_k t}(\cos(\omega_k t) + j\sin(\omega_k t)).$$

This result demonstrates a few important features of the poles of a LTI system. First, the system's natural response will have oscillatory dynamics when $\omega_k \neq 0$. The frequency of oscillation is given by ω_k. In other words, if the system has any poles with a nonzero imaginary part, the system displays oscillatory behavior. Second, the modulus of the k^{th} term is

$$\begin{aligned}\left|e^{\lambda_k t}\right| = \left|e^{\sigma_k t}(\cos(\omega_k t) + j\sin(\omega_k t))\right| = \left|e^{\sigma_k t}\right| \left|\cos(\omega_k t) + j\sin(\omega_k t)\right| = e^{\sigma_k t}\sqrt{\cos^2(\omega_k t) + \sin^2(\omega_k t)} \\ = e^{\sigma_k t}.\end{aligned}$$

Hence, the growth or decay of $e^{\lambda_k t}$ is dependent on the real part of λ_k; $\sigma_k < 0$ yields a decaying term, whereas $\sigma_k > 0$ yields a growing term. When $\sigma_k = 0$, the term might be growing (in the case of repeated roots) or might be persistently oscillating (in the case of unique, purely imaginary roots). A system is stable when all of its poles yield bounded behavior. When at least one of the system's poles has unbounded behavior (such as $\sigma_k > 0$), the system is called unstable.

Example 18: Take System 1 to be the non-inverted pendulum defined by the following equation of motion:

$$\ddot{\theta}(t) + \theta(t) = f(t).$$

The poles of the system may be found in a variety of ways. First, substitute the eigenfunction $\theta(t) = ce^{\lambda t}$ into the homogeneous differential equation:

$$\lambda^2 ce^{\lambda t} + ce^{\lambda t} = (\lambda^2 + 1)ce^{\lambda t} = 0 \implies (\lambda_1, \lambda_2) = (j, -j).$$

The poles of the system are therefore located at $\pm j$ in the complex plane. This is depicted graphically using a pole-zero plot, shown in Figure 4. Since the real part of these poles is zero and the poles are unique, there is no growth or decay; $\theta(t) = c_1 e^{jt} + c_2 e^{-jt},\ c_1, c_2 \in \mathbb{C}$. Since the natural response of this system is bounded, it is a stable system. More specifically, since the response is bounded but does not decay to zero, the system is termed marginally stable. The poles may also be found from the transfer function of the system:

$$\mathcal{L}\{\ddot{\theta}(t)\} + \mathcal{L}\{\theta(t)\} = \mathcal{L}\{f(t)\} \implies s^2\Theta(s) + \Theta(s) = F(s) \implies H(s) = \frac{\Theta(s)}{F(s)} = \frac{1}{s^2 + 1}.$$

The poles are roots of the denominator of the transfer function (this fact holds for all LTI systems). Therefore, $s^2 + 1 = 0 \implies s = \pm j$. A third method for finding the poles of a system uses the

state-space description. Letting $z_1(t) = \theta(t)$ and $z_2(t) = \dot{\theta}(t)$, the state equation of System 1 becomes

$$\dot{\mathbf{z}}(t) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \mathbf{z} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} f(t).$$

Since the poles of a system may be found by setting the denominator of the transfer function to zero, the poles may be found using the state-space description by noting that $H(s) = C(sI - A)^{-1}B + D$ has singularities at the points s which gives $\det(sI - A) = 0$. In other words, *the poles of the system are the eigenvalues of the A matrix.* The poles of System 1 are therefore found as follows:

$$\det(sI - A) = \det \begin{bmatrix} s & -1 \\ 1 & s \end{bmatrix} = s^2 + 1 = 0 \implies s = \pm j.$$

Regardless of the three methods for finding system poles presented here, the poles are the same; the poles of a system are intrinsic properties of the system itself, and are not dependent on the manner in which you mathematically analyze the system. In fact, using a similarity transformation to change the coordinates of the state-space description does not change the poles of the system!

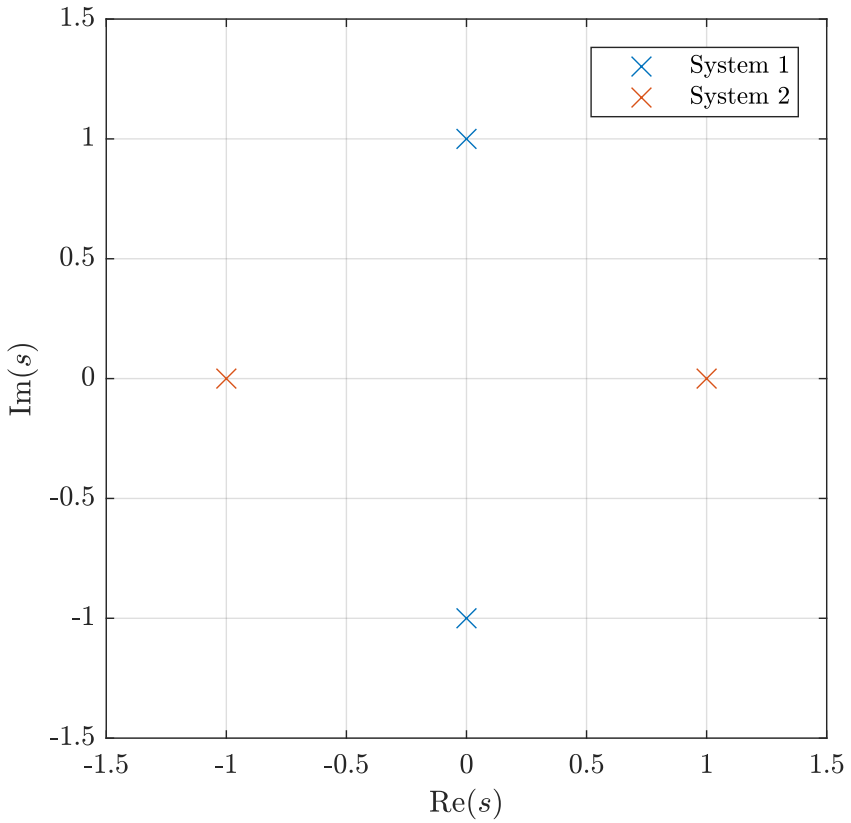

Figure 4: Pole-zero map of Systems 1 and 2. Note that System 1 has purely imaginary poles and therefore exhibits an undamped oscillatory natural motion. System 2 has one pole with a positive real part, which indicates the system's instability.

Let System 2 be the inverted pendulum defined by the following equation of motion:

$$\ddot{\theta}(t) - \theta(t) = f(t).$$

Using the Laplace transform to find the transfer function of the system gives

$$s^2\Theta(s) - \Theta(s) = F(s) \implies H(s) = \frac{\Theta(s)}{F(s)} = \frac{1}{s^2 - 1}.$$

The poles of System 2 are therefore found when the denominator of $H(s)$ equals zero;

$$s^2 - 1 = 0 \implies s = \pm 1.$$

In other words, the system's natural response takes the form $\theta(t) = c_1 e^t + c_2 e^{-t},\ c_1, c_2 \in \mathbb{C}$. Although one term does decay to zero, the other term diverges. Therefore, the system is unstable. This result matches intuition, since one would expect an inverted pendulum to naturally fall from its unstable equilibrium point in the upright position. The poles of System 2 are shown in Figure 4. △

Since the decay rate of a stable system is dictated by the real part of its poles, poles further to the left in the complex plane exhibit faster decay than poles closer to the origin. That is, if a system has two poles, $\{\lambda_1, \lambda_2\}$, such that

$$\mathrm{Re}(\lambda_1) \ll \mathrm{Re}(\lambda_2),$$

then the term corresponding to λ_2 decays faster than that corresponding to λ_1. When a system has one pole that is significantly slower than the others, that pole is said to be dominant. The slow pole dominates a system's natural response because the faster poles decay rapidly, leaving only the dynamics of the slow pole once the components corresponding to the fast poles become negligibly small.

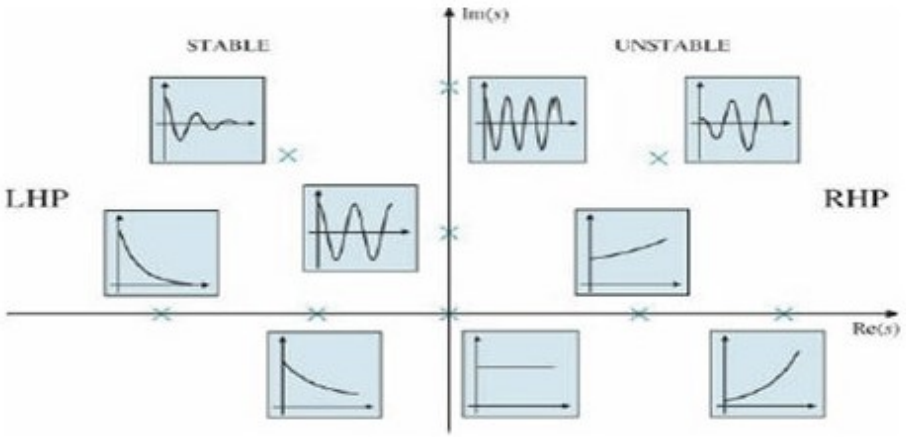

Figure 5: Pole locations and their corresponding natural responses. Poles in the right-half of the complex plane yields unstable behavior. Poles with a nonzero imaginary part yields oscillatory responses. Poles on the imaginary axis give undamped oscillation, whereas poles on the real axis give exponential dynamics without oscillations. The frequency of oscillation increases as the imaginary part of a pole increases in magnitude. The decay rate of a stable pole increases as the pole moves left in the complex plane.

2.7 Frequency Response

This section introduces the concept of frequency response of a LTI system. The frequency response of a system describes how the system responds to sinusoidal input of different frequencies. Let the

input to a LTI system be a pure sinusoid; $f(t) = Ae^{j(\omega t+\phi)} = A(\cos(\omega t+\phi) + j\sin(\omega t+\phi))$ where $A, \omega, \phi \in \mathbb{R}$. This input signal might be generated physically by applying to a mechanism a force that fluctuates in amplitude throughout time, or by applying an AC voltage across a linear circuit. In any case, this type of input is characterized by three values: the amplitude, A, the frequency, ω, and the phase shift, ϕ. When a LTI system is stable, the steady-state response to a purely sinusoidal input is a sinusoid of the same frequency, but possibly scaled in amplitude and shifted in phase. This can be shown as follows. Assume the system's response is $x(t) = Be^{j(\omega t+\gamma)},\ B, \gamma \in \mathbb{R}$. Then the differential equation of a LTI system yields

$$\begin{aligned}
\sum_{k=0}^{n} a_k x^{(k)}(t) &= f(t), \\
\sum_{k=0}^{n} a_k (j\omega)^k B e^{j(\omega t+\gamma)} &= Ae^{j(\omega t+\phi)}, \\
\left(\sum_{k=0}^{n} a_k (j\omega)^k\right) B e^{j\omega t} e^{j\gamma} &= Ae^{j\omega t} e^{j\phi}, \\
Be^{j\gamma} &= \underbrace{\frac{1}{\sum_{k=0}^{n} a_k (j\omega)^k}}_{=H(s)|_{s=j\omega}} Ae^{j\phi}, \\
Be^{j\gamma} = H(j\omega)Ae^{j\phi} &= |H(j\omega)|\, e^{j\angle H(j\omega)} Ae^{j\phi}, \\
\implies B = |H(j\omega)|\, A, &\quad \gamma = \angle H(j\omega) + \phi.
\end{aligned}$$

In other words, the system's response due to a sinusoidal input is scaled by the magnitude of the transfer function evaluated at the frequency of the sinusoid and is phase shifted by the angle of the transfer function evaluated at the frequency of the sinusoid;

$$x(t) = (|H(j\omega)|\, A)\, e^{j(\omega t+\angle H(j\omega)+\phi)}. \tag{9}$$

The transfer function, when evaluated at purely sinusoidal points, $s = j\omega$, is called the frequency response function of the system. Evaluating $H(j\omega)$ at a certain frequency yields a complex number with an amplitude and phase. The modulus of this complex number scales the sinusoidal input to give the amplitude of the output sinusoid. The angle of this complex number shifts the sinusoidal input to give the phase of the output sinusoid. When $H(j\omega)$ is plotted across a frequency interval, the amplitude and phase plots generate what are called Bode plots.

Example 19: Take the linearized non-inverted pendulum governed by the following model:

$$\ddot{\theta}(t) + \theta(t) = f(t).$$

The transfer function of this system is

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

The frequency response function is therefore

$$H(j\omega) = \frac{1}{(j\omega)^2+1} = \frac{1}{1-\omega^2}.$$

The modulus of this frequency response function is

$$|H(j\omega)| = \left|\frac{1}{1-\omega^2}\right| = \frac{1}{|1-\omega^2|},$$

and the angle is

$$\angle H(j\omega) = \angle\left(\frac{1}{1-\omega^2}\right) = 0 - \angle(1-\omega^2) = \begin{cases} 0^\circ, & \omega < 1, \\ -180^\circ, & \omega > 1. \end{cases}$$

These functions are shown in the Bode plots of Figure 6. Note that, since $s = \pm j$ are the poles of the system, the denominator of $H(s)$ equals zero at $s = \pm j$. In other words, a sinusoidal input of frequency $\omega = 1\,\text{rad/s}$ causes the modulus of the transfer function to diverge to infinity. This phenomenon is called resonance. Resonance may be intuitively understood by thinking of a playground swing. When a friend pushes the person in the swing at just the right frequency, the height of each swing cycle is increased until either friction overcomes the energy input, the swing goes slack, or the person in the swing falls out and breaks their neck. If damping were added to the system, the resonant peak in the Bode magnitude plot would not spike all the way to infinity, but would spike to some finite value. If damping was further increased, the resonant peak may disappear entirely.

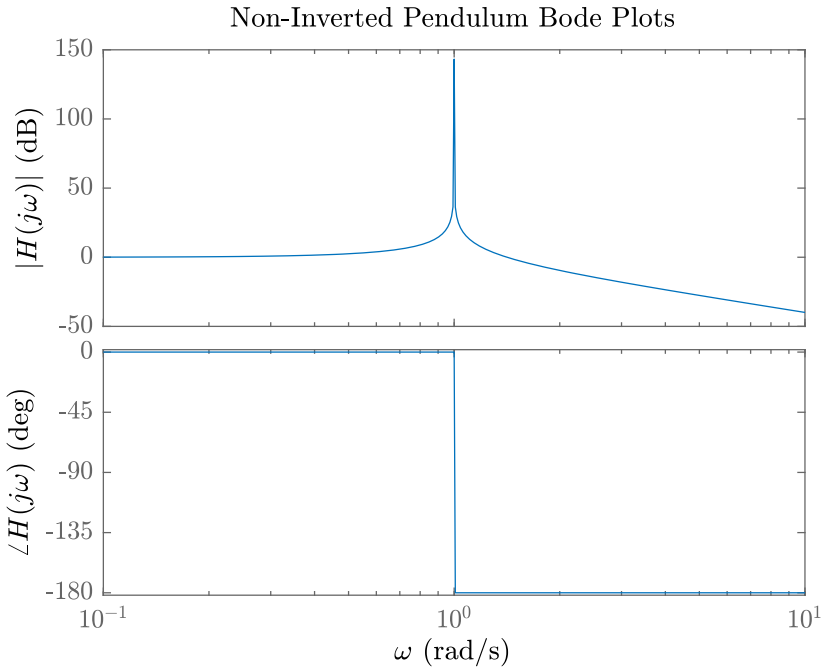

Figure 6: Bode plots of the non-inverted pendulum system without damping. Resonance occurs at $\omega = 1\,\text{rad/s}$, as evidenced by the peak in the Bode magnitude plot. Note that this system does not alter input sinusoids of low frequencies. However, for high-frequency sinusoids, the system attenuates (scales down) the input and yields an output signal that is of lower amplitude and shifted by -180°. As the frequency approaches infinity, the amplitude of the output signal, $\theta(t)$, approaches zero.

△

PROBLEM 2: This problem is intended to cover the mathematical basics of LTI dynamical systems. Assume a step response experiment has been performed on the system of interest. The input signal was

$$u(t) = \mu(t) = \begin{cases} 0, & t < 0, \\ 1, & t \geq 0, \end{cases}$$

The output of the system was recorded to be

$$y(t) = \frac{100}{101}(1 - e^{-t}(\cos(10t) + \sin(10t))), \quad t \geq 0.$$

a. Find the response of the system if a unit impulse was the input. Hint: Recall that $\delta(t) = \frac{d}{dt}\mu(t)$. For LTI systems, the impulse response and step response are related in the same manner; $h(t) = \frac{d}{dt}y(t)$.

b. Find the transfer function of the system. Hint: Use a Laplace transform table.

c. Find the poles of the system.

d. Is this system stable, marginally stable, or unstable?

e. Find the differential equation that governs this system. Hint: Recall that, without initial conditions, $\mathcal{L}\{\dot{x}(t)\} = sX(s)$. Therefore, use the transfer function to convert powers of s into derivatives to generate an ODE. The ODE should contain the output, $y(t)$, the input $u(t)$, and their derivatives.

f. Convert the differential equation into state-space form.

g. Find the eigenvalues of the state matrix from the state-space equations.

h. Plot the frequency response function (generate Bode plots). If MATLAB is readily available, the code in Listing 1 is useful.

Listing 1: MATLAB code for generating Bode plots.

```
s = tf('s');
H = 1/(s^2+1); % put your transfer function here
bode(H);
```

i. Does this system display resonance? If so, at what frequency does resonance occur?

j. If the input to the system is $u(t) = 5\sin(10t)$, what is the steady-state output? You may either solve the differential equation, use the frequency response function, or use a Bode plot to find an approximate solution.

3 Feedback Control

3.1 Motivation

The objective of control theory is to manipulate the natural behavior of a dynamical system in order to make it behave in a desirable fashion. For example, an inverted pendulum naturally falls over. By creating a control system, a sensor and actuator may be used to control the pendulum's angle and stabilize the system in the upright position. Mathematically, this is achieved by moving the poles of the system from the right-half of the complex plane to the left-half of the complex plane. This is one way of thinking of controller design: a controller is designed to move the poles of a system to a desirable location within the complex plane. This may include moving the poles to the left-half plane in order to stabilize a system, or moving stable poles further left to speed up the dynamic response of a system, or possibly moving complex poles closer to the real axis to lower the frequency of oscillation.

3.2 Time Domain Specifications

One way to design a controller is to specify certain desired characteristics of a system's step response. Some of these characteristics include:

- *Overshoot*: This is the percentage by which the response overshoots its final, steady-state value. It is typically desirable to minimize this value.

  EXAMPLE 20: If a system's steady-state response to a step is $\lim_{t\to\infty} y(t) = 5$, and its peak value is $\max_t y(t) = 5.5$, then the overshoot is $M_p = \frac{5.5-5}{5} = 0.10 = 10\%$. △

- *Rise Time*: The time it takes for the system's response to reach $\sim 90\%$ of its steady-state value. It is typically desirable to minimize this value.

- *Settling Time*: The time it takes for the system's response to settle within $\sim 2\%$ of its steady-state value. It is typically desirable to minimize this value.

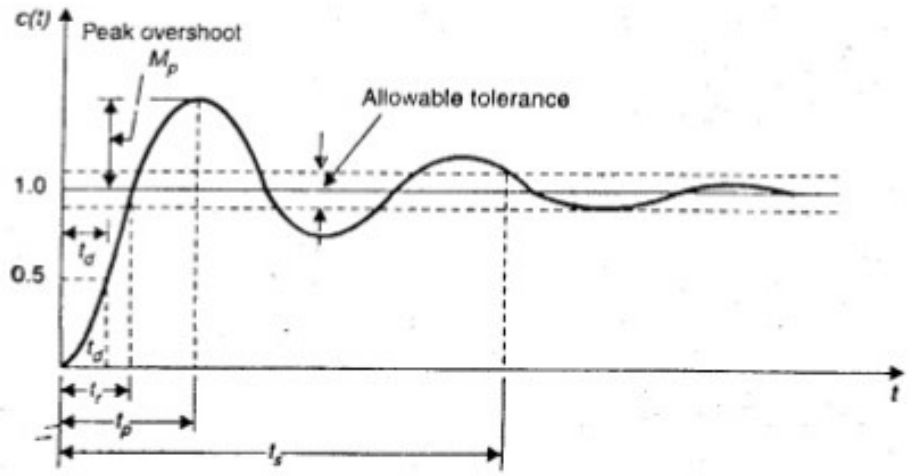

Figure 7: Step response characteristics. Specifying these time domain parameters translates to constraints in a controller design problem.

These characteristics have well-known analytical formulas for second-order systems. First-order characteristics are also well established. However, MATLAB has built-in functions to deal with these specifications and how they relate to controller design. Therefore, the analytical relations between these characteristics and pole location will not be discussed here.

3.3 Introduction to Controller Design

In this section, controller design will take place in the Laplace domain; signals and systems are represented as function of the complex variable s. Let $H(s)$ be the transfer function for some system of interest. The open-loop block diagram of the system is shown in Figure 8. Now suppose a controller is used to send the input signal to the open-loop system, and the output of the system is fed back and compared to some desired reference signal. This closed-loop control system is shown in Figure 9. The goal of controller design is to choose a controller, $C(s)$, such that the output signal converges to the reference signal; $Y(s) \to R(s)$. Furthermore, $C(s)$ should be designed such that this convergence occurs quickly and without poor behavior such as unnecessary oscillations in $Y(s)$.

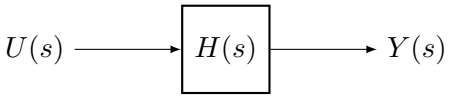

Figure 8: Open-loop system.

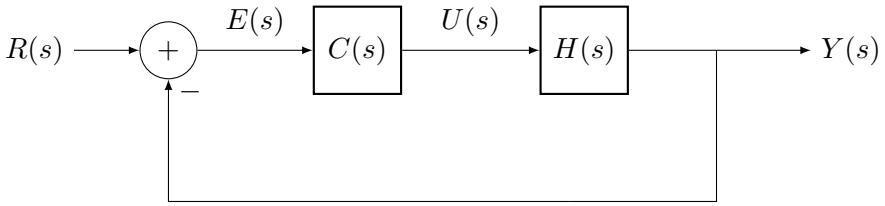

Figure 9: Unity feedback system. The open-loop system to be controlled is denoted as $H(s)$, and the controller is denoted $C(s)$.

EXAMPLE 21: Let the system of interest be defined by

$$H(s) = \frac{1}{s-1}.$$

This system has a pole at $\lambda = 1 \implies \text{Re}(\lambda) = 1 > 0$, and therefore the system is unstable. The goal of this problem is therefore to stabilize this system.

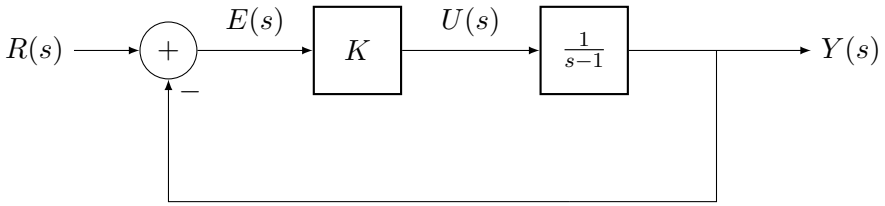

Figure 10: Closed-loop control system. The controller is taken to be a proportional controller; $C(s) = K$.

Suppose the control system of Figure 10 is created. For now, take the controller to be a "proportional controller." Controllers of this form simply scale the signal entering the controller; $C(s) = K \in \mathbb{R}$. Suppose now that the control system is treated as its own, self-contained system which has a unique transfer function, denoted $\tilde{H}(s)$. In other words, $\tilde{H}(s)$ defines the relationship between the input to the control system, $R(s)$, and the output of the control system, $Y(s)$. This transfer function is called the closed-loop transfer function. The poles of the closed-loop transfer function contain the stability information of the overall closed-loop control system. Therefore, by manipulating the controller constant K such that the poles of $\tilde{H}(s)$ are in the left-half of the complex plane, the problem is solved. The closed-loop transfer function may be found by analyzing the block diagram. Using the generalized unity feedback block diagram of Figure 9, the transfer function is determined as follows:

$$\begin{cases} Y(s) = H(s)U(s), \\ U(s) = C(s)E(s), \\ E(s) = R(s) - Y(s), \end{cases} \implies Y(s) = H(s)C(s)(R(s) - Y(s)),$$

$$(1 + H(s)C(s))Y(s) = H(s)C(s)R(s),$$

$$Y(s) = \frac{H(s)C(s)}{1 + H(s)C(s)}R(s),$$

$$= \frac{\frac{1}{s-1}K}{1 + \frac{1}{s-1}K}R(s),$$

$$Y(s) = \frac{K}{s - 1 + K}R(s).$$

Hence, the closed-loop transfer function is

$$\tilde{H}(s) = \frac{K}{s - 1 + K},$$

and therefore the closed-loop pole is at $\lambda - 1 + K = 0 \implies \lambda = 1 - K$. Since $\mathrm{Re}(\lambda) < 0$ yields a stable system, $K > 1$ can be chosen to stabilize the system. Figure 11 shows the step response of the system in the case that the controller is taken as $K = 0.5$. As expected, the system is unstable since $K < 1$. This is evidenced by the unbounded behavior of the system's response. On the other hand, Figure 12 shows the step responses for the cases $K_1 = 2$ and $K_2 = 4$. As expected, the system is stable for both cases, since $K_k > 1,\ k \in \{1, 2\}$. This is evidenced by the exponentially decaying responses. Furthermore, note that the controller $K_2 = 4$ yields a faster decay rate than $K_1 = 2$. This can be explained using the inverse Laplace transform:

$$\tilde{H}(s) = \frac{Y(s)}{R(s)} = \frac{K}{s - 1 + K},$$

$$\implies Y(s) = \frac{K}{s - 1 + K}R(s)$$

$$= \frac{K}{s - 1 + K}\mathcal{L}\{\mu(t)\}$$

$$= \frac{K}{s - 1 + K}\left(\frac{1}{s}\right).$$

Let

$$\frac{K}{s(s - 1 + K)} = \frac{A}{s} + \frac{B}{s - 1 + K}, \quad A, B \in \mathbb{C}.$$

Then

$$K = As + A(-1 + K) + Bs.$$

Equation polynomial coefficients of s gives $K = A(-1 + K) \implies A = \frac{K}{K-1}$ and $A + B = 0 \implies B = -\frac{K}{K-1}$. Therefore, $Y(s)$ can be rewritten as

$$Y(s) = \frac{K}{K-1}\left(\frac{1}{s} - \frac{1}{s-1+K}\right).$$

Using a Laplace transform table, $Y(s)$ may be transformed back into the time domain:

$$y(t) = \mathcal{L}^{-1}\{Y(s)\} = \frac{K}{K-1}\left(1 - e^{-(K-1)t}\right), \quad t \geq 0. \tag{10}$$

Two important pieces of information can be extracted from this equation. First, the exponential term decays to zero for $K > 1$, which matches the result found by analyzing the closed-loop pole locations in terms of controller constant, K. Second, the decay rate of the exponential term is faster for increasing values of K, hence the faster system response for $K_2 = 4$ when compared to $K_1 = 2$. Recall that stable poles in the complex plane generate faster decay rates as the pole location moves left. This general rule could have also been used to characterize K_2 as the "faster" controller; $K_1 = 2 \implies \lambda_1 = 1 - K_1 = 1 - 2 = -1$, whereas $K_2 = 4 \implies \lambda_2 = 1 - K_2 = 1 - 4 = -3$.

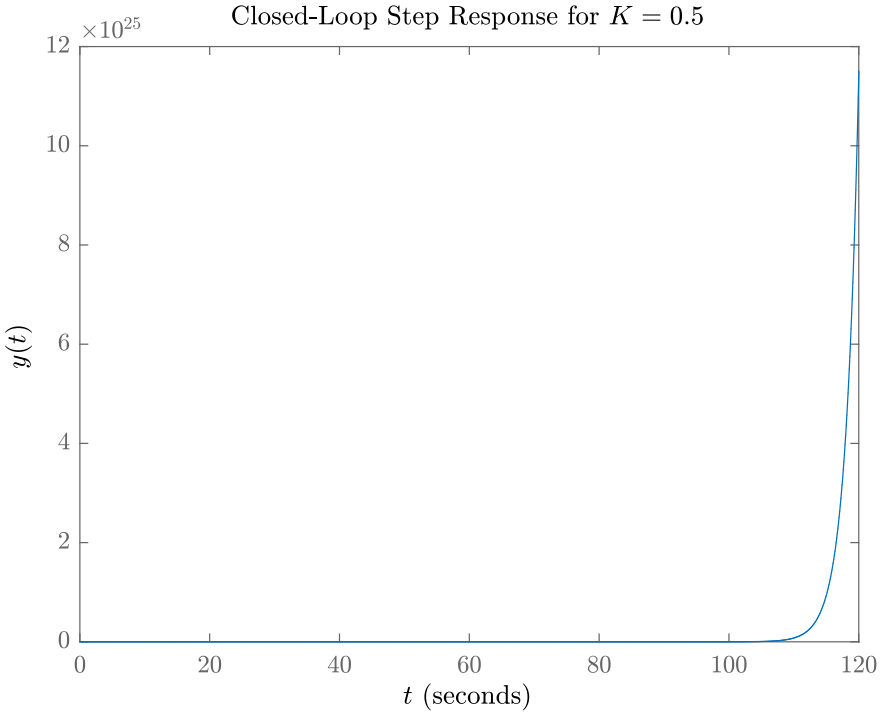

Figure 11: The divergent step response demonstrates the closed-loop system's instability for the controller gain $K = 0.5$.

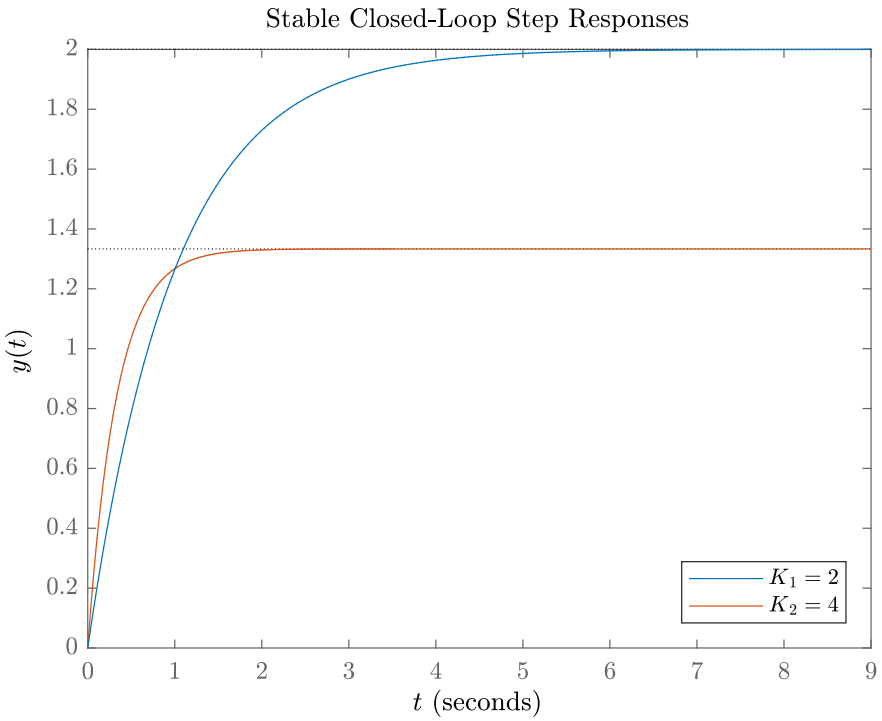

Figure 12: For $K_1 = 2$ and $K_2 = 4$, the system's response converges toward a constant steady-state value, demonstrating the closed-loop stability for these controller gains.

There are two final questions to study before adequately exhausting this problem. First, why does the controller $C(s) = K_1 = 2$ yield a higher steady-state response than $C(s) = K_2 = 4$? This may be answered in a variety of ways. The first method of answering this question is to use the time domain step response formula (10) derived through the inverse Laplace transform. Simply taking the limit as $t \to \infty$ gives

$$y_{ss} = \lim_{t\to\infty} y(t) = \lim_{t\to\infty} \frac{K}{K-1}\left(1 - e^{-(K-1)t}\right) = \frac{K}{K-1}, \quad K > 1.$$

Thus, the steady-state response value is dependent on the controller gain K. For $K_1 = 2$, the steady-state value is $y_{ss} = \frac{2}{2-1} = 2$, whereas for $K_2 = 4$, the response gives $y_{ss} = \frac{4}{4-1} = \frac{4}{3} \approx 1.3$. These analytical results match the graphical results obtained through MATLAB. The other method for determining steady-state response value is through the use of the "Final Value Theorem." The Final Value Theorem uses the definition of the Laplace transform to show that when the response is stable, $y_{ss} = \lim_{t\to\infty} y(t) = \lim_{s\to 0} sY(s)$. This result may be explained in an intuitive manner as well: when $s \to 0$, the corresponding eigenfunction limits to $\lim_{s\to 0} e^{st} = 1$. In other words, a single pole at the origin corresponds to a constant function. This is also why $\mathcal{L}^{-1}\{\frac{1}{s}\} = \mu(t) \implies \mathcal{L}^{-1}\{\frac{1}{s}\} = 1,\ t \geq 0$, and why the natural response corresponding to $s = 0$ yields a constant value in Figure 5. Applying the Final Value Theorem to the closed-loop transfer function of this system therefore gives

$$\begin{aligned} y_{ss} &= \lim_{s\to 0} sY(s) = \lim_{s\to 0} s\tilde{H}(s)R(s) = \lim_{s\to 0} s\tilde{H}(s)\mathcal{L}\{\mu(t)\} = \lim_{s\to 0} s\tilde{H}(s)\left(\frac{1}{s}\right) \\ &= \lim_{s\to 0} \tilde{H}(s) = \lim_{s\to 0} \frac{K}{s-1+K} = \frac{K}{K-1}, \end{aligned}$$

which gives the same result as the analysis performed using limits in the time domain.

The second question to study is the following: if $K < 1$ yields an unstable system and $K > 1$ yields a stable system, what happens when $K = 1$? In other words, what happens when a controller is chosen such that the closed-loop system's poles are at "critical points?" The answer to this question depends on the system being analyzed. One temptation may be to set $K = 1$ in the closed-loop transfer function or step response (10), then either use the Final Value Theorem or take the limit as $t \to \infty$. However, these methods for determining steady-state behavior depend on the system being stable! As of now, the stability of the system is unknown. However, one method for answering this question involves the conversion of the closed-loop transfer function into the system's corresponding differential equation. Recall that $\mathcal{L}\{\dot{x}(t)\} = sX(s)$ without taking initial conditions into account. Then the transfer function gives

$$\tilde{H}(s) = \frac{Y(s)}{R(s)} = \frac{K}{s - 1 + K} \implies (s - 1 + K)Y(s) = KR(s) \implies \dot{y}(t) + (K - 1)y(t) = Kr(t).$$

Now setting $K = 1$ and solving the differential equation yields

$$\dot{y}(t) = r(t) \implies y(t) = \int_{0^-}^{t} r(\tau)d\tau.$$

Taking the reference signal to be the unit step, $r(t) = \mu(t)$, the output becomes the unit ramp signal;

$$y(t) = \hat{r}(t) - \hat{r}(0^-),$$

where the ramp is defined as $\hat{r}(t) \coloneqq \begin{cases} 0, & t < 0, \\ t, & t \geq 0. \end{cases}$ Hence, $\hat{r}(0^-) = 0$, which yields

$$y(t) = t, \quad t \geq 0.$$

Therefore, the step response when $K = 1$ is described by a linear increase in y. This behavior is unbounded and therefore taking the controller to be $C(s) = K = 1$ results in an unstable closed-loop system. △

Although the controller design procedure outlined in Example 21 is relatively straight forward and easily implemented in an analytical fashion, many other control design problems are not as accommodating. When the controller becomes more complicated than a simple proportional controller or the open-loop system of interest is of a higher order, alternative controller synthesis tools, such as those found in MATLAB's Control System Toolbox, become the preferred approach for controller design problems. These computational tools make use of the frequency response concepts introduced in Section 2.7, the step and impulse response ideas discussed throughout this document, and two other very common kinds of plots found throughout the field of control theory: Nyquist plots and root locus plots. Nyquist plots are useful in analyzing a system's stability margins, which give a quantitative idea of how close a system is to being unstable. These stability margins can be used as design constraints in the controller synthesis process. In other words, if the open-loop system's model is crude or susceptible to variations throughout time or operating conditions, it might be advantageous to design a more "robust" controller that maintains large stability margins to reduce the likelihood of the system going unstable due to the uncertainties in the model or parametric deviations through time. This stability margin analysis may also be carried out using frequency response plots, and therefore Nyquist plots will not be formally introduced in this document. Root locus plots, on the other hand, offer a plethora of graphical interpretations of a system's dynamics and how they change according to various controllers and their gain values.

3.4 Root Locus Plots

A root locus plot is simply a graph showing the locations of a system's closed-loop poles, and how those locations within the complex plane change as a controller constant is varied. The rules for sketching root locus plots by hand are included in Appendix A. Reading through these rules is highly encouraged. However, the important aspects of root locus plots are reviewed here. Some of the qualitative rules of root locus plots are as follows:

- Root locus plots are plots of a system's closed-loop poles in the complex plane.
- Root locus plots are symmetric across the real axis, so long as the coefficients of the system's characteristic polynomial are real. This is the case for physical systems.
- Each branch of a root locus corresponds to one of the system's closed-loop poles.
- For causal systems, each branch of a root locus starts at a pole of the open-loop system.
- The branches of the locus converge toward zeros of the open-loop system, if any exist. Zeros of a system are points in the complex plane that make the numerator of the system's transfer function equal zero. Some system's have no zeros, and therefore some locus branches diverge to infinity. In the case that there are more open-loop poles than open-loop zeros (causal systems), the number of poles minus the number of zeros gives the number of divergent branches of the locus.

Strictly speaking, the concept of a root locus plot may be extended to a more broad class of mathematical problems: how do the roots of a polynomial change as one coefficient of the polynomial changes? This purely mathematical problem is applied to the field of control theory in order to assist with controller design and analysis. Unfortunately, the most important feature of root locus plots is often the least understood, and therefore it is worth restating. *A root locus plot shows how a closed-loop system's poles move throughout the complex plane as a controller parameter is varied. The closed-loop poles start at the same locations as the open-loop system's poles, when the controller constant is $K = 0$. The closed-loop poles then move throughout the complex plane as K increases, with some closed-loop poles converging toward the open-loop system's zeros.*

Example 22: Let the system of interest be the same as that from Example 21;

$$H(s) = \frac{1}{s-1}.$$

Clearly, this open-loop system is unstable with a single pole at $\lambda = 1$. In Example 21, it was shown analytically that the system is stabilized by a proportional controller $C(s) = K > 1$. Using the Matlab code of Listing 2, the root locus plot of this system with a proportional controller $C(s) = K$ is graphed in Figure 13. Note that, as found in Example 21, increasing K moves the closed-loop pole leftward in the complex plane, thus speeding up the system's step response.

Listing 2: Matlab code for generating root locus plots.

```
1 s = tf('s');
2 H = 1/(s - 1);
3 rlocus(H);
```

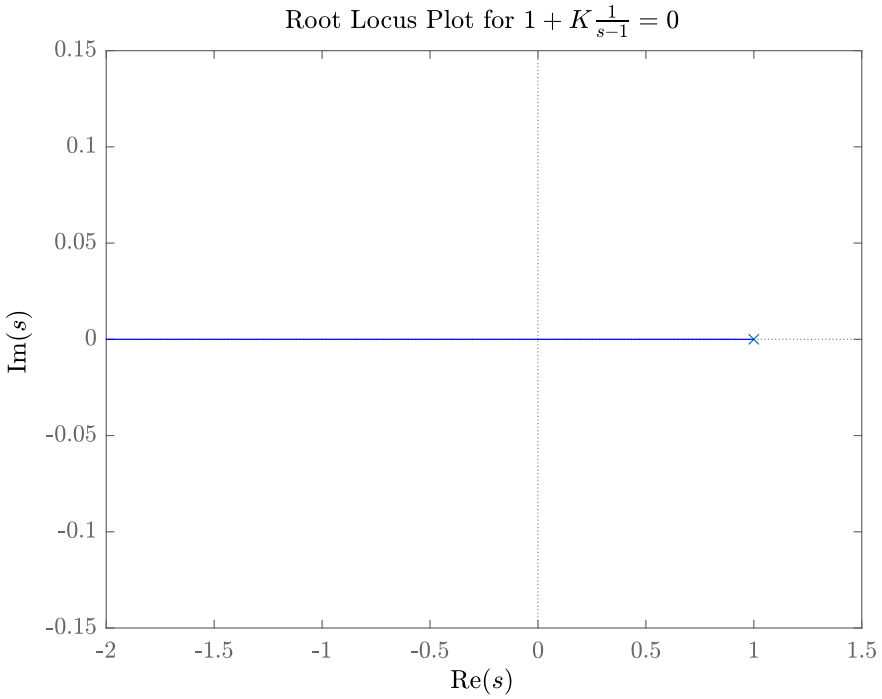

Figure 13: Root locus plot for the open-loop plant $H(s) = \frac{1}{s-1}$ using a proportional controller $C(s) = K$. There is only one locus branch, since the closed-loop system's characteristic polynomial is first-order. The closed-loop pole starts at the open-loop pole, $s = 1$, denoted by the $\times$. The closed-loop branch then moves leftward as K increases. When $K = 1$, the closed-loop pole crosses over the point $s = 0$, which gives stable closed-loop poles for $K > 1$.

This example yields a relatively bland root locus plot. More interesting locus patterns are given in Example 23. $\triangle$

Example 23: Let the system of interest be defined by the following transfer function:

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

Using this transfer function in the MATLAB code of Listing 2, the root locus plot of Figure 14 is generated. Note that the open-loop system has poles at $s^2 - 2 + 2 = 0 \implies (\lambda_1, \lambda_2) = (1 + j, 1 - j)$, which both have positive real parts, and therefore the open-loop system is unstable. However, with a high enough controller gain, $C(s) = K$, the root locus plot shows that the closed-loop system can be stabilized. This critical gain value can be found in MATLAB by simply clicking one of the root locus branches at the point where the locus branch crosses the imaginary axis. This critical gain may also be found analytically through a variety of methods such as the Routh-Hurwitz stability criterion. Using any method, it turns out that when $K = 2$, the closed-loop poles become purely imaginary: $(\lambda_1, \lambda_2) = (j2\sqrt{2}, -j2\sqrt{2})$. For $K > 2$, the poles move into the left-half of the complex plane and therefore any controller $C(s) = K > 2$ stabilizes this system. Furthermore, the branches of the locus meet on the real axis around $s \approx -7$ when $K \approx 16$. This is also an important gain value, since any gain larger than this yields purely real closed-loop poles. Therefore, proportional controllers with a gain $K > 16$ not only stabilize this system, but they also remove oscillatory components from the system's natural responses! However, for $K \gg 16$, one of the branches moves back rightward toward the open-loop zero at $s = 3$, which actually slows down that closed-loop

pole. Hence, the ideal controller which stabilizes the system, removes oscillations, and maintains fast responses would likely be chosen to be $K \approx 16$. This special gain yields what is termed "critical damping."

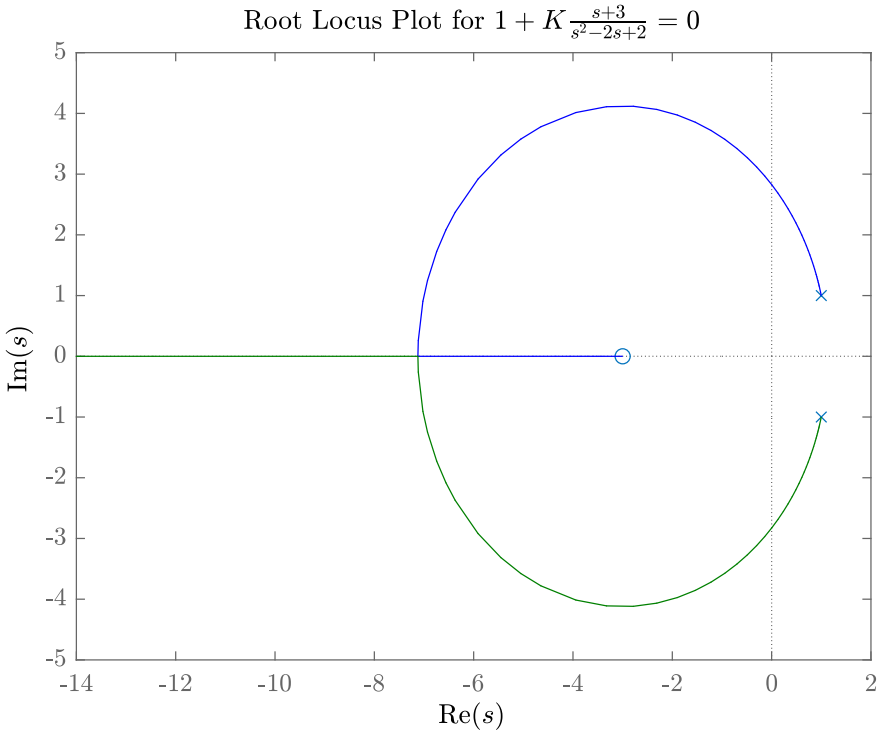

Figure 14: Root locus plot for the open-loop plant $H(s) = \frac{s+3}{s^2-2s+2}$ using a proportional controller $C(s) = K$.

△

Example 24: Suppose a system is defined by the following transfer function:

$$H(s) = \frac{1}{s^2 - 2s + 2}.$$

This system is similar to that of Example 23, since it contains the same unstable open-loop poles. Using a proportional controller, $C(s) = K$, the root locus plot is shown in Figure 15. Note that, no matter the value of the controller gain K, the closed-loop system's poles remain in the right-half plane, and therefore a proportional controller cannot be used to stabilize this system.

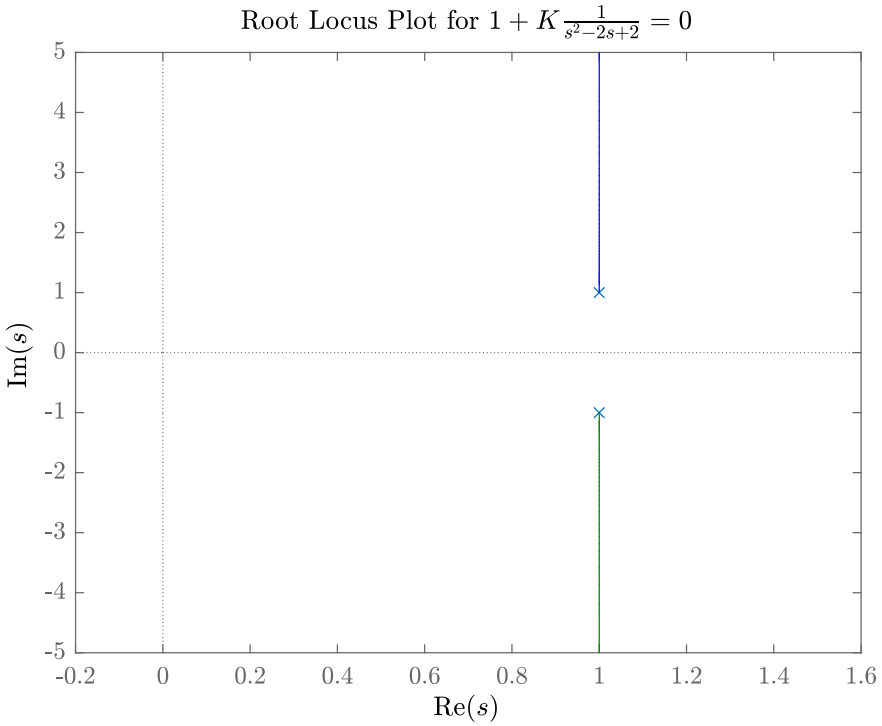

Figure 15: Root locus plot for the open-loop plant $H(s) = \frac{1}{s^2-2s+2}$ using a proportional controller $C(s) = K$. The real part of the closed-loop poles remains positive, and therefore a proportional controller cannot stabilize this system.

Since a simple proportional controller cannot stabilize this system, a more sophisticated approach may be taken. Suppose now the controller is taken to be of the form $C(s) = K(1 + s)$, where K is the still the tuning parameter. Using the unity feedback configuration shown in Figure 9, the closed-loop transfer function may be found as follows:

$$\begin{cases} Y(s) = H(s)U(s), \\ U(s) = C(s)E(s), \\ E(s) = R(s) - Y(s), \end{cases} \implies Y(s) = H(s)C(s)(R(s) - Y(s)),$$

$$(1 + H(s)C(s))Y(s) = H(s)C(s)R(s),$$

$$Y(s) = \frac{H(s)C(s)}{1 + H(s)C(s)}R(s).$$

Hence, the closed-loop transfer function is $\tilde{H}(s) = \frac{H(s)C(s)}{1+H(s)C(s)}$, which gives closed-loop poles satisfying the following characteristic equation:

$$1 + H(s)C(s) = 0.$$

This equation holds true for any $H(s)$ and $C(s)$ placed in the unity feedback configuration of Figure 9. This characteristic polynomial needs to be placed into root locus form before generating the root locus plot of interest:

$$1 + H(s)C(s) = 1 + \left(\frac{1}{s^2 - 2s + 2}\right)(K(1 + s)) = 1 + K\left(\frac{s + 1}{s^2 - 2s + 2}\right) = 0.$$

Hence, the equivalent open-loop transfer function, which takes into account both the original open-loop system's transfer function and the form of the controller, becomes

$$\hat{H}(s) = \frac{s+1}{s^2 - 2s + 2}.$$

This equivalent open-loop transfer function is the transfer function to call MATLAB's `rlocus` function on, as shown in Listing 3. This root locus plot is shown in Figure 16. As seen, this updated controller does have the ability to stabilize the system. Clicking the plot at one of the imaginary axis crossings gives the controller gain as $K \approx 2$. Hence, $K > 2$ yields a stable system.

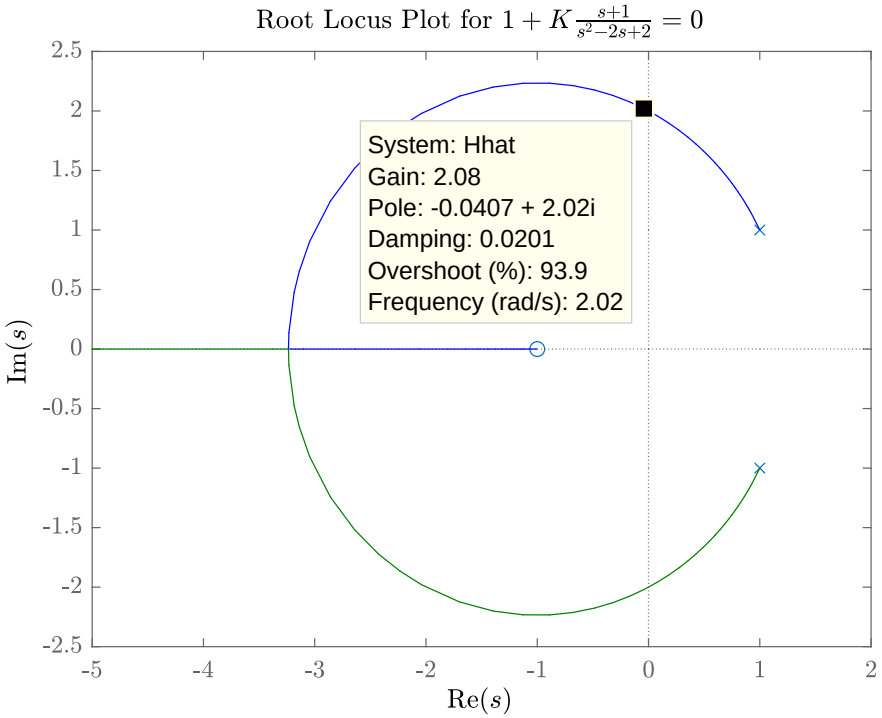

Figure 16: Root locus plot for the open-loop plant $H(s) = \frac{1}{s^2-2s+2}$ using a controller $C(s) = K(1+s)$. The closed-loop poles move to the left-half of the complex plane for $K > 2$, and therefore the system is stabilizable.

Listing 3: MATLAB code for generating root locus plots.

```
s = tf('s');
H = 1/(s^2 - 2*s + 2);
C = 1 + s;
Hhat = H*C;
figure();
rlocus(H);  % unstable
figure();
rlocus(Hhat);   % stable
```

△

3.5 Common Controllers

In Section 3.4, two different types of controllers were used to stabilize systems. The first was a proportional controller: $C(s) = K$. The second was of the form $C(s) = K(1 + s)$, which is termed a "proportional-derivative" controller, or PD controller for short. These controllers are called proportional-derivative controllers because they are formed as a linear combination of a proportional controller and a derivative term. Recall that $\mathcal{L}\{\dot{x}(t)\} = sX(s)$, which shows that derivatives appear as powers of s in the Laplace domain. Hence, a transfer function of the form $C(s) = Ks$ performs both a proportional scaling as well as a differentiation of a signal. Therefore, a PD controller, $C(s) = K(1 + s) = K + Ks$ adds together a scaled signal with its derivative. Similarly, since powers of $\frac{1}{s}$ in the Laplace domain are equivalent to integration in the time domain, a controller with the form $C(s) = K\left(1 + \frac{1}{s}\right)$ adds together a scaled signal with its integral. This controller is termed a "proportional-integral" controller, or PI controller for short. Some common controllers are listed and discussed in the following Sections.

3.5.1 P Controller

The P controller (proportional controller) simply scales a signal. These controllers take the form

$$C(s) = K_P, \quad K_P \in \mathbb{R}. \tag{11}$$

For many simple systems, a P controller in a unity feedback configuration can be used to stabilize or speed up a system's dynamics. However, there is often negative trade-offs, such as the introduction of overshoot and high-frequency oscillations, when increasing K_P to speed up a system's dynamics.

EXAMPLE 25: Let the open-loop system be

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

and the closed-loop system be a unity feedback loop with the proportional controller

$$C(s) = 5.$$

Then using MATLAB the open- and closed-loop step responses are plotted in Figure 17. Note that using a proportional gain of $K_P = 5$ did speed up the system's response, but that an increase in overshoot and oscillatory frequency was also introduced.

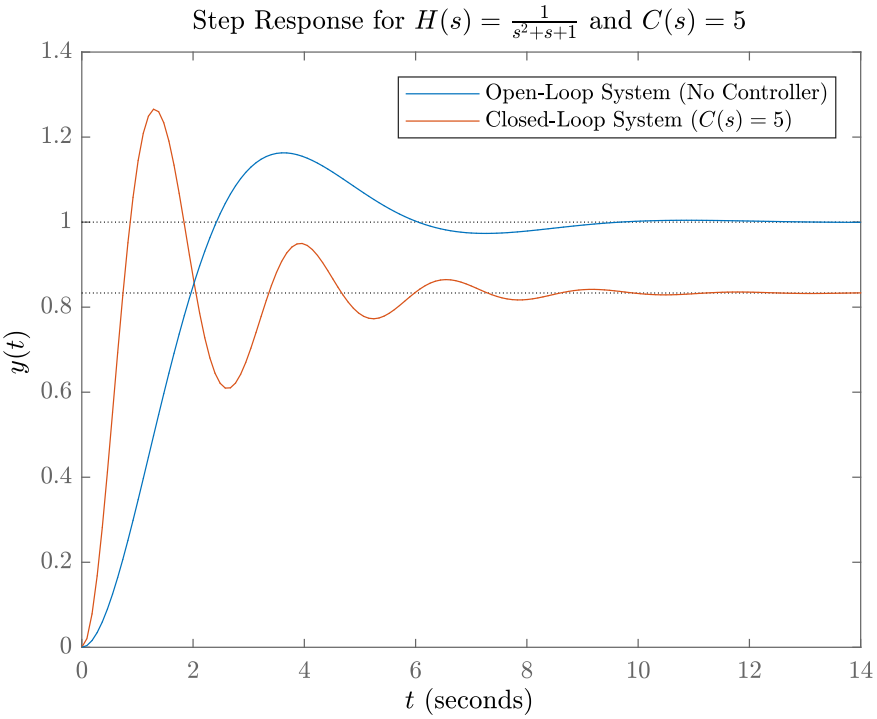

Figure 17: Step responses for the open-loop and closed-loop system using proportional control in a unity feedback configuration.

△

3.5.2 PD Controller

The PD controller (proportional-derivative controller) takes in a signal, then outputs the scaled signal summed with a scaled derivative of the signal. Mathematically, these controllers take the form

$$C(s) = K_P + K_D s, \quad K_P, K_D \in \mathbb{R}. \tag{12}$$

Note that generally, $K_P \neq K_D$, and therefore PD controllers have two tuning parameters. Another method of representing a PD controller is in terms of the location of its zero and an overall tuning gain;

$$C(s) = K_{PD}(s + \alpha),$$

where $s = -\alpha$ is the zero of the controller. This form of representing a PD controller is useful, as it allows for strategic placement of the zero within the complex plane, which might be used to "cancel" one of the open-loop system's slow, stable poles.

Example 26: Let the open-loop system of interest have the following transfer function:

$$H(s) = \frac{1}{s^2 + 11s + 10}.$$

This system has one pole at $\lambda_1 = -10$ and another at $\lambda_2 = -1$. Clearly, the system is already stable. However, the pole $\lambda_1 = -1$ is much slower than the pole $\lambda_2 = -10$. Hence, a PD controller might be used in a unity feedback loop in order to reduce the effects of the slow pole and therefore speed up the response of the system. To do this, place the zero of the PD controller somewhere

near $s = -1$. Suppose the zero is taken as $s = -\alpha = -1.1$. Here, the controller gain will be chosen arbitrarily as $K_{PD} = 2$. Therefore, the controller's transfer function becomes

$$C(s) = 2(s + 1.1).$$

The open- and closed-loop step responses are shown in Figure 18. As expected, the controller's zero near $s = -1$ roughly cancels the open-loop system's pole, $\lambda_1 = -1$. This reduces the slow effects of this pole and leaves the dynamics of the closed-loop system to be dominated by the fast pole, $\lambda_2 = -10$.

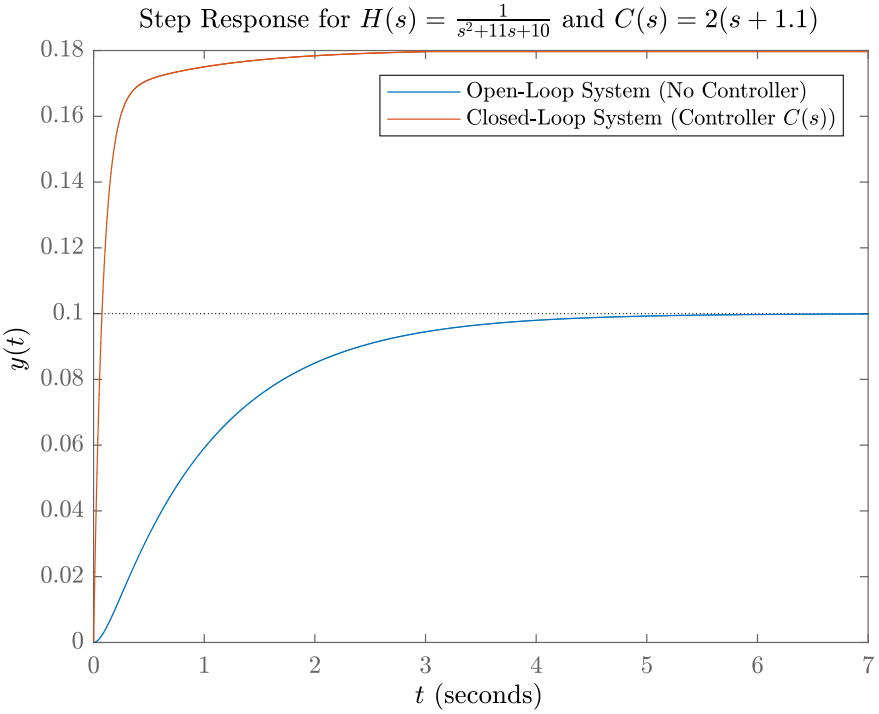

Figure 18: Step responses for the open-loop and closed-loop system using PD control in a unity feedback configuration. The zero of the controller was chosen to roughly cancel the slow pole of the open-loop system.

This strategic placement of zeros to remove the effects of certain poles is called "pole-zero cancellation." In practice, it is virtually impossible to ensure a perfect pole-zero cancellation, as slight deviations of the pole or zero location within the complex plane results in an imperfect cancellation. Although "near" pole-zero cancellations still reduce the effect of that pole on the dynamics of the system, an imperfect pole-zero cancellation of an unstable pole still results in the presence of an unstable pole, regardless of how reduced its effects are. Therefore, *a controller should not be designed to cancel an unstable pole!* Rather, a feedback loop should be employed to move the closed-loop poles into the stable region of the complex plane. △

One important aspect of PD control in a unity feedback loop is the smoothing effect it has on signals. In a sense, the derivative term anticipates changes in the error signal and adjusts the input to the plant accordingly. In other words, if the error signal is rapidly increasing at some point in time, then the derivative term acts according to this rate of change in order to counteract the increase of error in the future. This result is useful, as it tends to decrease overshoot in a system's step response.

3.5.3 PI Controller

The PI controller (proportional-integral controller) takes in a signal, then outputs the scaled signal summed with a scaled integral of the signal. Mathematically, these controllers take the form

$$C(s) = K_P + K_I \frac{1}{s}, \quad K_P, K_D \in \mathbb{R}.$$

As with the PD controller, generally $K_P \neq K_I$, and therefore PI controllers have two tuning parameters. Another method of representing a PI controller is in terms of a zero, a tuning gain, and a pole at the origin;

$$C(s) = K_{PI} \frac{s+\alpha}{s},$$

where $s = -\alpha$ is the zero of the controller. One of the uses of PI control in a unity feedback loop is to remove errors at steady-state. This may be seen by finding the transfer function from $R(s)$ to $E(s)$ in the unity feedback loop of Figure 9:

$$\begin{cases} E(s) = R(s) - Y(s), \\ Y(s) = H(s)U(s), \\ U(s) = C(s)E(s), \end{cases} \implies E(s) = R(s) - H(s)C(s)E(s),$$

$$(1 + H(s)C(s))E(s) = R(s),$$

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

Taking the controller to be a PI controller gives

$$E(s) = \frac{1}{1 + H(s)K_{PI}\frac{s+\alpha}{s}} R(s) = \frac{s}{s + K_{PI}(s+\alpha)H(s)} R(s).$$

Hence, the Final Value Theorem gives the steady-state error to be

$$e_{ss} = \lim_{s\to 0} sE(s) = \lim_{s\to 0} s \frac{s}{s + K_{PI}(s+\alpha)H(s)} R(s).$$

Taking the reference to be a step therefore gives

$$\begin{aligned} e_{ss} &= \lim_{s\to 0} s \frac{s}{s + K_{PI}(s+\alpha)H(s)} \mathcal{L}\{\mu(t)\} = \lim_{s\to 0} s \frac{s}{s + K_{PI}(s+\alpha)H(s)} \frac{1}{s} \\ &= \lim_{s\to 0} \frac{s}{s + K_{PI}(s+\alpha)H(s)}, \\ \implies e_{ss} &= 0, \end{aligned}$$

so long as $\lim_{s\to 0} H(s) \neq 0$. In other words, PI controllers may be used to eliminate steady-state error in a step response. This may be intuitively thought of as the controller adding up (integrating) the error over time and adjusting the control signal accordingly. So, as long as the error at steady-state is nonzero, the integral term of the PI controller will continue to supply a control input to the plant in order to adjust the steady-state dynamics to tend toward $e_{ss} = 0$.

Example 27: Let the system of interest be

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

and the closed-loop system be a unity feedback loop with the PI controller

$$C(s) = \frac{s+10}{50s}.$$

Note that, without any control, the open-loop system has a steady-state error equal to twice the size of the step reference input;

$$y_{ss} = \lim_{s\to 0} sY(s) = \lim_{s\to 0} sH(s)\mathcal{L}\{A\mu(t)\} = \lim_{s\to 0} s\frac{2}{s^2+s+1}\left(\frac{A}{s}\right) = 2A,$$

where A is the step size. Therefore, the steady-state error is $e_{ss} = r_{ss} - y_{ss} = A - 2A = -A$. However, since $\lim_{s\to 0} H(s) \neq 0$, a PI controller is able to remove this steady-state error. This is shown in Figure 19, where the step size was taken to be $A = 1$; $r(t) = \mu(t)$. Note, however, that the PI controller appears to have slightly slowed down the system's response.

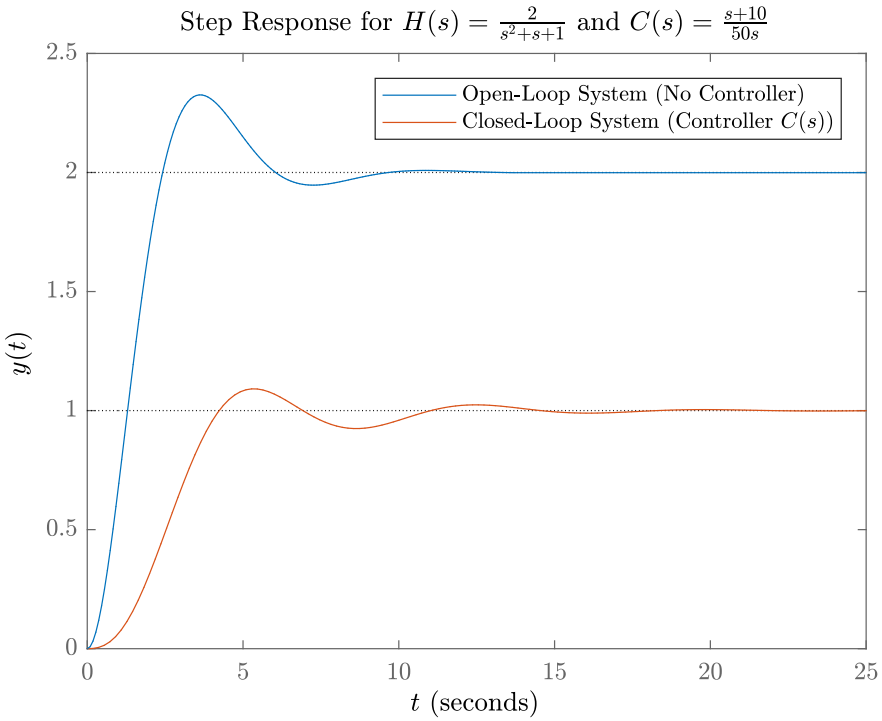

Figure 19: Step responses for the open-loop and closed-loop system using PI control in a unity feedback configuration. The integral term of the controller removes the steady-state error to the reference unit step input signal.

△

3.5.4 PID Controller

The PID controller (proportional-integral-derivative controller) linearly combines a proportional controller with an integral term and a derivative term. PID controllers take the form

$$C(s) = K_P + K_I\frac{1}{s} + K_D s, \quad K_P, K_I, K_D \in \mathbb{R}. \tag{13}$$

Clearly, PID controllers have three tuning parameters. A PID controller may also be put into the following form:

$$C(s) = K_{PID}\frac{(s+\alpha_1)(s+\alpha_2)}{s}.$$

In other words, a PID controller is composed of two zeros and a pole at the origin. Poles at the origin are typically called "integrators" and zeros at the origin are termed "differentiators." PID controllers are very popular in practice, as they are able to exploit all the benefits of the P, I, and D terms, while remaining fairly simple to work with and tune. One problem, however, is that many students or hobbyists learn about these popular controllers and use a PID algorithm without any underlying idea of their system's uncompensated dynamics. This leads people to use a guess-and-check approach to tuning the three constants of their PID controller. This is very bad practice for a multitude of reasons. First, it may take a very long time to stumble upon a combination of K_P, K_I, and K_D that results in performance that meets the design requirements of the system. Second, arbitrary tuning of these constants inherently results in "good enough" performance, which necessarily removes the ability to optimize the controller with respect to certain constraints or preferred dynamics. For example, a PID controller might be heuristically tuned until the system's performance meets the desired behavior. However, there may be a different PID controller that results in similar acceptable performance, but requires much less energy to drive the system. The third and possibly most important reason why one should never heuristically tune a PID controller is that it might result in an unstable system! Remember, an unstable system on paper or in MATLAB is not that big of a deal. However, an unstable system in real life could result in hundreds or thousands of dollars of mechanical and electrical components being ruined, or even the possibility of personal injury. PID controllers are not perfect and are not always the safe choice they seem to be. This is demonstrated in the example that follows.

EXAMPLE 28: The system of interest has the following transfer function:

$$H(s) = \frac{s(s+1)}{s^2-2s+2}.$$

This open-loop system has poles at $(\lambda_1, \lambda_2) = (1+j, 1-j)$. Hence, the uncompensated plant is unstable. Suppose a PID controller is used in a unity feedback loop, and that the controller gains are arbitrarily chosen as follows:

$$\begin{aligned} K_P &= -2, \\ K_I &= 26, \\ K_D &= 1. \end{aligned}$$

These values seem innocent enough, right? Of course, these are just initial guesses. These parameters will be tuned later to try and find "good enough" performance. This initial controller therefore becomes

$$C(s) = -2 + 26\frac{1}{s} + s = \frac{s^2-2s+26}{s}.$$

Now comes the time to test the controller. Suppose the input to the closed-loop system is a unit step. Performing the simulation in MATLAB gives the plot shown in Figure 20.

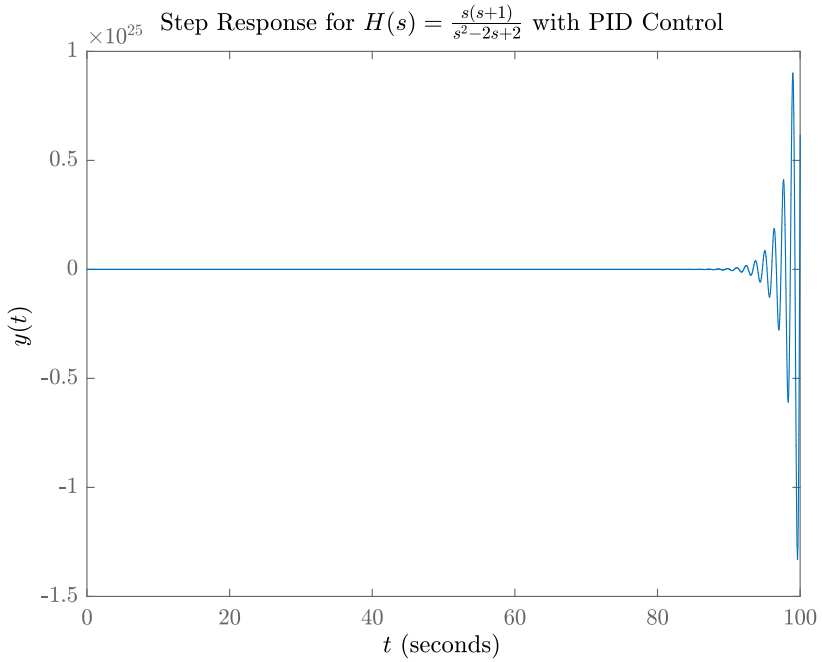

Figure 20: Step response of the closed-loop system using PID control in a unity feedback configuration. The PID constants were chosen arbitrarily as a starting point for controller tuning. However, the system displays unbounded, unstable behavior, demonstrating the dangers of guess-and-check tuning.

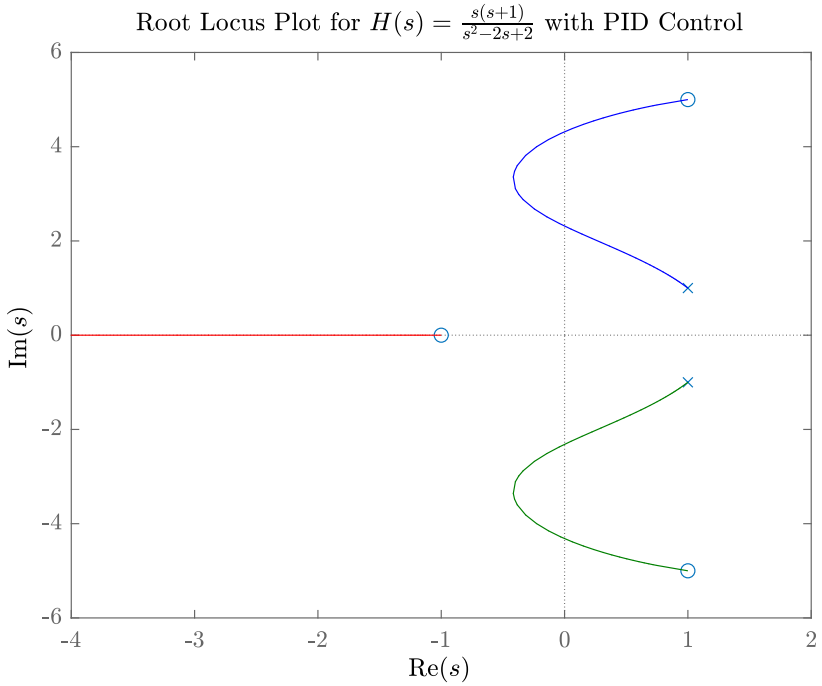

Figure 21: Root locus plot of the system $H(s)$ using PID control.

Clearly, something went wrong with this control scheme! Sure, the open-loop plant started off as unstable, but shouldn't the magical and ubiquitous PID controller solve all these problems? To answer this question, the root locus plot of the system $H(s)$ with the PID controller $C(s)$ is shown in Figure 21. Note that the characteristic polynomial that is being plotted against tuning parameter K is

$$1 + KH(s)C(s) = 1 + K\left(\frac{s(s+1)}{s^2 - 2s + 2}\right)\left(\frac{s^2 - 2s + 26}{s}\right) = 0.$$

As seen, there are three closed-loop poles, one of which remains real and stable for all K. However, the two remaining poles start off unstable when $K = 0$, then migrate into the left-half of the complex plane yielding a stable system, then turn back into the right-half plane making the closed-loop system unstable as $K \to \infty$. Using MATLAB, it is easy to find that $K \in (0.113, 0.359)$ yields a stable system. In other words, the PID controller

$$C(s) = K\frac{s^2 - 2s + 26}{s}, \quad K \in (0.113, 0.359),$$

will stabilize the system, but any other K outside this range will yield an unstable system. This is a rather tight constraint on K! Without having a decent model for the system, the root locus plot cannot be generated and therefore heuristically tuning a PID controller for this system would likely turn into a very frustrating, and possibly expensive or dangerous endeavor. Updating the controller so that $K = 0.2$ gives the closed-loop step response seen in Figure 22. As determined using the root locus plot, the closed-loop system is now stable.

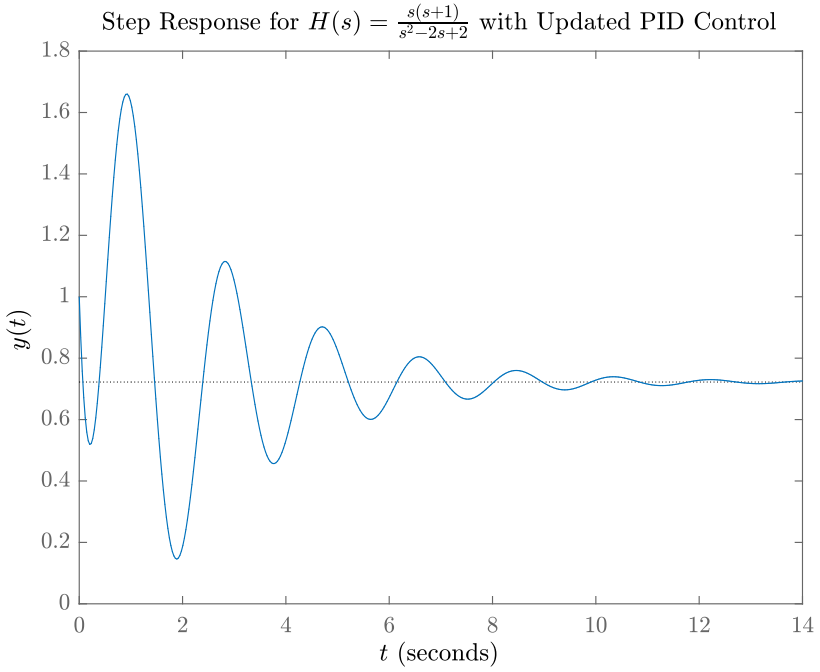

Figure 22: Step response of the closed-loop system using PID control in a unity feedback configuration. The PID constants were updated using a root locus plot to ensure the closed-loop system was stable.

△

3.5.5 Notch Filter

Another type of controller, typically placed in an open-loop series connection with the plant, is called a notch filter. Notch filters are useful in suppressing resonance. In other words, if a system has a high resonant peak at a certain frequency, and it is possible that the system will be excited at or near that resonant frequency, then one way of preventing catastrophic failure is to use a notch filter which attenuates that system's response at or near the resonant frequency. Notch filters typically take the following form:

$$C(s) = K_N \frac{s^2 + 2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta'\omega_n' s + \omega_n'^2}, \quad K_N, \zeta, \omega_n, \zeta', \omega_n' \in \mathbb{R}.$$

When placed in series with the plant, the parameters ζ and ω_n may be tuned to "cancel" the resonant poles of the plant and replace them with the poles defined by ζ' and ω_n'.

EXAMPLE 29: The plant of interest is defined by

$$H(s) = \frac{10000}{s^2 + 2s + 10000}.$$

As shown in Figure 23, a large resonant peak occurs at $\omega \approx 100\,\text{rad/s}$. To suppress this resonant behavior, suppose a notch filter is taken such that its zeros cancel the poles of the plant, its poles are critically damped ($\zeta' = 1$) at the natural frequency of $\omega_n' = 100\,\text{rad/s}$, and its gain is chosen to give unit gain at low frequencies:

$$C(s) = \frac{s^2 + 2s + 10000}{s^2 + 200s + 10000}.$$

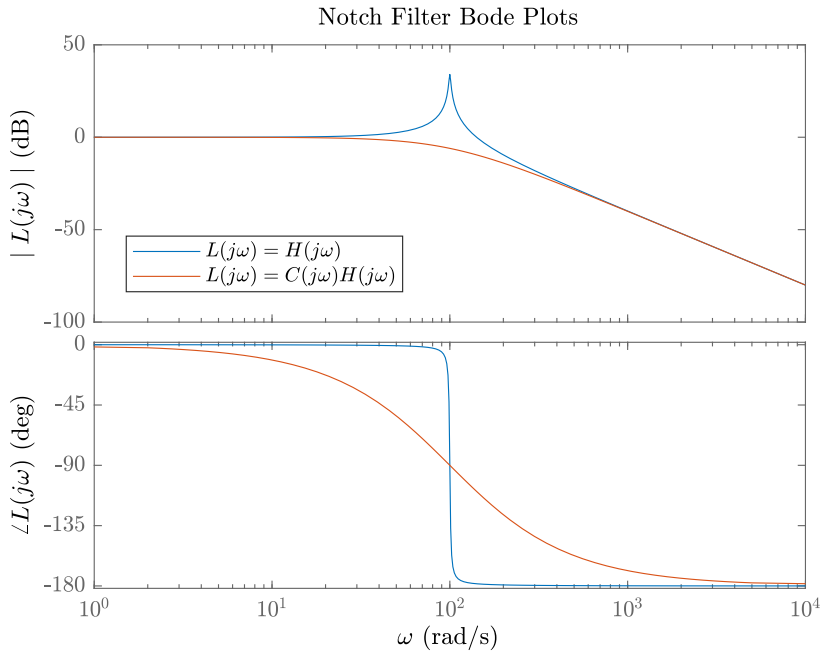

Figure 23: Bode plot of open-loop systems, denoted $L(s)$. The open-loop plant, $H(s) = \frac{10000}{s^2+2s+10000}$, displays resonant behavior at $\omega \approx 100\,\text{rad/s}$. A notch filter, $C(s) = \frac{s^2+2s+10000}{s^2+200s+10000}$, is used in series with the plant to suppress the resonance and yield critically damped poles.

The bode plot of the newly compensated system is also shown in Figure 23. Clearly, the notch filter successfully removed the resonant peak. Furthermore, the step responses of both the uncompensated and compensated systems are shown in Figure 24.

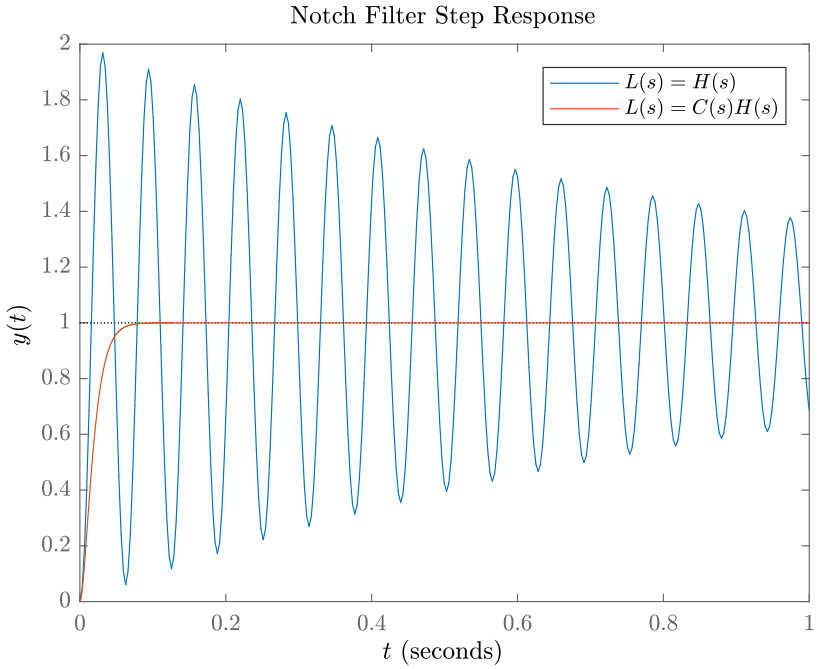

Figure 24: Step response of open-loop systems, denoted $L(s)$. The open-loop plant, $H(s) = \frac{10000}{s^2+2s+10000}$, has highly underdamped poles, yielding undesirable oscillations. A notch filter, $C(s) = \frac{s^2+2s+10000}{s^2+200s+10000}$, is used in series with the plant to suppress the resonance and yield critically damped poles.

△

3.6 Computer Design of Feedback Controllers

In practice, the process of designing a controller is typically done using the assistance of a computer. In this document, controller synthesis will be outlined using MATLAB's `sisotool` function, which is part of the Control System Toolbox. The process will be outlined using an example.

EXAMPLE 30: The unity feedback system of Figure 25 is given. The additional input, $D(s)$, models a disturbance input. Suppose a controller $C(s)$ must be designed to meet the following requirements:

- The output's overshoot should not exceed 5% in the presence of a unit disturbance step input.
- The steady-state output due to a constant unit disturbance must be $y_{ss} < \frac{4}{5}$.
- The closed-loop system should maintain a 90% rise time of $t_r < 1\,\text{s}$ to a unit step input $R(s)$.

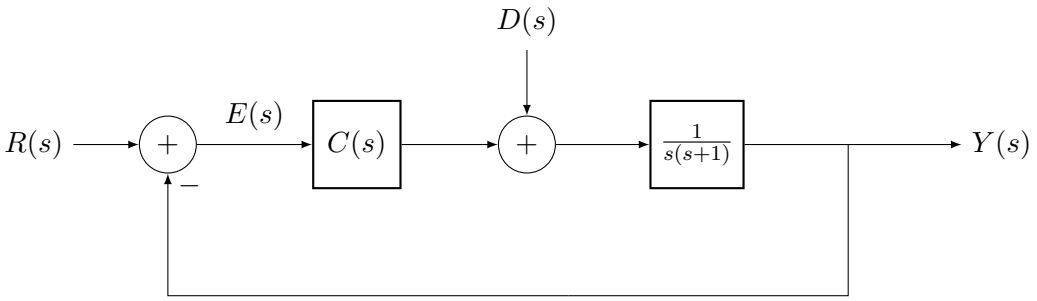

Figure 25: Closed-loop control system. The plant is given by $G(s) = \frac{1}{s(s+1)}$ and a disturbance input is modeled by $D(s)$.

The first step in designing $C(s)$ is to load the plant into MATLAB and open `sisotool`. The code is shown in Listing 4.

Listing 4: MATLAB code for controller design.

```
s = tf('s');
G = 1/(s*(s+1));     % plant transfer function
sisotool(G);
```

The `sisotool` window should open automatically, and it should look similar to Figure 26. Some of the plots might be different, depending on which plots are included in the default `sisotool` settings.

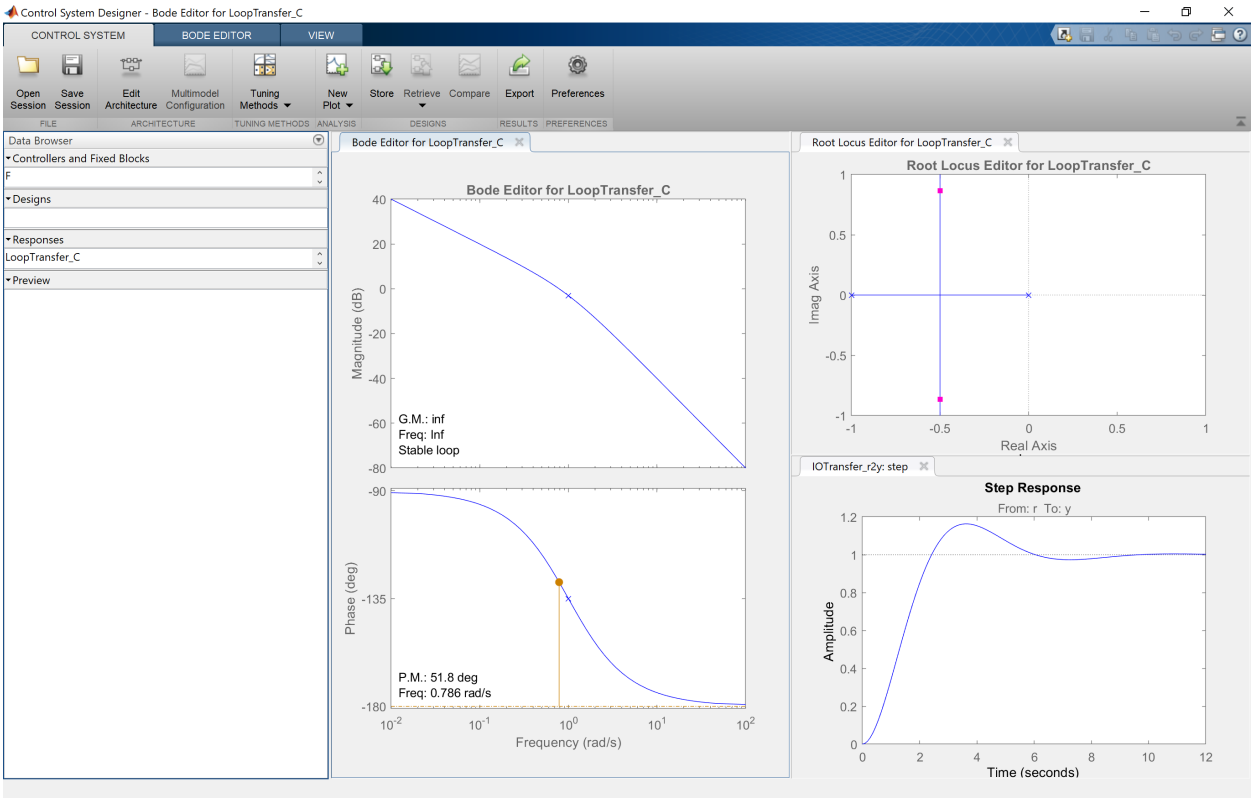

Figure 26: The `sisotool` graphical user interface.

By clicking the "Edit Architecture" button, it can be verified that the control system configuration does in fact match the unity feedback configuration of Figure 25. Note that, by default, MATLAB uses the notation $G(s)$ to represent the plant and $H(s)$ to represent the feedback gain or sensor transfer function. By default, $H(s)$ should be $H(s) = 1$, which matches the unity feedback setup used in this example.

Since some of the design specifications were stated in terms of the disturbance input $D(s)$, the appropriate plots must be generated before proceeding with the design of $C(s)$. Since the disturbance input is given as a constant, a step input for $D(s)$ should be used. This is accomplished by clicking the "New Plot" button and selecting "Step Response," then choosing the input-output relation between what MATLAB calls `du` and `y`, which gives the relation between the disturbance input $D(s)$ and the output $Y(s)$. If the step response plot between $R(s)$ and $Y(s)$ is not automatically shown, use the same method for generating it. This process is outlined in Figures 27 and 28.

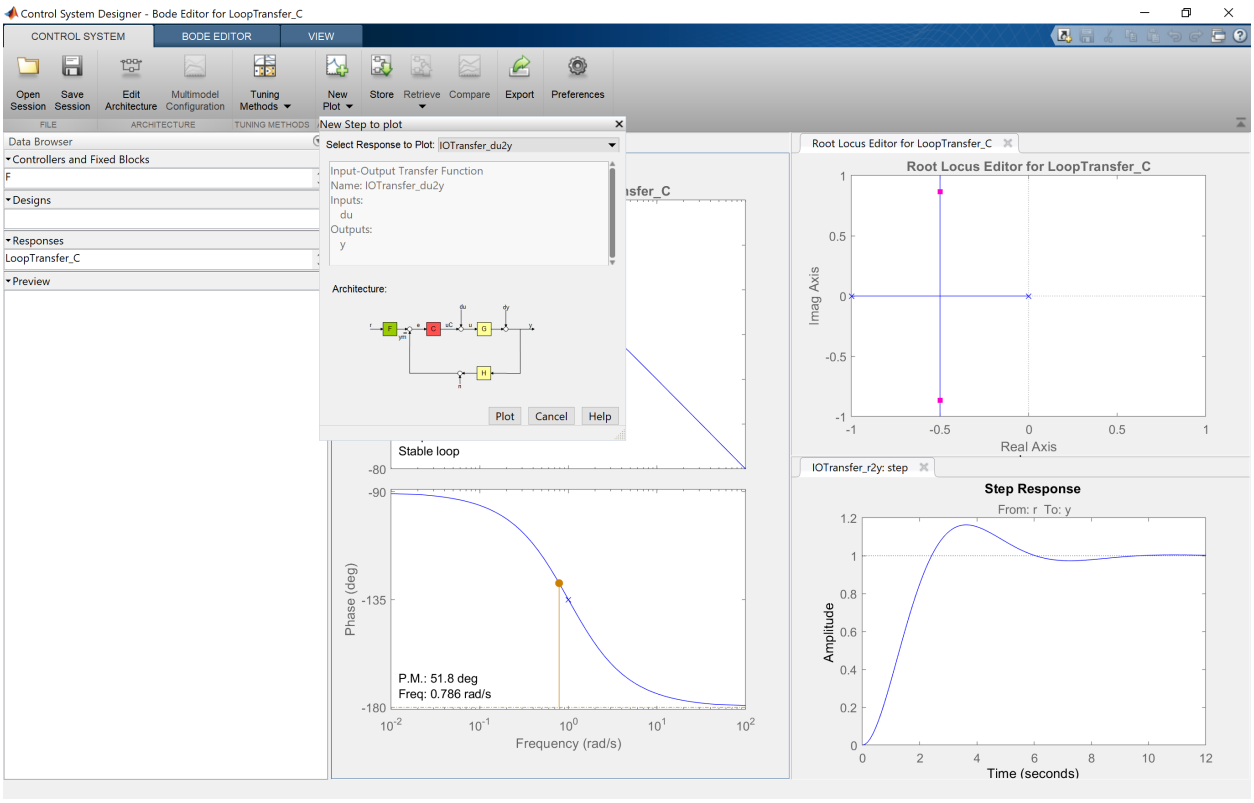

Figure 27: Adding a new plot to analyze.

From the $D(s)$ to $Y(s)$ step response plot, it is clear that the default controller, $C(s) = 1$, yields around a 16% overshoot. Therefore, a derivative term should likely be added to the controller to reduce the overshoot. Furthermore, the steady-state output to a constant unit disturbance appears to be $y_{ss} = 1$, which is greater than the desired $y_{ss} < \frac{4}{5}$. These response characteristics may be found by right-clicking the step response plot and selecting certain "Characteristics." To adjust the steady-state gain of the system, a proportional term should likely be added to the controller. The proportional term will also be useful in speeding up the rise time. Hence, the design requirements suggest a PD controller might satisfy the problem.

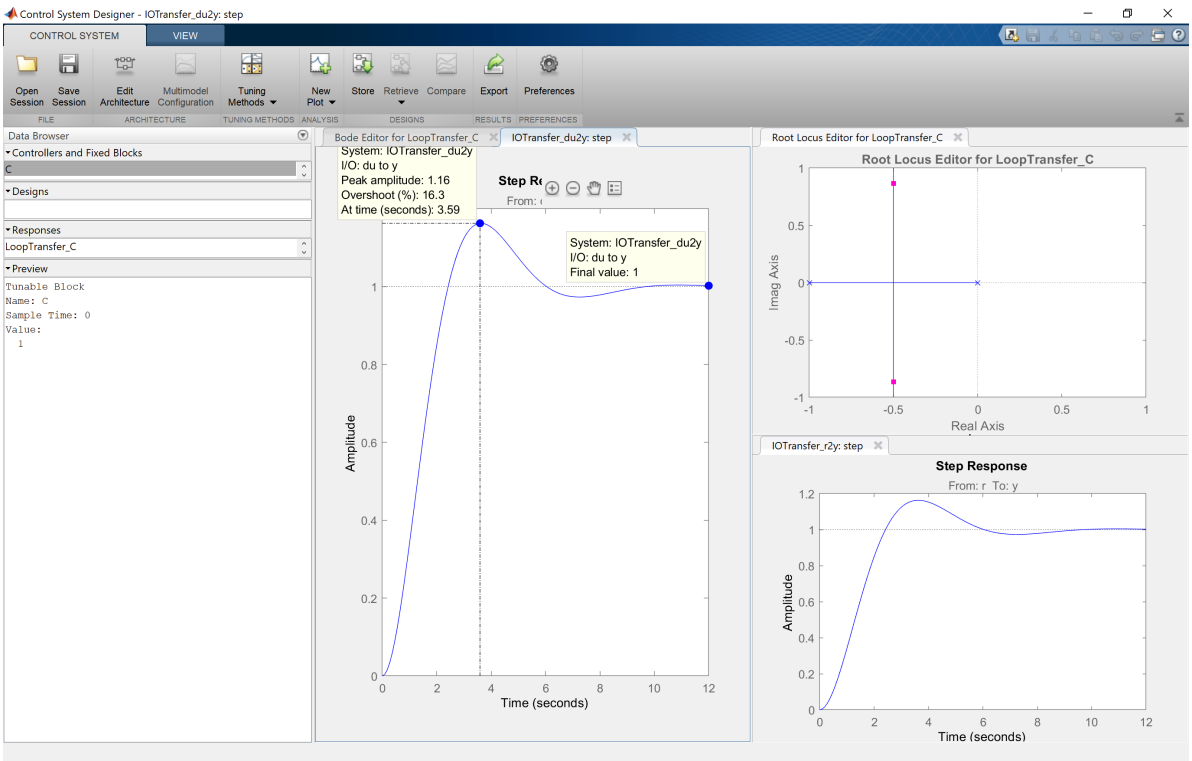

Figure 28: The new step response plot between $D(s)$ and $Y(s)$ is shown. Response characteristics may be displayed by right-clicking on the plot of interest and choosing certain "Characteristics."

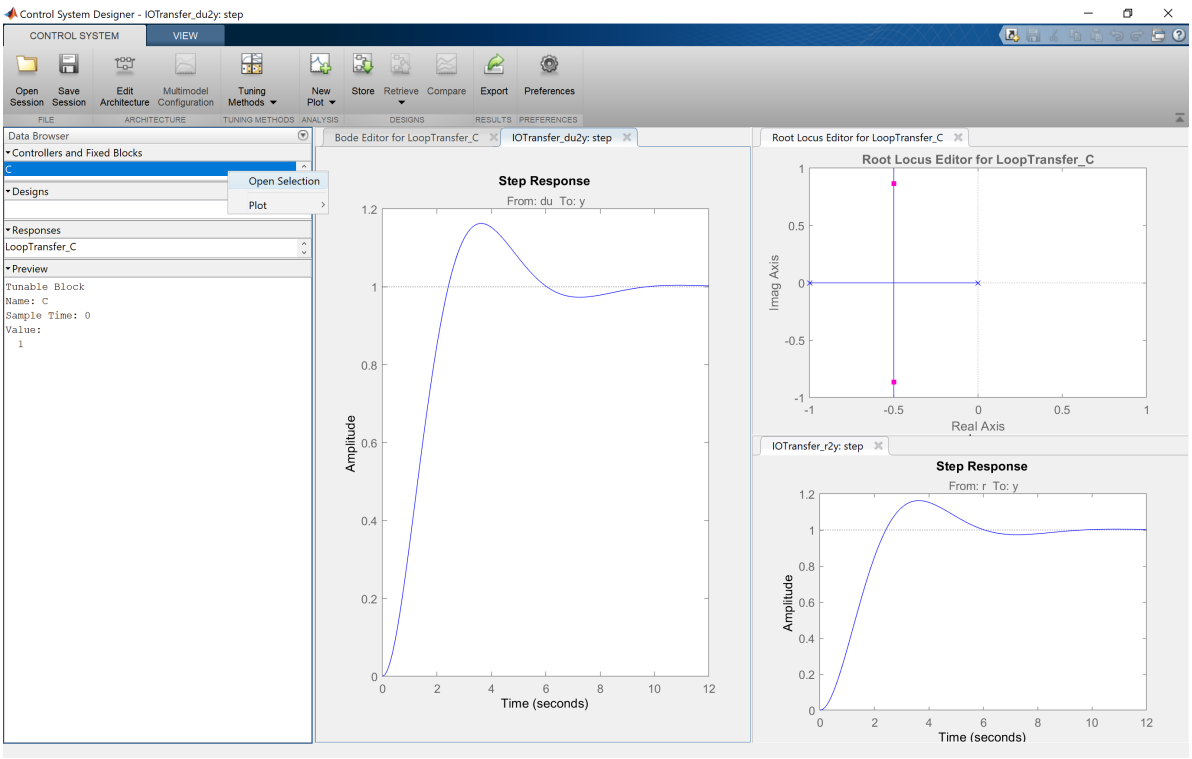

Figure 29: Use the "Data Browser" panel to open and adjust the controller.

To edit and view the controller, choose the `C` variable from the "Controllers and Fixed Blocks" drop-down menu in the "Data Browser" panel. Right-click on `C` and choose "Open Selection" in order to edit the controller. To add make `C` a PD controller, a real zero should be added, as shown by (12). This is done by right-clicking in the "Dynamics" box of the "Compensator Editor" and adding a "Real Zero." By clicking on the newly added zero, the "Compensator Editor" should look similar to that in Figure 30.

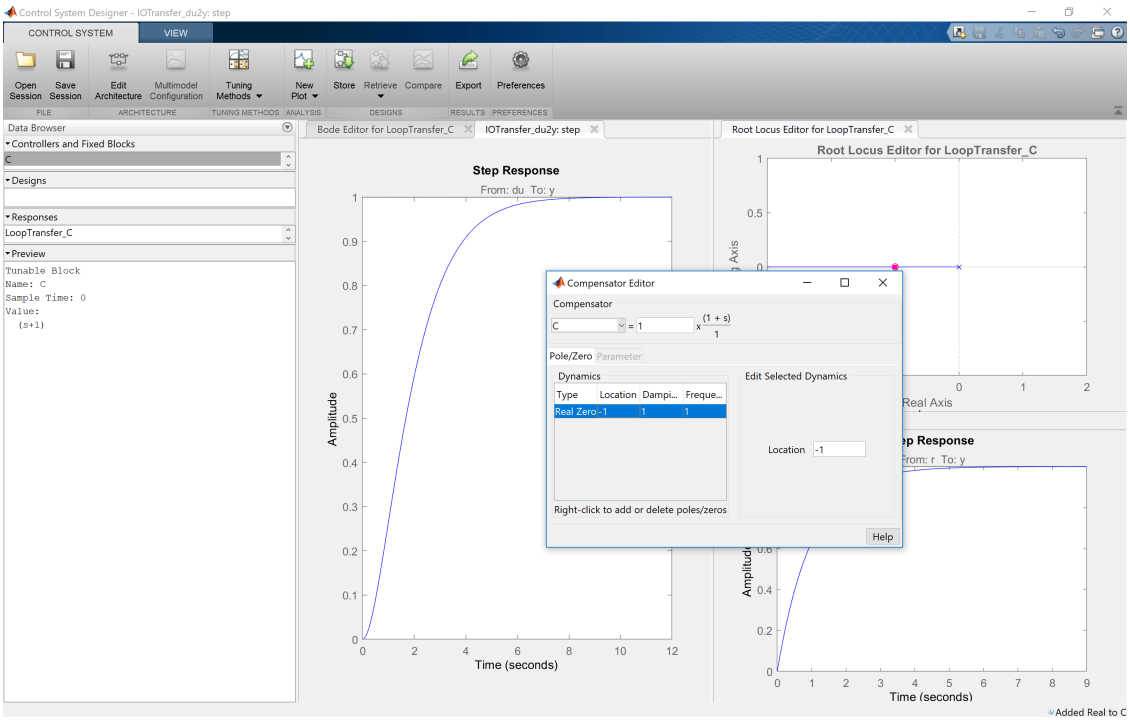

Figure 30: Adding a zero to the controller using the "Dynamics" box of the "Compensator Editor."

The zero that gets added might automatically be set at $s = -\alpha = -1$ in the complex plane. Coincidentally, this zero cancels one of the poles of the open-loop plant. This removes the oscillatory step behavior of the closed-loop system, as specified by the first design constraint on overshoot. By moving the zero farther left in the complex plane ($\alpha > 1$), K_P is made smaller, and therefore the overshoot returns. This is evidenced by the change in the root locus plot, where the closed-loop poles start taking on complex values for certain controller gains. By moving the zero to the right ($\alpha < 1$), K_P is made larger and one of the closed-loop poles is being made slower. In this case, all closed-loop poles remain real and therefore prevent oscillations from occurring in the step response. Therefore, taking $s = -\alpha = -1$ should suffice and yield well-balanced results.

MATLAB has a built in feature for inserting common design specifications. To do so, right-click in the plot of interest and select "New" from the "Design Requirements" option. As shown in Figure 31, the 5% overshoot and $y_{ss} < \frac{4}{5}$ final value are inserted. The 90% rise time requirement, $t_r < 1\,\text{s}$, can be added to the step response plot between $R(s)$ and $Y(s)$ in a similar manner. As shown in Figure 32, MATLAB shades the regions of the step response plots that are off-limits. The lower bound on the steady-state response from $D(s)$ to $Y(s)$ may be ignored, since the design specification states $y_{ss} < \frac{4}{5}$. Note that MATLAB is capable of imposing design requirements on other kinds of plots as well, such as Bode and root locus plots. The goal now is to adjust the controller gain until the responses meet the specifications.

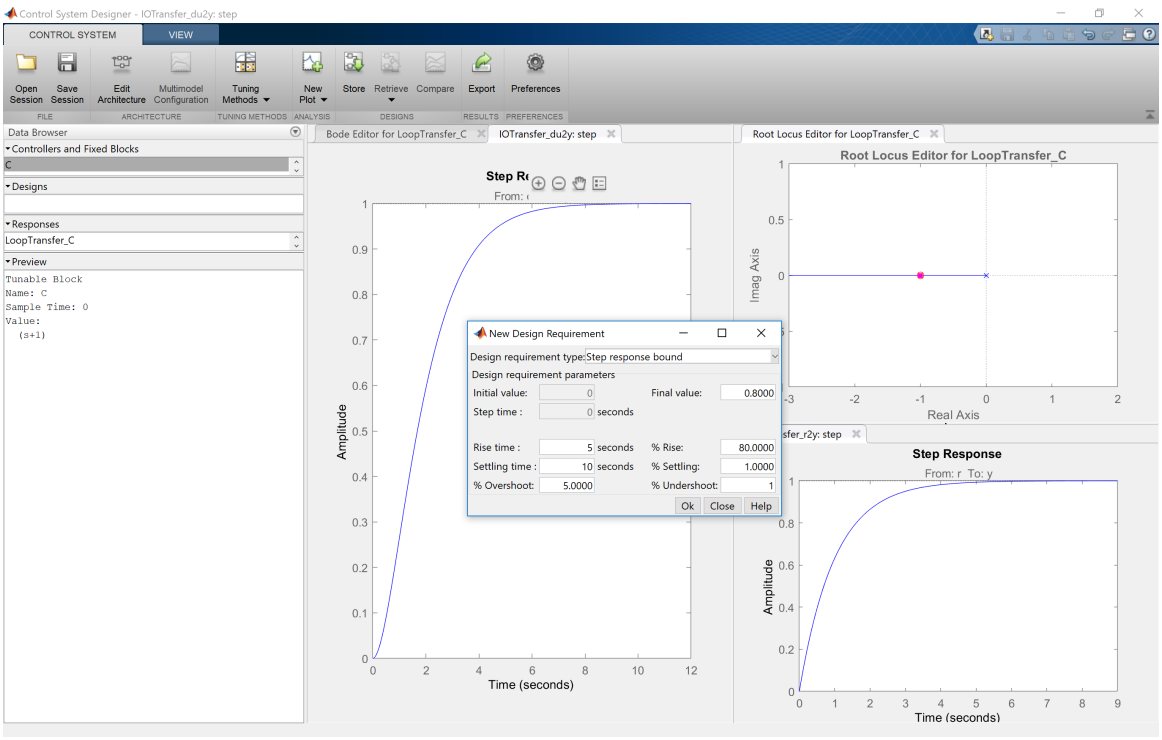

Figure 31: Adding design requirements by right-clicking in the step response plot and choosing "New" from the "Design Requirements" option.

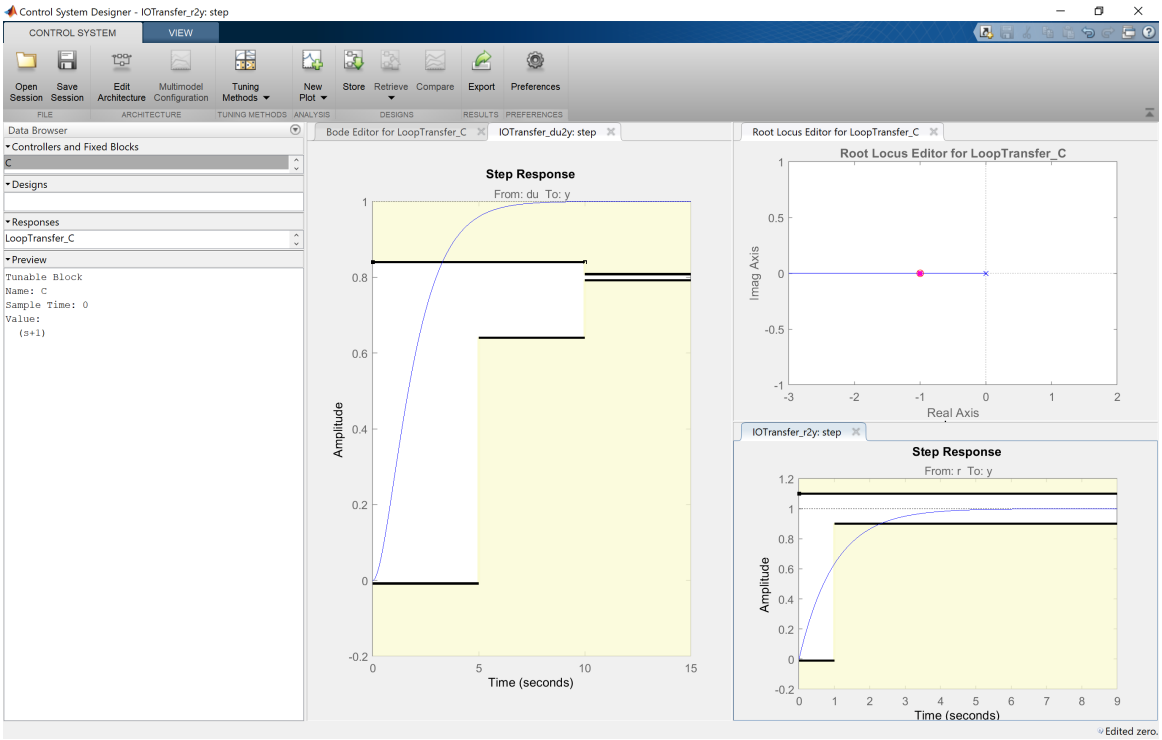

Figure 32: The shaded regions denote what is "off-limits". The controller should yield step response plots which completely avoid the shaded regions.

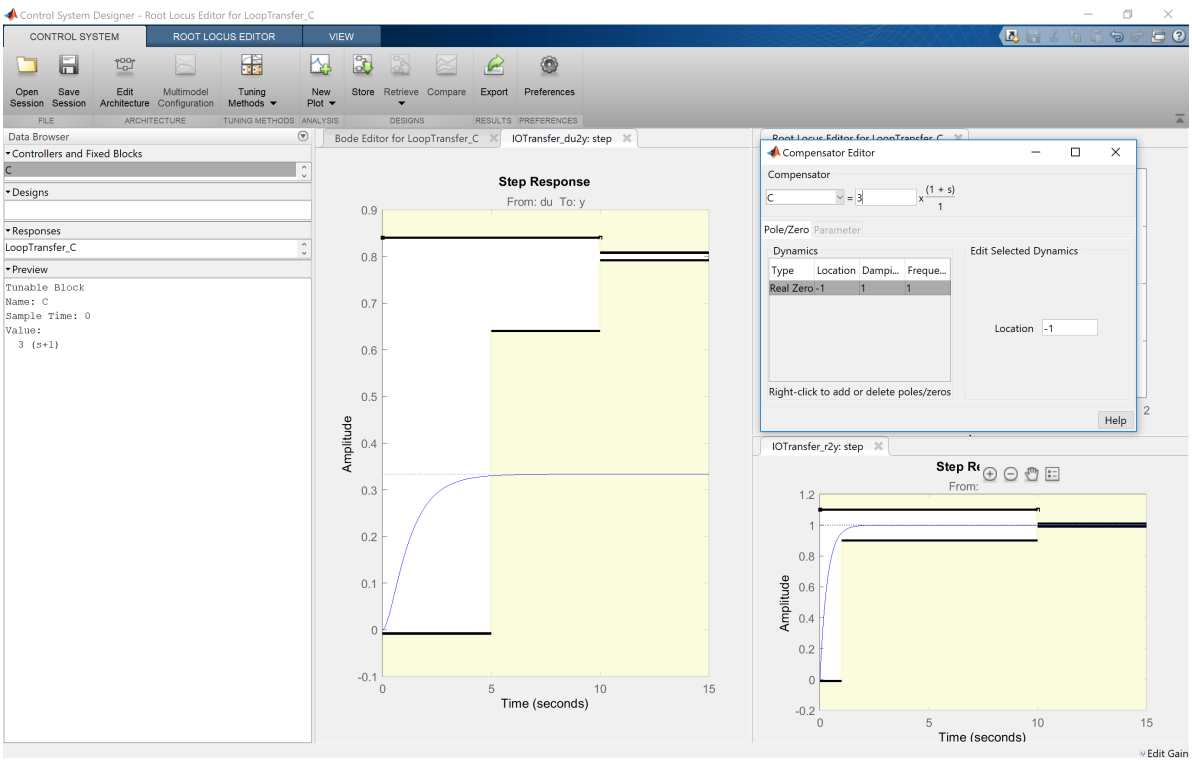

Figure 33: By adjusting the controller gain, the system's responses to step references and disturbances meet the design requirements.

As shown in Figure 33, adjusting the controller gain to $K_{PD} = 3$ within the "Compensator Editor" window yields the desired responses. Actually, increasing the controller gain past this point ($K_{PD} > 3$) speeds up the response and lowers the steady-state response in the presence of a disturbance input. These two effects may be desirable, however, in practice raising the controller gain might lead to what is called "control saturation." Control saturation occurs when a controller acts so aggressively that certain components within the system cannot keep up. For example, a DC motor might operate within the voltage range $V \in [-3\,\mathrm{V}, 3\,\mathrm{V}]$. However, if a controller is designed too aggressively, the control signal might require that the DC motor operate at $7\,\mathrm{V}$ to meet the intended closed-loop dynamics. This physical constraint clearly imposes realistic limitations on how aggressive a controller may be designed, and in fact, these constraints may be designed for by analyzing the input-output behavior between the reference or disturbance signal and the control input to the plant. For the current problem, the controller gain shall be left as $K_{PD} = 3$, since it yields the desired performance and remains conservative in terms of actuator requirements. By choosing the variable `C` from the "Controllers and Fixed Blocks" drop-down menu in the "Data Browser" panel, the final transfer function $C(s)$ is given as

$$C(s) = K_{PD}(s + \alpha) = 3(s + 1).$$

△

PROBLEM 3: An inverted pendulum is configured as shown in Figure 34. For this problem, all numerical values are assumed to be in SI units for convenience. The mass at the end of the pendulum is

$$m = 2.$$

The length of the pendulum is

$$l = 1.$$

The cart on which the pendulum is mounted has a mass

$$M = 5.$$

Assume gravitational acceleration is $g = 10$.

a. First start with a simplified problem; assume the cart is fixed and a motor may directly apply a torque, τ, to the base of the pendulum arm. Modeling the plant as an LTI system, design a controller to stabilize the system. Take the transfer function of the DC motor to be $M(s) = \frac{\mathcal{T}(s)}{V(s)} = \frac{1}{s+10}$, where $\mathcal{T}(s)$ is the motor's torque and $V(s)$ is the input voltage. Hint: Model the output as the angle $\theta(t)$. The reference input to the control system should therefore be the desired angle. The input to the controller should be an error signal, $E(s) = R(s) - \Theta(s)$, and the output of the controller can be the motor voltage, $V(s)$.

b. Design a controller that yields a rise time $t_r < 1$, an overshoot of $M_p < 0.1 = 10\%$, and a settling time of $t_s < 5$, all in the presence of a unit step reference input.

c. For extra practice, try repeating the Parts (a) and (b) including the moving cart. In other words, the actuator is no longer a motor at the pendulum base, but rather it is an external force pushing or pulling on the cart. Take the actuator's transfer function to be $A(s) = \frac{F(s)}{V(s)} = \frac{1}{s+10}$, where $F(s)$ is the actuator's force and $V(s)$ is the input voltage.

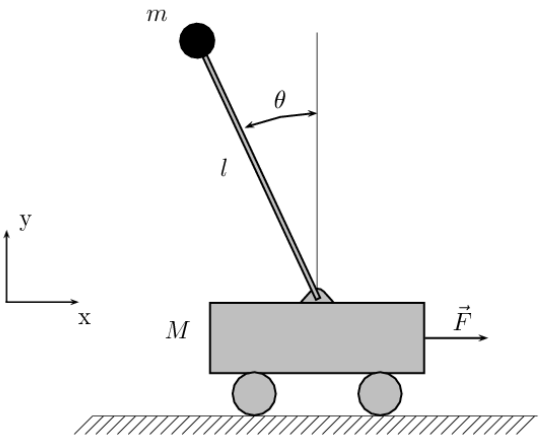

Figure 34: The inverted pendulum plant.

4 Additional Topics in Control Theory

Although the topics introduced in this document will likely suffice in designing a controller that can meet design requirements in practice, plenty of other fields within control theory have been developed to attack problems from a certain perspective. Some of the most popular subfields of control include the following:

- Digital Control
- Optimal Control
- Model Predictive Control
- Robust Control

The field of digital control is centered around the analysis and design of controllers which act in a discretized manner. This field of control is extremely important, since controllers are typically implemented using a digital computer. The two primary approaches to digital controller design include indirect and direct design. Indirect design typically involves the design of a controller using continuous-time models and analysis (as found throughout this document), then converting the continuous-time controller into an equivalent discrete-time controller. This is a desirable approach due to its simplicity. However, indirect digital controller design might introduce errors or changes in the dynamic behavior due to approximations and conversions between the continuous- and discrete-time domains. The second approach to digital control is termed direct design. Direct design typically involves the generation of a discrete-time equivalent model of the system of interest, then designing a controller in the discrete-time domain. The benefit to direct design is that the simulations and analysis of the discrete-time controller will match the actual performance closer than using an indirect approach. Regardless of whether direct or indirect design is used, MATLAB's Control System Toolbox may be used to analyze and design the controller.

Optimal control theory is the subject of finding a controller that meets certain design constraints or inequalities and simultaneously optimizes some cost function. An example might include finding the controller which meets certain rise time and overshoot requirements, while minimizing the current draw of the closed-loop system's actuator. Another example might include finding the controller which minimizes the H_∞ norm of a closed-loop system.

Model predictive control is similar to optimal control in the fact that it generally solves an optimization problem to determine the best control law to use. However, model predictive control utilizes real-time optimizations over finite time intervals. This allows for updates of the system's state measurements or estimations using empirical data and thus on-the-fly control law optimization which might yield more robust or better performance than an infinite horizon optimal control solution. MATLAB offers a Model Predictive Control Toolbox which allows for the design and simulation of model predictive controllers.

Another useful controller design approach is termed robust control. Robust control takes into account uncertainties in the plant's model or some of its parameters such that the control system maintains desirable behavior even when bounded perturbations in the plant's model or disturbances occur. MATLAB's Robust Control Toolbox is a well-established tool in analyzing and solving robust control problems.

Appendix A: Rules for Sketching Root Locus Plots

Rule #1: Root Locus Form

The characteristic equation of the closed-loop transfer function can be put into the form

$$1 + KL(s) = 0, \tag{14}$$

where K is the controller parameter to be varied from 0 to ∞, and $L(s) = \frac{b(s)}{a(s)}$ can be found by algebraic manipulation of the open-loop transfer function governing the dynamics of the plant, where $a(s)$ and $b(s)$ are both monic polynomials.

Rule #2: Divergent Branches

Note that (14) can also be rewritten as

$$1 + K\frac{b(s)}{a(s)} = 0.$$

At $K = 0$, $a(s) = 0$, and therefore the locus starts at the poles of $L(s)$. As $K \to \infty$, $b(s) \to 0$, and therefore the locus ends at the zeros of $L(s)$. For a strictly proper $L(s)$ with $m = \deg(b(s))$ and $n = \deg(a(s))$, $n > m$, and therefore m branches of the locus will be absorbed by the zeros of $L(s)$, and

$$d = n - m \tag{15}$$

branches will diverge as $K \to \infty$.

Rule #3: Real-Axis Segments

Equation (14) can be rewritten as

$$L(s) = -\frac{1}{K},$$

which, for $K \in \mathbb{R}_{\geq 0}$, implies $\angle L(s) = \pi \pm 2\pi(k-1)$ for $k \in \mathbb{Z}_{>0}$. That is, $L(s)$ yields a real and negative value for every point $s \in \mathbb{C}$ on the locus. Let s_0 be a real-axis point on the locus. It must be true that

$$\angle L(s_0) = \angle\frac{b(s_0)}{a(s_0)} = \angle b(s_0) - \angle a(s_0) = \pi \pm 2\pi(k-1).$$

$a(s_0)$ and $b(s_0)$ can be written as products of the n poles and m zeros of $L(s)$, respectively;

$$a(s_0) = \prod_{i=1}^{n}(s_0 - p_i),$$
$$b(s_0) = \prod_{i=1}^{m}(s_0 - z_i),$$

and therefore

$$\sum_{i=1}^{m}\angle(s_0 - z_i) - \sum_{i=1}^{n}\angle(s_0 - p_i) = \pi \pm 2\pi(k-1).$$

Note that for any zero $z \in \mathbb{C}$, both z and its conjugate z^* are roots of $b(s_0)$. Since $\angle z = -\angle z^*$, $\angle(s_0 - z) + \angle(s_0 - z^*) = 0$, and there is no phase contribution to due to complex zeros. The same

argument holds for complex poles. The remaining poles and zeros must real. For the r poles and zeros to the right of s_0, $\angle(s_0 - p) = \angle(s_0 - z) = \pi$. For the l poles and zeros to the left of s_0, $\angle(s_0 - z) = \angle(s_0 - p) = 0$. Therefore,

$$\sum_{i=1}^{m} \angle(s_0 - z_i) - \sum_{i=1}^{n} \angle(s_0 - p_i) = \sum_{i=1}^{r} \pi - \sum_{i=1}^{l} 0 = \pi \pm 2\pi(k-1),$$
$$\pi r = \pi \pm 2\pi(k-1),$$
$$r = 1 \pm 2(k-1), \quad k \in \mathbb{Z}_{>0}. \tag{16}$$

Therefore, an odd number of real poles and zeros must be to the right of any real-axis points on the locus. In other words, the real-axis segments of the locus fall to the left an odd number of poles and zeros.

Rule #4: Center and Angle of Asymptotes

Rewriting (14),

$$1 + K\frac{\prod_{i=1}^{m}(s - z_i)}{\prod_{i=1}^{n}(s - p_i)} = 0.$$

Note that through polynomial expansion,

$$\prod_{i=1}^{m}(s - z_i) = s^m - s^{m-1}\sum_{i=1}^{m} z_i + s^{m-2}\sum_{i=1}^{m-1} z_i \sum_{j=i+1}^{m} z_j - s^{m-3}\sum_{i=1}^{m-2} z_i \sum_{j=i+1}^{m-1} z_j \sum_{k=j+1}^{m} z_k + \cdots$$

Note that for strictly proper systems with $n > m$, $K \to \infty$ requires either $b(s) \to 0$ or $|s| \to \infty$. For the $n - m$ locus branches that aren't absorbed by zeros, $|s| \to \infty$. As $|s| \to \infty$, the leading terms dominate, and therefore the characteristic equation can be approximated by

$$0 \approx 1 + K\frac{s^m - s^{m-1}\sum_{i=1}^{m} z_i}{s^n - s^{n-1}\sum_{i=1}^{n} p_i},$$
$$= 1 + K\frac{1 - s^{-1}\sum_{i=1}^{m} z_i}{s^{n-m} - s^{n-m-1}\sum_{i=1}^{n} p_i},$$

where by binomial approximation,

$$1 - s^{-1}\sum_{i=1}^{m} z_i \approx \left(1 + s^{-1}\sum_{i=1}^{m} z_i\right)^{-1}$$

for large $|s|$. Hence,

$$0 \approx 1 + K\frac{1}{\left(s^{n-m} - s^{n-m-1}\sum_{i=1}^{n} p_i\right)\left(1 + s^{-1}\sum_{i=1}^{m} z_i\right)}.$$

Dropping the lowest order term yields

$$0 \approx 1 + K\frac{1}{s^{n-m} - s^{n-m-1}\left(\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} z_i\right)},$$
$$= 1 + K\frac{1}{s^{n-m} - \eta s^{n-m-1}},$$

where $\eta = \sum_{i=1}^{n} p_i - \sum_{i=1}^{m} z_i$. Assuming the divergent branches extend from a central point α, the first two terms of binomial expansion give

$$(s-\alpha)^{n-m} \approx s^{n-m} - (n-m)\alpha s^{n-m-1},$$

and therefore

$$0 \approx 1 + K\frac{1}{(s-\alpha)^{n-m}},$$

so long as $\eta = (n-m)\alpha$. Thus,

$$\alpha = \frac{\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} z_i}{n-m}. \tag{17}$$

Note that at a point s_0 that is on a divergent branch of the locus such that $K = -\frac{1}{L(s_0)} \gg 1$, $|s_0|$ is very large, and therefore the poles and zeros appear to be clustered at a single point. Thus, $\angle(s_0 - p) = \angle(s_0 - z) = \phi_k$ for each pole and zero;

$$\angle L(s_0) = \sum_{i=1}^{m} \angle(s_0 - z_i) - \sum_{i=1}^{n} \angle(s_0 - p_i) = (m-n)\phi_k = \pi \pm 2\pi(k-1).$$

Note that $\pi \pm 2\pi(k-1)$ is equivalent to $-\pi \mp 2\pi(k-1)$. The later will be used for notational convenience. Therefore,

$$\phi_k = \frac{-\pi \mp 2\pi(k-1)}{m-n}.$$

Hence, the angles of the divergent branches approach the angles

$$\phi_k = \frac{\pi \pm 2\pi(k-1)}{n-m}, \quad k = 1, 2, \ldots n-m. \tag{18}$$

Rule #5: Angles of Departure and Arrival

Since $\angle L(s_0) = \pi \pm 2\pi(k-1)$ for every point s_0 on the locus,

$$\begin{aligned}\pi \pm 2\pi(k-1) &= \sum_{i=1}^{m} \angle(s_0 - z_i) - \sum_{i=1}^{n} \angle(s_0 - p_i) \\ &= \sum_{i=1}^{m} \angle(s_0 - z_i) - \sum_{\substack{i=1 \\ i\neq j}}^{n} \angle(s_0 - p_i) - q_j \angle(s_0 - p_j),\end{aligned}$$

where p_j is a pole of multiplicity q_j. Therefore, the angle of departure from the pole p_j may be determined by allowing $s_0 \to p_j$;

$$\gamma_j = \lim_{s_0 \to p_j} \angle(s_0 - p_j) = \frac{1}{q_j}\left(\sum_{i=1}^{m} \angle(p_j - z_i) - \sum_{\substack{i=1 \\ i\neq j}}^{n} \angle(p_j - p_i) - \pi \mp 2\pi(k-1)\right), \quad k \in \mathbb{Z}_{>0}. \tag{19}$$

Similarly, the angle of arrival at a zero z_j of multiplicity q_j is

$$\psi_j = \lim_{s_0 \to z_j} \angle(s_0 - z_j) = \frac{1}{q_j}\left(\sum_{i=1}^{n} \angle(z_j - p_i) - \sum_{\substack{i=1 \\ i\neq j}}^{m} \angle(z_j - z_i) + \pi \pm 2\pi(k-1)\right), \quad k \in \mathbb{Z}_{>0}. \tag{20}$$

Rule #6: Imaginary-Axis Crossings

Any imaginary-axis crossings $\tilde{s} = \pm j\omega_0$ can be found using Routh-Hurwitz criterion.

Rule #7: Break-In and Breakaway Points

At real-axis break-in and breakaway points, K will be at a minimum or maximum value for some real s. That is, break-in and breakaway points correspond to minimum and maximum values of K for the real-axis segments on the locus. Let $\mathbb{L}$ be the set of all points s on the locus. Then any roots $\hat{s}$ of

$$\left.\frac{dK}{ds}\right|_{s=\hat{s}} = 0, \tag{21}$$

such that $\hat{s} \in \mathbb{L} \cap \mathbb{R}$ are break-in or breakaway points.