Lab 3 Report

Problem Statement

This lab focuses on **discrete-time control system design** using techniques such as **root locus**, **loop shaping**, and **linear state feedback**. The behavior of a simplified plant system will be analyzed to determine its discretized equivalent, using controllers to help meet specified performance criteria. System requirements include zero steady-state error, minimizing settling times, and limiting percentage overshoot. In Problems 1 and 2, the focus is on determining the parameters and linearizing the system around a pseudo-equilibrium point, followed by selecting an appropriate sampling period and creating the associated discrete-time system for the linearized plant. Problems 3, 4, and 5 involve designing controllers using three different methods. Each controller design is evaluated for stability and performance against both the **linearized local model** and the **global nonlinear model**.

In this lab, the following constants will be used, in accordance with the laboratory instructions: $a = 7$, $b = 3$, $c = 4$.

Electrical and Electronic Engineering
University College Dublin

EEEN 40010
Control Theory

CT3
Digital Control

Declaration of Authorship

I declare that all material in this assessment is my own work except where there is clear acknowledgement and appropriate reference to the work of others.

Signed: Devon James Knox
Date: 1 December, 2024

Devon James Knox

Problem 1: Determine k, μ, and ν

Introduction

The problem asks to find k, μ, and ν such that $(e, i, \omega, \theta) = (2a{+}5\text{ V}, 20\text{A}, 180\text{ rpm}, \frac{\pi}{4}\text{ rad})$ is a pseudo-equilibrium point of the system. Defining this pseudo-equilibrium point as the operating point, the linearisation about this operating point will be determined. The pseudo-equilibrium point ensures the derivatives of state variables are zero, representing a balanced steady state. The following dynamics govern the system under steady-state conditions:

$$\begin{aligned} L\frac{di(t)}{dt} + Ri(t) + k_e\omega(t) &= e(t) \\ J\frac{d\omega(t)}{dt} &= T(t) - \mu\omega(t) \\ I\frac{d^2\theta(t)}{dt^2} &= F(t)\cos(\theta(t)) - mg\sin(\theta(t)) - \eta\frac{d\theta(t)}{dt} \end{aligned} \tag{1}$$

Constants Used for this problem

Here are the constants that are involved in this system:

Variable	Value
R	$0.7\,\Omega$
L	$0.0013\,\text{H}$
I	$0.073\,\text{kg}\cdot\text{m}^2$
J	$0.033\,\text{kg}\cdot\text{m}^2$
η	$0.403\,\text{kg}\cdot\text{m/s}$
g	$9.81\,\text{m/s}^2$
m	$0.734\,\text{kg}$

Pseudo-equilibrium point

Using the personal digits, the pseudo-equilibrium point of this system is as follows:

Parameter	Value
e	$19\,\mathrm{V}$
i	$20\,\mathrm{A}$
ω	$6\pi\,\mathrm{rad/s}$
θ	$\frac{\pi}{4}\,\mathrm{rad}$

Simplified Equations at Pseudo-Equilibrium Point

At the designated pseudo-equilibrium point, the derivatives of the state variables of the system must all be equal to zero. Thus, taking $\frac{di}{dt} = 0$, $\frac{d\omega}{dt} = 0$, and $\frac{d^2\theta}{dt^2} = 0$ (with $\frac{d\theta}{dt} = 0$), the equations in (1) can be simplified as follows:

- For the motor dynamics equation:

$$L\frac{di(t)}{dt} + Ri(t) + k_e\omega(t) = e(t)$$

At steady state, $\frac{di}{dt} = 0$, so the equation reduces to:

$$Ri + k_e\omega = e$$

Substituting the given values: $R = 0.7\,\Omega$, $i = 20\,\mathrm{A}$, $\omega = 6\pi\,\mathrm{rad/s}$, and $e = 19\,\mathrm{V}$:

$$0.7(20) + k_e(6\pi) = 19$$

Solving for k_e:

$$k_e = k = \frac{19 - 14}{6\pi} \approx 0.265\,\mathrm{Nm/A}$$

- For the angular velocity dynamics equation:

$$J\frac{d\omega(t)}{dt} = T(t) - \mu\omega(t)$$

At steady state, $\frac{d\omega}{dt} = 0$, so:

$$T = \mu\omega$$

Since $T = k_m i$ and $k_m = k_e$, substituting $k_m = 0.265\,\mathrm{Nm/A}$, $i = 20\,\mathrm{A}$, and $\omega = 6\pi\,\mathrm{rad/s}$:

$$T = 0.265(20) = 5.305\,\mathrm{Nm}$$

Substituting T into $T = \mu\omega$:

$$5.305 = \mu(6\pi)$$

Solving for μ:

$$\mu = \frac{5.305}{6\pi} \approx 0.281$$

- For the plate dynamics equation:

$$I\frac{d^2\theta(t)}{dt^2} = F(t)\cos(\theta(t)) - mg\sin(\theta(t)) - \eta\frac{d\theta(t)}{dt}$$

At steady state, $\frac{d^2\theta}{dt^2} = 0$ and $\frac{d\theta}{dt} = 0$, so:

$$F(t)\cos(\theta) = mg\sin(\theta)$$

Substituting $\theta = \frac{\pi}{4}$, $\cos(\frac{\pi}{4}) = \sin(\frac{\pi}{4}) = 0.7071$, $m = 0.734\,\text{kg}$, and $g = 9.81\,\text{m/s}^2$:

$$F(0.7071) = (0.734)(9.81)(0.7071)$$

$$F = 7.20054\,\text{N}$$

Since $F = \nu T$, substituting $T = 5.305\,\text{Nm}$:

$$7.20054 = \nu(5.305)$$

Solving for ν:

$$\nu = \frac{7.20054}{5.305} \approx 1.357$$

Therefore, $k = 0.265$ Nm/A, $\mu = 0.281$, and $\nu = 1.357$ N/Nm.

Linearisation

The motor dynamics and angular velocity equations are already linear, so the only equation that needs to be linearized about the pseudo-equilibrium operating point is the plate dynamics equation:

$$0.073\frac{d^2\theta}{dt^2} = 7.20054i(t)\cos(\theta(t)) - 7.20054\sin(\theta(t)) - 0.403\frac{d\theta}{dt}.$$

Using the following trigonometric expansions:

$$\cos(\theta_0 + \delta\theta) \approx \cos(\theta_0) - \sin(\theta_0)\delta\theta,$$

$$\sin(\theta_0 + \delta\theta) \approx \sin(\theta_0) + \cos(\theta_0)\delta\theta,$$

the equation can be rewritten as:

$$0.073\frac{d^2\theta}{dt^2} = 7.20054\big(i_0 + \delta i\big)\big(\cos(\theta_0) - \sin(\theta_0)\delta\theta\big) - 7.20054\big(\sin(\theta_0) + \cos(\theta_0)\delta\theta\big) - 0.403\frac{d(\delta\theta)}{dt}$$

Expanding this gives:

$$0.073\frac{d^2\theta}{dt^2} = 7.20054i_0\cos(\theta_0) - 7.20054i_0\sin(\theta_0)\delta\theta + 7.20054\delta i\cos(\theta_0) - 7.20054\sin(\theta_0) \\ -7.20054\cos(\theta_0)\delta\theta - 0.403\frac{d(\delta\theta)}{dt}$$

At equilibrium, the constant terms cancel each other out due to satisfying the balance of forces:

$$7.20054i_0\cos(\theta_0) = 7.20054\sin(\theta_0)$$

$$0.073\frac{d^2\theta}{dt^2} = -7.20054i_0\sin(\theta_0)\delta\theta + 7.20054\delta i\cos(\theta_0) - 7.20054\cos(\theta_0)\delta\theta - 0.403\frac{d(\delta\theta)}{dt}$$

Substitute the following equilibrium values: $i_0 = 20$, $\theta_0 = \frac{\pi}{4}$, and $\sin\left(\frac{\pi}{4}\right) = \cos\left(\frac{\pi}{4}\right) = 0.7071$:

$$0.073\frac{d^2\theta}{dt^2} = -7.20054(20)(0.7071)\delta\theta + 7.20054\delta i(0.7071) - 7.20054(0.7071)\delta\theta - 0.403\frac{d(\delta\theta)}{dt}$$

$$0.073\frac{d^2\theta}{dt^2} = -101.83\delta\theta + 5.092\delta i - 5.092\delta\theta - 0.403\frac{d(\delta\theta)}{dt}$$

$$0.073\frac{d^2\theta}{dt^2} = -106.92\delta\theta + 5.092\delta i - 0.403\frac{d(\delta\theta)}{dt}$$

Therefore, the linearised plate dynamics equation is:

$$0.073\frac{d^2\theta}{dt^2} = -106.92\theta(t) + 5.092i(t) - 0.403\frac{d\theta(t)}{dt} \tag{2}$$

Equations for linearisation about operating point

$$0.013\frac{di(t)}{dt} + 0.7i(t) + 0.265\omega(t) = e(t)$$

$$0.033\frac{d\omega(t)}{dt} = 0.265i(t) - 0.281\omega(t)$$

$$0.073\frac{d^2\theta}{dt^2} = -106.92\theta(t) + 5.092i(t) - 0.403\frac{d\theta(t)}{dt} \tag{3}$$

Problem 2

Introduction

This problem asks to find a sampling period so that there are at least 10 samples before the 2% settling time of the plant. From here, the associated discrete-time system for the linearized plant will be found, assuming a zero-order hold circuit is used.

Sampling Period

First, the following state variables are defined:

$$x_1 = i(t), x_2 = \omega(t), x_3 = \theta(t), x_4 = \dot{\theta}(t)$$

From here, the linearised equations from equation (3) become:

$$\begin{aligned} 0.013\dot{x}_1 + 0.7x_1 + 0.265x_2 &= e(t) \\ 0.033\dot{x}_2 &= 0.265x_1 - 0.281x_2 \\ 0.073\dot{x}_4 &= -106.92x_3 + 5.092x_1 - 0.403x_4 \end{aligned} \tag{4}$$

Equation (4) can now be represented in state-space form:

$$\dot{x} = Ax + Be(t),$$

where:

$$A = \begin{bmatrix} -53.8 & -20.4 & 0 & 0 \\ 8.0 & -8.5 & 0 & 0 \\ 0 & 0 & 0 & 0.001 \\ 69.8 & 0 & -1464.7 & -0.0055 \end{bmatrix}, \quad B = \begin{bmatrix} 76.923 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The output equation is

$$y = Cx$$

where

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \end{bmatrix}.$$

Using this representation, the step response can be found in MATLAB:

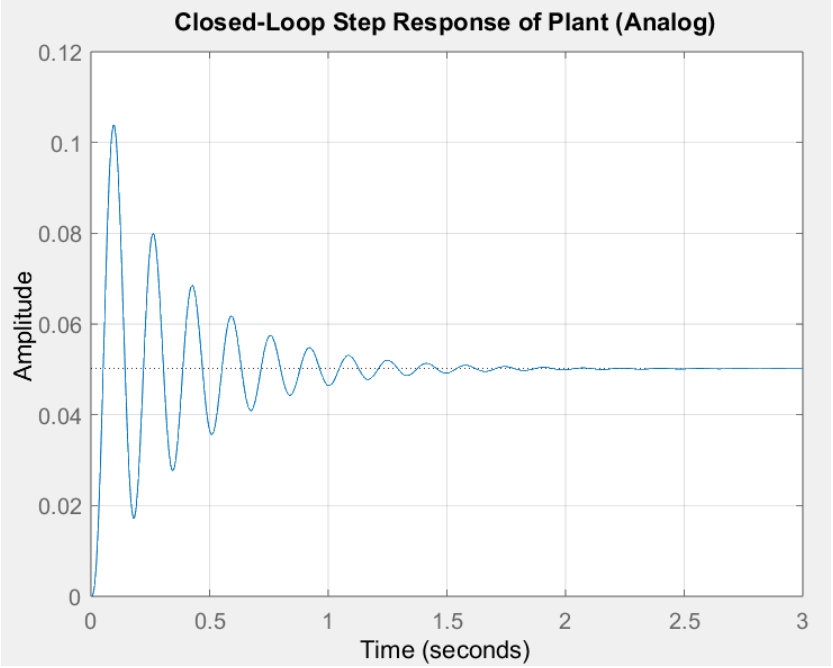

Figure 1: Closed-loop step response information of linearized plant (Analog)

```
Step Response Information:
          RiseTime: 0.0293
     TransientTime: 1.4262
      SettlingTime: 1.4295
       SettlingMin: 0.0172
       SettlingMax: 0.1039
         Overshoot: 107.1848
        Undershoot: 0
              Peak: 0.1039
          PeakTime: 0.0960
```

Figure 2: Closed-loop step response information of linearized plant (Analog)

From the MATLAB output above, it is observed that the settling time of the linearised plant is 1.43 seconds. To find the appropriate sampling period, this value must be divided by at least 10 (which is 0.143), however, to ensure that the discrete-time representation accurately captures the dynamics of the system, **a smaller sampling period of 0.05 seconds is selected**. This helps reduce aliasing and improves the fidelity of the associated discrete-time model.

Associated Discrete-Time System

Discrete-Time System

The following state-space matrices represent the discrete-time system for the linearized plant, obtained using a zero-order hold circuit with a sampling period of $T = 0.05$ seconds (see Code section for each problem at the bottom of this document):

$$A_d = \begin{bmatrix} 0.0346 & -0.2470 & 0 & 0 \\ 0.0973 & 0.5840 & 0 & 0 \\ 0.0241 & -0.0110 & -0.2292 & 0.0215 \\ 0.1133 & -0.3985 & -31.5358 & -0.3481 \end{bmatrix}, \quad B_d = \begin{bmatrix} 1.2616 \\ 0.3106 \\ 0.0471 \\ 1.8575 \end{bmatrix}$$

The output equation for the system is given by:

$$C_d = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}, \quad D_d = \begin{bmatrix} 0 \end{bmatrix}$$

The discrete-time representation of the system can be expressed as:

$$x[k+1] = A_d x[k] + B_d u[k],$$

$$y[k] = C_d x[k] + D_d u[k].$$

Here is the step response of the discrete-time system for the given sampling period:

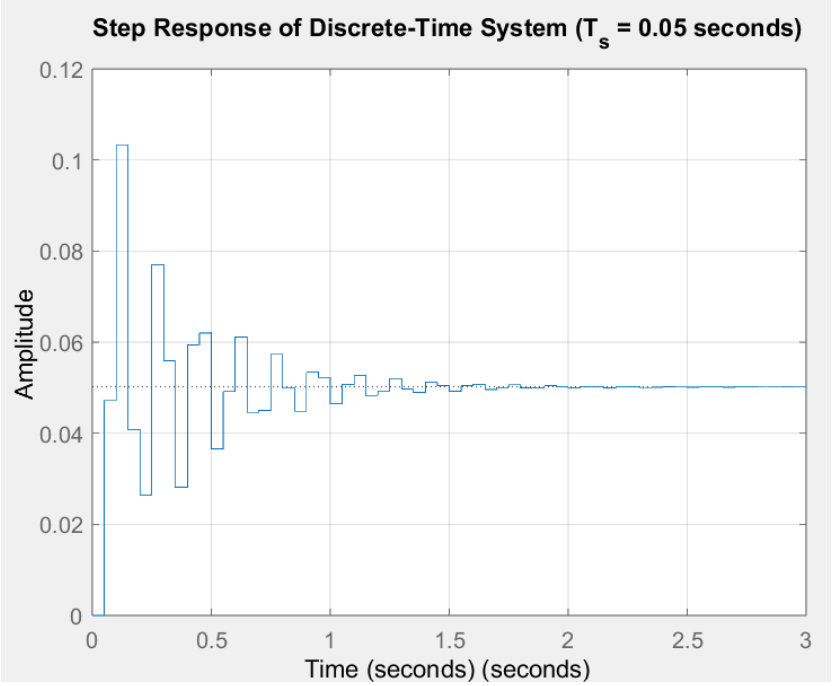

Figure 3: Step Response of the Discrete-Time System

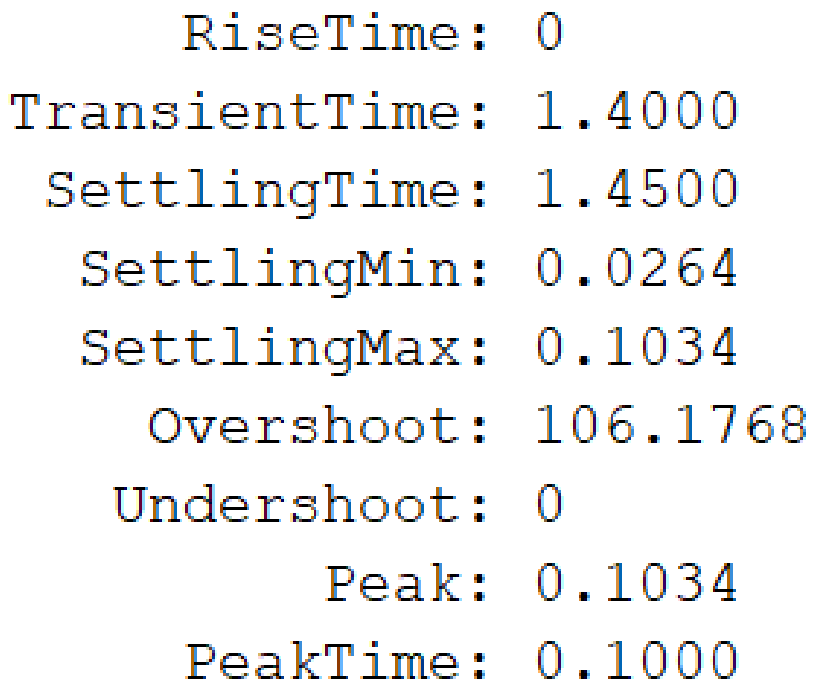

Figure 4: Step Response Characteristics of the Discrete-Time System

Problem 3

Introduction

Problem 3 asks to design a discrete-time linear state feedback controller with observer to achieve a zero steady-state error to a step input, a 2% settling time not exceeding 60% of that of the plant, and a percentage overshoot not more than 40% of that of the plant (or 25%, whichever is larger). Then, the behaviour of the resulting closed loop system using both the linearised local model and the original global model will be checked.

Convert Specification to Desired Poles

To achieve the design objectives, the following specifications are converted into pole locations for the closed-loop system.

Settling Time

The settling time of the original plant is $t_s(2\%) = 1.4295\,\text{s}$. The closed-loop system must have a 2% settling time not exceeding 60% of the plant's settling time:

$$t_{s,\text{desired}}(2\%) = 0.6 \times 1.4295 = 0.8577\,\text{s}.$$

To account for error, the desired 2% settling time must not exceed 0.85 seconds.

Steady-State Error

To achieve zero steady-state error to a step input, a pre-amplifier gain will be included in the controller design.

Percent Overshoot

The overshoot of the original plant is approximately 107.18%. The closed-loop system's overshoot must not exceed:

$$\max(0.4 \times 107.18, 25) = 42.87\%.$$

Dominant Poles

Using the following equation, the range of values for the dominant pole can be found:

$$\begin{aligned} t_s(2\%) = \frac{4}{|\sigma|} \leq 0.85, \quad \text{where } \sigma = \zeta\omega_n \\ 0.85 \geq \frac{4}{|\sigma|} \\ 4.707 \leq |\sigma| \end{aligned} \tag{5}$$

Taking σ to be negative (since that is required for stability), $\sigma \leq -4.707$

Damping Ratio

The following equation relates percentage overshoot (PO) and damping ratio (ζ):

$$PO = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \cdot 100 \tag{6}$$

Now, the the damping ratio can be determined:

$$e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \cdot 100 \leq 42.87$$

$$\frac{-\pi\zeta}{\sqrt{1-\zeta^2}} \leq \ln\left(\frac{42.87}{100}\right)$$

$$\frac{-\pi\zeta}{\sqrt{1-\zeta^2}} \leq \ln(0.4287)$$

$$\zeta \geq 0.416.$$

Thus, to account for error, let $\zeta = 0.42$. With sampling period $T = 0.05\,\text{s}$ and the desired real part of the poles in the s-domain ($\sigma = -4.707$), the dominant pole in the z-domain is determined as:

$$z_{\text{dominant}} = e^{\sigma T}.$$

Therefore:

$$z_{\text{dominant}} = e^{-4.707 \times 0.05} = 0.79$$

For simplicity, let $z_{\text{dominant}} = 0.8$

Design Linear State Feedback Controller

To meet the performance specifications, the closed-loop poles were chosen to ensure stability, appropriate settling time, and reduced overshoot. After analysis, the following poles were selected so that a single, positive dominant pole (close to $z_{dominant} = 0.8$) to reduce oscillations will be followed by smaller positive poles (placed farther inside the unit circle to reduce influence on transient response):

$$p = [0.6, 0.3, 0.2, 0.1]$$

The state feedback gain K was then computed from the poles using MATLAB's *place* command:

$$K = \begin{bmatrix} -0.1132 & -0.1992 & -9.5825 & -0.2705 \end{bmatrix}$$

This feedback gain ensures that the desired settling time of 0.85 seconds and percentage overshoot of under 42.87% are both met.

Pre-Amplifier

To achieve zero steady-state error for a step input, a pre-amplifier gain $\bar{N}$ was calculated, which allows the system output to track the reference input without error. Furthermore, this pre-amplifier gain compensates for the effect of the feedback gain K, ensuring that both steady and transient state responses are sufficient. The pre-amplifier gain was computed with the formula:

$$\bar{N} = \frac{1}{C\big(I - A + BK\big)^{-1}B} = 4.0450$$

Design Observer

An observer was designed to reconstruct the state variables required for the feedback controller. To avoid interference with the system, the observer poles were chosen to be faster than the slowest closed-loop pole. The selected observer poles were:

$$p_{\text{observer}} = [0.4, 0.35, 0.25, 0.2]$$

Using the equation

$$A_{observer} = A - LC$$

where $A_{observer}$ consists of eigenvalues that are equal to $p_{observer}$, the observer gain L was found using MATLAB's *place* command:

$$L = \begin{bmatrix} -0.1509 \\ -0.1323 \\ -1.1584 \\ -10.6763 \end{bmatrix}$$

This gain ensures the observer converges quickly to the final state values.

Check Behavior

The step response of the closed-loop system was analyzed for both the linearized local model and the original global model.

Linearized Local Model

After the closed-loop system was simulated, the poles were as follows: Using the MATLAB *pole* function, the poles of the closed-loop system with the added linear state feedback controller are as follows:

$$\text{Poles: } \{0.6, 0.1, 0.2, 0.3, 0.2, 0.25, 0.35, 0.4\}$$

This aligns with the decision to have a single, positive dominant pole in addition to smaller positive ones in order to reduce oscillations during the step response. **Because all of these poles are within the unit circle on the complex plane, the system is stable.**
The linearized local model represents the system behavior near the operating point. The step response of the closed-loop system with the designed controller is shown below:

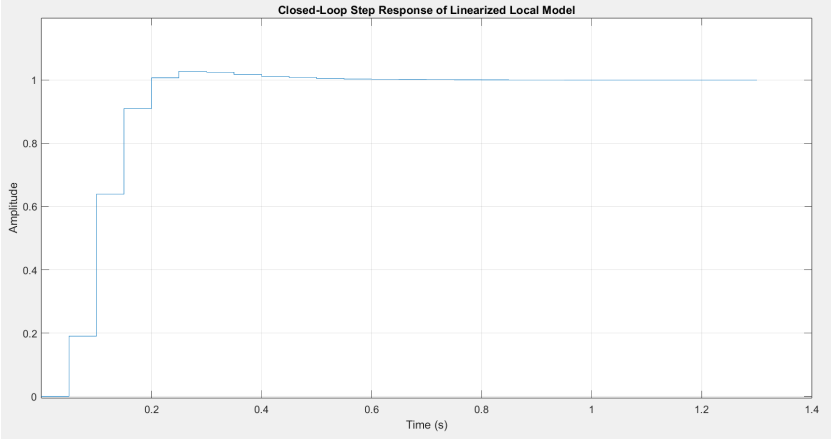

Figure 5: Closed-Loop Step Response of the Linearized Local Model with Linear State Feedback Controller

```
          RiseTime: 0.1000
    TransientTime: 0.3500
     SettlingTime: 0.3500
      SettlingMin: 0.9098
      SettlingMax: 1.0275
        Overshoot: 2.7522
       Undershoot: 0
             Peak: 1.0275
         PeakTime: 0.2500
```

Figure 6: Closed-Loop Step Response Information of the Linearized Local Model with Linear State Feedback Controller

The 2% settling time is equal to 0.35 seconds, which does not exceed 60% of the settling time of the plant, which is 0.85 seconds. Furthermore, the

steady-state error to a step input is zero, as the DC gain of the closed-loop system is approximately 1 (found using MATLAB's *dcgain(tf)* command). Lastly, the percent overshoot is 2.75%, which does not exceed the 42.87% specification. **Therefore, when using the linearised local model, the closed-loop system with the added linear state feedback controller meets all of the design specifications.**

Original Global Model

The original global model incorporates the system's nonlinearities. The global model incorporates nonlinear dynamics with the use of the added cubic term ϵx^3 where $\epsilon = 0.1$. The step response of the nonlinear global model was simulated using MATLAB's `ode45` solver (see Original Global Model code, *global_model.m*, in the Code section at the bottom of this document for full system of equations used). The resulting behavior is shown below:

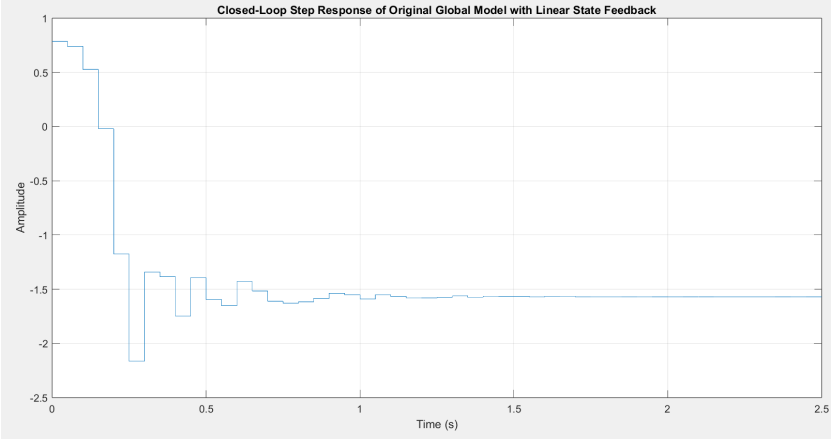

Figure 7: Closed-Loop Step Response of the Original Global Model with Linear State Feedback Controller

```
Step Response Characteristics (Global Model):
         RiseTime: 0.0561
    TransientTime: 0.7938
     SettlingTime: 0.9059
      SettlingMin: -2.1646
      SettlingMax: -1.3432
        Overshoot: 37.7973
       Undershoot: 49.9988
             Peak: 2.1646
         PeakTime: 0.2500
```

Figure 8: Closed-Loop Step Response Information of the Original Global Model with Linear State Feedback Controller

The 2% settling time is equal to 0.9059 seconds, which slightly exceeds 60% of the settling time of the plant, which is 0.85 seconds. Furthermore, the steady-state error to a step input is not zero, as the DC gain of the closed-loop system is approximately -1.57 (found using MATLAB's *dcgain(tf)* command). Lastly, the percent overshoot is 37.80%, which does not exceed the 42.87% specification.

Therefore, when using the original global model, the closed-loop system with the added linear state feedback controller meets most of the design specifications. Because the linear state feedback controller helps the closed-loop system meet some, but not all of the design specifications when the system is tested using the global model, the system is somewhat robust, similar to the PID and lead controllers (discussed in problems 4 and 5).

Problem 4

Introduction

Problem 4 asks to design a discrete-time PID/lead/lag controller using the root locus method to meet the same specifications as in problem 3. Then the behavior of the resulting closed loop system will be checked using both the linearised local model and the original global model.

Convert Specification to Desired Dominant Pole(s)

First, since the specifications are the same as those in problem 3, with the damping ratio $\zeta = 0.42$ and $z_{dominant} = 0.8$ (so the dominant pole will be placed near here). See this section in problem 3 to see how these values were determined.

Design Controller Using Root Locus

PI Controller

The following transfer function represents a PI controller in the z-domain:

$$G_c(z) = k_p + k_i \frac{z}{z-1},$$

with k denoting the respective proportional and integral gains. To simplify the design process, this transfer function can be viewed as:

$$G_c(z) = k \frac{z-r}{z-1},$$

where

$$r = \frac{k_p}{k_p + k_i}, \quad k = k_p + k_i$$

This reformulation helps to easily shape the root locus and place the closed-loop poles in the desired region. The objective is to:

- Place the dominant poles near $z_{\text{dominant}} = 0.8$ for the required settling time and damping ratio.
- Ensure the root locus branches have a damping ratio $\zeta \geq 0.42$ to limit the overshoot to $\leq 42.87\%$.

- Guarantee zero steady-state error to a step input using the integral action.

After analyzing the root locus of the system with the PI controller, it was found that a PI controller simply cannot meet the design specifications. This is because the dominant poles could not be placed in the desired region of the z-domain (with a damping ratio $\zeta \geq 0.42$ and magnitude ≤ 0.8) without causing other poles to move into unstable or undesirable regions. Thus, a PID controller must be employed to allow the system to meet the desired closed-loop specifications.

PID Controller

A PID controller was designed to provide additional flexibility in shaping the root locus. The derivative term introduces a zero in the z-domain that helps stabilize the system and improve transient response by increasing the damping ratio of the poles. The PID controller transfer function in the z-domain is given by:

$$G_c(z) = k_p + k_i \frac{z}{z-1} + k_d(z-1),$$

where k_d is the derivative gain. Placing the dominant pole near 0.8 to satisfy the damping ratio of 0.42, the gain k was then interactively chosen using the *rlocfind* command in MATLAB.

Root Locus

Here is the root locus of the plant when the PID controller is added:

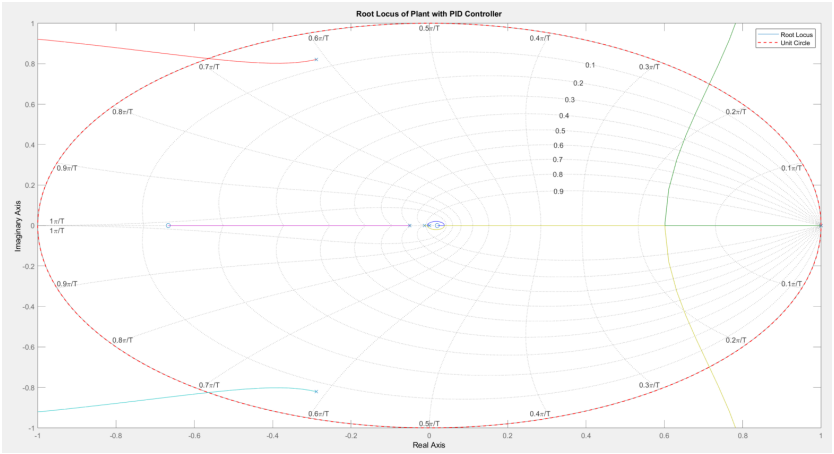

Figure 9: Root locus of plant with added PID controller (data points for selecting gain not shown)

With the selected gain from the root locus being $k = 1.85$, the PID controller parameters could be found:

$$k_p = 1.25, k_i = 0.5, k_d = 0.1$$

Therefore, the PID controller takes on the following form:

$$G_c(z) = 1.25 + 0.5\frac{z}{z-1} + 0.1(z-1) \tag{7}$$

Check Behavior

After the closed-loop system was simulated, the poles were as follows: Using the MATLAB *pole* function, the poles of the closed-loop system with the added PID controller are as follows:

Poles: $\{-0.3360+0.8064i, -0.3360-0.8064i, 0.6233+0.2744i, 0.6233-0.2744i, -0.2630, 0.0204\}$

Because all of these poles are within the unit circle on the complex plane, the system is stable.

Now, the behavior of the resulting closed-loop system with the added PID controller will be checked, using both the linearised local model and original global model. The global model incorporates nonlinear dynamics with the use of the cubic term ϵx^3, where $\epsilon = 0.1$.

0.1 Linearized Local Model

The following table and plot show the closed-loop step response for the linearized local model:

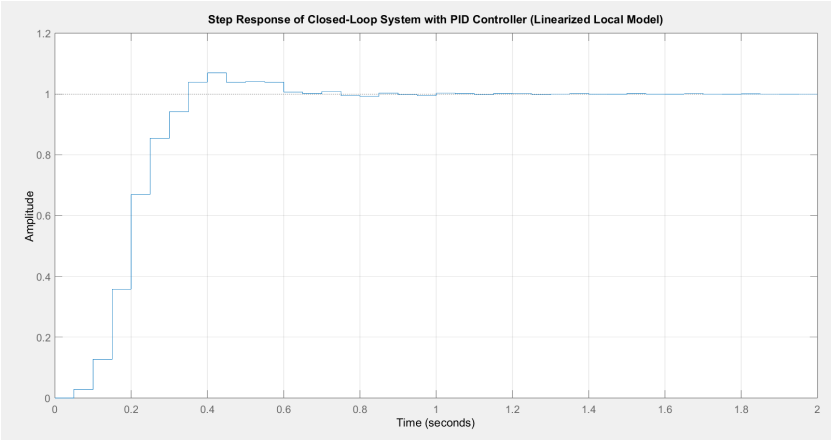

Figure 10: Closed-loop step response for linearized local model with added PID controller

```
Step Response Characteristics (Linearized Local Model):
         RiseTime: 0.2000
    TransientTime: 0.6000
     SettlingTime: 0.6000
      SettlingMin: 0.9410
      SettlingMax: 1.0693
        Overshoot: 6.9277
       Undershoot: 0
             Peak: 1.0693
         PeakTime: 0.4000
```

Figure 11: Closed-loop step response information for linearized local model with added PID controller

The 2% settling time is equal to 0.6 seconds, which does not exceed 60% of the settling time of the plant, which is 0.85 seconds. Furthermore, the steady-state error to a step input is zero, as the DC gain of the closed-loop system is 1 (found using MATLAB's *dcgain(tf)* command). Lastly, the percent overshoot is 6.93%, which does not exceed the 42.87% specification.

Therefore, when using the linearised local model, the closed-loop system with the added PID controller meets all of the design specifications.

0.2 Original Global Model

The following table and plot show the closed-loop step response for the original global model:

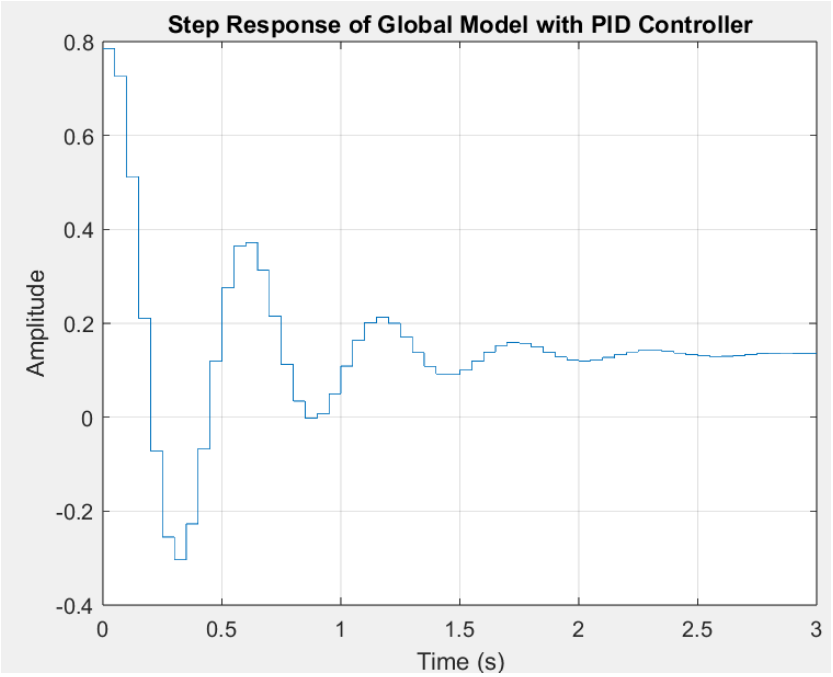

Figure 12: Closed-loop step response for original global model with added PID controller

```
Step Response Characteristics (Global Model):
         RiseTime: 0
    TransientTime: 2.0303
     SettlingTime: 2.6530
      SettlingMin: -0.3038
      SettlingMax: 0.7854
        Overshoot: 485.2984
       Undershoot: 226.4269
             Peak: 0.7854
         PeakTime: 0
```

Figure 13: Closed-loop step response information for original global model with added PID controller

Compared to the step response of the linearised model, the global model's step response displays much more oscillatory behavior, which is likely due to the presence of nonlinear dynamics (namely the ϵx^3 term). This response reflects the complexities that nonlinear systems present, as operating around equilibrium or pseudo-equilibrium points allow for a more stable and predictable response.
The 2% settling time is equal to 2.6530 seconds, which far exceeds 60% of the settling time of the plant, which is 0.85 seconds. Furthermore, the steady-state error to a step input is not zero, as the DC gain of the closed-loop system is approximately 0.135 (found using MATLAB's *dcgain(tf)* command). Lastly, the percent overshoot is 485.30%, which greatly exceeded the 42.87% specification.

Therefore, when using the original global model, the closed-loop system with the added PID controller (determined using the root locus method) doesn't do a satisfactory job of meeting the design specifications. Therefore, when the system is tested using the global model, the system with the added PID controller is not very robust.

Problem 5

Introduction

Problem 5 asks to design a discrete-time PID/lead/lag controller using loop shaping to meet the same specifications as in problem 3. Then the behavior of the resulting closed loop system will be checked using both the linearised local model and the original global model.

Convert Specifications to Desired Frequency Response

- **Damping ratio and phase margin**
  Since the damping ratio $\zeta \geq 0.42$, using the following phase margin approximation:
  $$PM \cong 100 \times \zeta \tag{8}$$
  The required phase margin is $PM = 100 \times 0.42 = 42°$; however, for a robust safety margin, let $PM_{des} \geq 60°$.

- **Gain crossover frequency**
  Using equation (5) which relates 2% settling time to the damping coefficient ζ and natural frequency ω_n, the natural frequency can be found with some algebraic manipulation; an inequality is expressed to account for the 2% settling time not surpassing 0.85 seconds (as calculated earlier):
  $$\begin{aligned} t_s(2\%) \cong \frac{4}{\zeta \cdot \omega_n} &\leq 0.85 \\ \omega_n &\geq \frac{4}{0.85 \cdot \zeta} \\ \omega_n &\geq 11.10 \frac{\text{rad}}{\text{s}} \end{aligned}$$
  The following equation is used to approximate gain crossover frequency ω_{gc}:
  $$\omega_{gc} \approx \omega_n \sqrt{\sqrt{1 + 4 \cdot \zeta^4} - 2 \cdot \zeta^2} \tag{9}$$
  Using the above equation and the corresponding values for ζ and ω_n:

  $\omega_{gc} \cong 9.34 \dfrac{\text{rad}}{\text{s}} \Rightarrow$ after various attempts such as $\omega_{gc,des} = 10, 15, 20, 30 \ \dfrac{rad}{s}$,

  let $\omega_{gc,des} = 39.7 \ \frac{rad}{s}$.

- **Gain crossover frequency in discrete time**
  With a sampling time of $T = 0.005$ seconds (this was altered from the sampling rate of $T = 0.05$ seconds used prior, as dealing with loop shaping requires a higher sampling rate to accurately capture and analyze the high-frequency dynamics of the system), the discretized gain crossover frequency is: $\Omega_{gc,des} = \omega_{gc,des} \cdot T = 39.7 * 0.005 = 0.1985 \frac{\text{rad}}{\text{sample}}$
  Now, the bode plot of the plant will be plotted in terms of ω (not Ω even though the system is concerned with discrete time), and the phase and gain will be determined at the desired gain crossover frequency of $\omega_{gc} = 39.7 \frac{\text{rad}}{\text{sec}}$

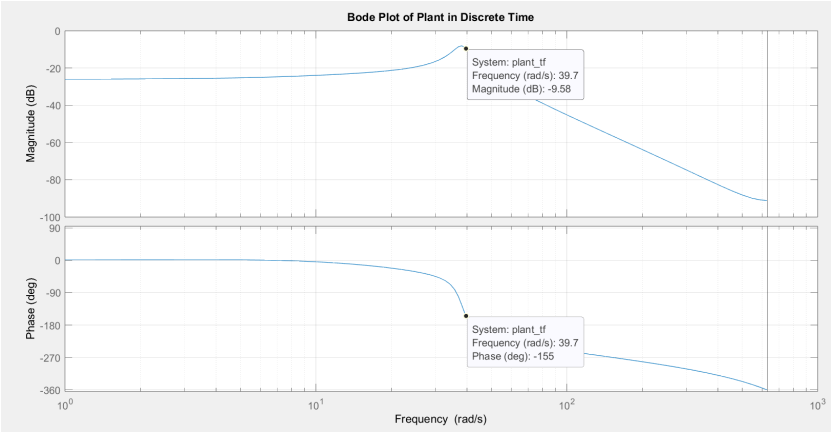

Figure 14: Bode plot of plant in discrete time

The above graph shows that at the specified gain crossover frequency, the magnitude and phase is -9.58 dB and -155°, respectively. Now, the controller can be designed using loop shaping.

Design Controller Using Loop Shaping

The following equations can find the gain of the controller and the phase of the controller at the crossover frequency:

Magnitude Condition

$$20 \log_{10} \left(\left| G_c \left(e^{j39.7T} \right) \right| \right) + 20 \log_{10} (|G_{pd}(e^{j39.7T})|) = 0 \text{ dB}$$

$$20 \log_{10} \left(\left| G_c \left(e^{j39.7T} \right) \right| \right) - 9.85 = 0 \text{ dB}$$

$$20\log_{10}\left(\left|G_c\left(e^{j39.7T}\right)\right|\right) = 9.85\,\text{dB}.$$

Phase Margin Equation

$$\text{Arg}\left(G_c\left(e^{j39.7T}\right)\right) + \text{Arg}\left(G_{pd}\left(e^{j39.7T}\right)\right) \geq -180^{^\circ} + 60^{^\circ}$$

$$\text{Arg}\left(G_c\left(e^{j39.7T}\right)\right) - 155^{^\circ} \geq -180^{^\circ} + 60^{^\circ},$$

$$\text{Arg}\left(G_c\left(e^{j39.7T}\right)\right) \geq 35^{^\circ}$$

Therefore, the desired phase of the controller is set to $35^{^\circ} = 0.61$ rad. Using the following transfer function for the controller, gain k and zero location r can be determined:

$$G_c(z) = k(z - r)$$

At the Gain Crossover Frequency:

$$\frac{sin(0.1985)}{cos(0.1985) - r} \geq \tan(0.61)$$

$$\frac{0.197}{0.980 - r} \geq \tan(0.61)$$

$$0.197 \geq (0.980 - r) \cdot \tan(0.61)$$

$$\frac{0.197}{\tan(0.61)} \geq 0.980 - r$$

$r \geq 0.980 - \dfrac{0.197}{\tan(0.61)} \Rightarrow$ after testing multiple r values according to this inequality, the final value is $r = 0.25$.

To Find k, the magnitude condition is employed:

$$20\log_{10}(k|e^{j0.1985} - 0.25|) = 9.58\ \text{dB}$$

$$20\log_{10}(k) + 20\log_{10}(|e^{j0.1985} - 0.25|) = 9.58\ \text{dB}$$

$$20\log_{10}(k) + 20\log_{10}(0.76) = 9.58\ \text{dB}$$

$$20\log_{10}(k) = 3.01 - 20\log_{10}(0.76)$$

$$\log_{10}(k) = \frac{3.01 - 20\log_{10}(0.76)}{20}$$

$$k = 10^{\frac{3.01-20\log_{10}(0.76)}{20}}$$

$$k = 2.5$$

Thus, $k = 2.5$ and $r = 0.25$, so therefore the controller employed using the loop shaping technique is as follows:

$$G_c(z) = 2.5(z - 0.25)$$

Check Behavior

With the discrete-time lead controller, the zero at $r = 0.25$ creates a positive phase shift for frequencies near ω_{gc} (and thus Ω_{gc}), which in turn increases phase margin, stability, and damping ratio to help reach the performance requirements. Now, the behavior of the resulting closed-loop system with the added lead controller will be checked, using both the linearised local model and original global model. The global model incorporates nonlinear dynamics with the use of the cubic term ϵx^3, where $\epsilon = 0.1$.

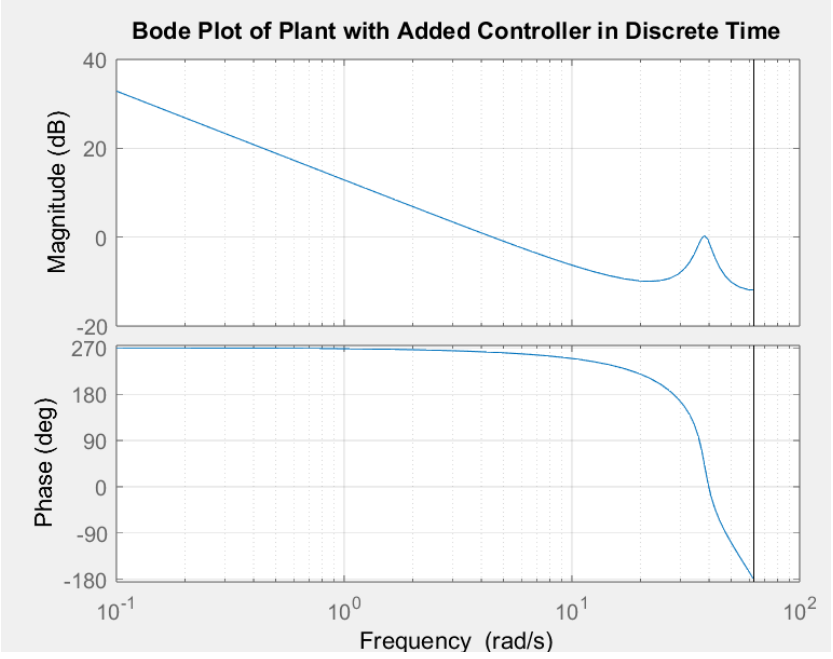

Figure 15: Bode plot of plant with added controller in discrete time

From the bode plot, it is evident that the controller improves performance at low-frequency inputs; furthermore, the gain crossover frequency aligns with the desired $\omega_{gc} = 39.7\frac{rad}{sec}$, indicating an appropriately fast response. In addition, the downward slope of the phase indicates that the system is stable and there is minimal lag due to the added controller.
Using the MATLAB *pole* function, the poles of the closed-loop system with the added lead controller are as follows:

$$\text{Poles: } \{0.7766, -0.0903 + 0.7875i, -0.0903 - 0.7875i, -0.3640, 0.0107\}$$

Because all of these poles are within the unit circle on the complex plane, the system is stable.

0.3 Linearized Local Model

The following table and plot show the closed-loop step response for the linearized local model:

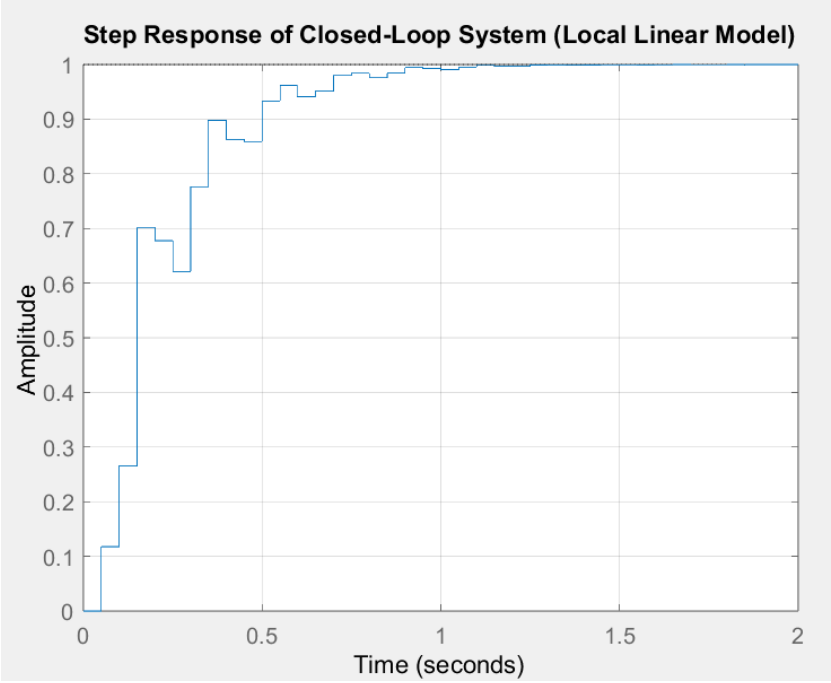

Figure 16: Closed-loop step response for linearized local model with added controller

```
Step Response Characteristics (Linearized Local Model):
         RiseTime: 0.4500
    TransientTime: 0.8500
     SettlingTime: 0.8500
      SettlingMin: 0.9333
      SettlingMax: 1.0000
        Overshoot: 2.5371e-06
       Undershoot: 0
             Peak: 1.0000
         PeakTime: 3.1500
```

Figure 17: Closed-loop step response information for linearized local model with added controller

The 2% settling time is equal to 0.85 seconds, which is exactly 60% of the settling time of the plant (technically 0.8577 seconds, but was set to 0.85 seconds to add a safety margin). Furthermore, the steady-state error to a step input is zero, as the DC gain of the closed-loop system is 1 (found using MATLAB's *dcgain(tf)* command). Lastly, the percent overshoot is approximately 0, which does not exceed the 42.87% specification.

Therefore, when using the linearised local model, the closed-loop system with the added controller meets all of the design specifications.

0.4 Original Global Model

The following table and plot show the closed-loop step response for the original global model:

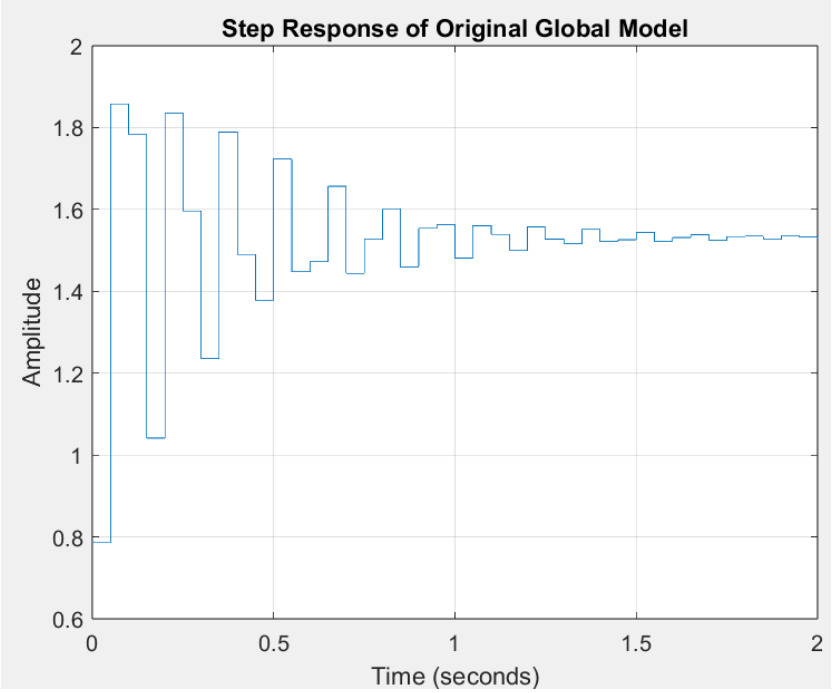

Figure 18: Closed-loop step response for original global model with added controller

```
Step Response Characteristics (Global Model):
         RiseTime: 0.0275
    TransientTime: 1.5020
     SettlingTime: 1.0532
      SettlingMin: 1.0414
      SettlingMax: 1.8582
        Overshoot: 21.5684
       Undershoot: 0
             Peak: 1.8582
         PeakTime: 0.0500
```

Figure 19: Closed-loop step response information for original global model with added controller

The 2% settling time is equal to 1.0532 seconds, which exceeds 60% of the settling time of the plant, which is 0.8577 seconds; however this is still very close to reaching it. Furthermore, the steady-state error to a step input is not zero, as the DC gain of the closed-loop system is approximately 1.55

(found using MATLAB's *dcgain(tf)* command). Lastly, the percent overshoot is 21.57%, which does not exceed the 42.87% specification.

Therefore, when using the original global model, the closed-loop system with the added controller meets most of the design specifications. Because this lead controller (found by employing loop shaping) meets some, but not all of the design specifications when the closed-loop system is tested using the global model, the system is not very robust.

Conclusion

In this report, three different controllers were employed and tested against both the linearised local model and the original global model: the linear state feedback controller (Problem 3), the PID controller (Problem 4), and the lead controller designed using loop shaping (Problem 5). All three controllers achieved the desired performance specifications in the linearized local model.

However, none of the three controllers achieved all of the performance specifications in the global model. Among the three, the linear state feedback controller demonstrated the best performance and was relatively robust. The lead controller that was employed via loop shaping also performed well and was a close second in terms of meeting the specifications.

The nonlinear dynamics made the global model behavior significantly more complicated. Unlike the linearized local model, the original global model captures real-world nonlinearities, such as the effects of damping and external forces. These complexities made achieving the desired performance specifications much more challenging, as the controllers that were designed based on linear assumptions struggled to properly adapt. This highlights the importance of considering nonlinear dynamics during the design phase for controllers that will be used in real-world applications.

Code

Original Global Model (*global_model.m*)

This function (used in problems 3, 4, and 5) was used to help simulate the closed-loop step response of the original global model with each controller, which was compared to the response of the linearised local model.

```
function dxdt = global_model(t, x, u, k, mu, nu, m, g, eta, R, L, J, I)
% State variables: x(1)=i, x(2)=w, x(3)=theta, x(4)=theta_dot

i = x(1);
w = x(2);
theta = x(3);
theta_dot = x(4);

di = (u - R*i - k*w)/L; % L di/dt + R i + k w = e(t) = u
dw = (k*i - mu*w)/J; % J dw/dt = k i - mu w
dtheta = theta_dot; % dtheta/dt = theta_dot

% I d^2theta/dt^2 = nu*k*i*cos(theta) - m*g*sin(theta) - eta*theta_dot
dtheta_dot = (nu*k*i*cos(theta) - m*g*sin(theta) - eta*theta_dot)/I;

dxdt = [di; dw; dtheta; dtheta_dot];
end
```

Figure 20: Code for Original Global Model

Problem 2

```
% 0.013 x1_dot + 0.7 x1 + 0.265 x2 = e(t)
A11 = 0.7 / 0.013;
A12 = 0.265 / 0.013;
B1 = 1 / 0.013; % Coefficient for e(t)

% 0.033 x2_dot = 0.265 x1 - 0.281 x2
A21 = 0.265 / 0.033;
A22 = -0.281 / 0.033;

% 0.073 x4_dot = -106.92 x3 + 5.092 x1 - 0.403 x4
A41 = 5.092 / 0.073;
A43 = -106.92 / 0.073;
A44 = -0.403 / 0.073;

% State-space matrices
A = [-A11, -A12, 0, 0; A21, A22, 0, 0; 0, 0, 0, 1;
   A41, 0, A43, A44];
B = [B1; 0; 0; 0];
C = [0, 0, 1, 0];
D = [0];

% State-space model
sys = ss(A, B, C, D);

% Compute discrete-time model with Ts
Ts = 0.05; % Sampling period
sysd = c2d(sys, Ts, 'zoh'); % Zero-order hold
```

Figure 21: Code for finding linearised state-space model of plant and simulating closed-loop step response (additional code for plotting and formatting output not shown)

Problem 3

```
% Pseudo-equilibrium point
e_eq = 19; % Voltage
i_eq = 20; % Current
omega_eq = 6 * pi; % Angular velocity
theta_eq = pi / 4; % Angular position

% Linearized state-space matrices
A = [-53.846, -20.385, 0, 0;
      8.030, -8.515, 0, 0;
      0,      0,      0, 1;
     69.726,  0, -1464.38, -5.521];
B = [76.923; 0; 0; 0];
C = [0, 0, 1, 0];
D = 0;
Ts = 0.05; % Sampling time

% Convert to discrete-time
sys = ss(A, B, C, D);
sysd = c2d(sys, Ts, 'zoh');
Ad = sysd.A;
Bd = sysd.B;
Cd = sysd.C;
Dd = sysd.D;

% Feedback and Observer Design
poles_feedback = [0.6, 0.3, 0.2, 0.1];
K = place(Ad, Bd, poles_feedback);
N_bar = 1 / (Cd * ((eye(size(Ad)) - Ad + Bd * K) \
   Bd));
poles_observer = [0.4, 0.35, 0.25, 0.2];
L = place(Ad', Cd', poles_observer)';

% Linearized Local Model Simulation
Acl_local = [Ad - Bd * K, Bd * K; zeros(size(Ad)),
   Ad - L * Cd];
Bcl_local = [Bd * N_bar; zeros(size(Bd))];
Ccl_local = [Cd, zeros(size(Cd))];
Dcl_local = Dd;
sys_cl_local = ss(Acl_local, Bcl_local, Ccl_local,
   Dcl_local, Ts);
t_local = 0:Ts:1.3;
[y_cl_local, t_cl_local] = step(sys_cl_local,
   t_local);
```


Figure 22: Code for Linear State Feedback Design (code for plotting and formatting linearised local model not included)

```
% Global model parameters already in workspace
epsilon = 0.1; % Nonlinear term coefficient

% Initial conditions
x0_global = [i_eq; omega_eq; theta_eq; 0]; % Pseudo-
    equilibrium as initial state
r_global = 1; % Unit step input
t_global = 0:Ts:2.5; % Time
x_global = zeros(length(t_global), 4); % Initialize
    state matrix
x_global(1, :) = x0_global; % Initial state

% Simulate nonlinear model for each sampling
    interval
for n = 1:length(t_global) - 1
    u_global = N_bar * (r_global - K * (x_global(n,
        :)' - [i_eq; omega_eq; theta_eq; 0]));
    [~, x_temp] = ode45(@(t, x) global_model(t, x,
        u_global, k_val, mu, nu, m_val, g_val,
        eta_val, R_val, L_val, J_val, I_val) ...
         - epsilon * x(3)^3, [t_global(n), t_global(n
             + 1)], x_global(n, :)');
    x_global(n + 1, :) = x_temp(end, :); % Update
        state
end

% Extract theta (3rd state) from global model
theta_global = x_global(:, 3);
```

Figure 23: Code for simulating original global model with linear state feedback controller (plotting and formatting output not included)

Problem 4

```
% Plant state-space matrices Ad, Bd, Cd, Dd, as well
    as the plant transfer function and sampling time
    are already in Workspace
Ts = 0.05 % Sampling period
% Create plant transfer function
plant_ss = ss(Ad, Bd, Cd, Dd, Ts);
plant_tf = tf(plant_ss);

% Define PID controller parameters
k_p = 1.25;
k_i = 0.5;
k_d = 0.1;

% PID controller transfer function
numerator = [k_i + k_d, k_p - k_d, k_i]; %
   Coefficients for z^2, z^1, z^0
denominator = [1, -1, 0];                %
   Coefficients for z^2, z^1, z^0
controller_tf = tf(numerator, denominator, Ts);

% Combine plant and controller in open-loop tf
open_loop_tf = series(controller_tf, plant_tf);

% Closed-loop system
closed_loop_tf = feedback(open_loop_tf, 1);
```

Figure 24: Code for simulating closed-loop step response with added PID controller for local linearised model (additional code for formatting and plotting not included)

```
t_global = 0:Ts:3; % Time vector
x_global = zeros(length(t_global), 4); % State
    matrix
x_global(1, :) = [20; 6 * pi; pi / 4; 0]; % Initial
    conditions

y_global = zeros(length(t_global), 1); % Output
    vector for theta
y_global(1) = Cd * x_global(1, :)'; % Initial theta

% Parameters for the global model
k = 0.265; mu = 0.281; nu = 1.357;
m = 0.734; g = 9.81; eta = 0.403;
R = 0.7; L = 0.013; J = 0.033; I = 0.073;
epsilon = 0.1; % Nonlinearity coefficient

% Simulate global model
for n = 1:length(t_global) - 1
    % Compute control input
    e_global = 1 - Cd * x_global(n, :)'; % Reference
        input is 1
    u_global = k_p * e_global + ...
               k_i * sum(e_global) * Ts + ...
               k_d * (e_global - (Cd * x_global(max(
                  n - 1, 1), :)')) / Ts;

    % Simulate over the sampling interval using
       global_model
    [~, x_temp] = ode45(@(t, x) global_model(t, x,
       u_global, k, mu, nu, m, g, eta, R, L, J, I)
       ...
                        - epsilon * x(3)^3, [
                           t_global(n), t_global(n +
                            1)], x_global(n, :)');
    % Update state and output
    x_global(n + 1, :) = x_temp(end, :);
    y_global(n + 1) = Cd * x_global(n + 1, :)';
end
```

Figure 25: Code for simulating original global model with added PID controller (code for formatting and plotting not included)

Problem 5

```
% Control parameters found previously
k = 2.5;  % Gain
r = 0.25; % Zero location
Ts = 0.005; % Sampling time
% Plant state-space matrices Ad, Dd, Cd, Dd already
   determined
% Transfer function
plant_ss = ss(Ad, Bd, Cd, Dd, Ts);
plant_tf = tf(plant_ss);
numerator = [k, -k * r]; % Gc(z) = k(z - r)
denominator = [1, -1];
controller_tf = tf(numerator, denominator, Ts);
open_loop_tf = series(controller_tf, plant_tf);
closed_loop_tf = feedback(open_loop_tf, 1);
```

Figure 26: Code for simulating closed-loop local linearised system with added lead controller (using loop shaping)

```
% Parameters for global model
epsilon = 0.1;  % Nonlinearity
k_param = 0.265; mu = 0.281; nu = 1.357;
m = 0.734; g = 9.81; eta = 0.403;
R = 0.7; L = 0.0013; J = 0.033; I = 0.073;

% Initial conditions
x0_global = [20; 6 * pi; pi / 4; 0];
time_global = 0:Ts:2; % Time
x_global = zeros(length(time_global), 4); % State
   matrix
x_global(1, :) = x0_global; % Initial state
y_global = zeros(length(time_global), 1); % Output
   vector
y_global(1) = Cd * x_global(1, :)'; % Initial output
    (theta)

% Simulate global model
for n = 1:length(time_global) - 1
    % Compute control input
    u_global = k * (1 - r * x_global(n, 3)); % Step
       input reference is 1

    % Simulate over sampling interval using
       global_model.m
    [~, x_temp] = ode45(@(t, x) global_model(t, x,
       u_global, ...
        k_param, mu, nu, m, g, eta, R, L, J, I), ...
        [time_global(n), time_global(n + 1)],
           x_global(n, :)');

    % Update state and output
    x_global(n + 1, :) = x_temp(end, :);
    y_global(n + 1) = Cd * x_global(n + 1, :)';
end
```

Figure 27: Code for simulating closed-loop step response of global model (with added lead controller)

```
% Plant Transfer Function
A = [-53.846, -20.385, 0, 0;
       8.030, -8.515, 0, 0;
       0,       0,       0, 1;
      69.726,  0, -1464.38, -5.521];
B = [76.923; 0; 0; 0];
C = [0, 0, 1, 0];
D = [0];
Ts = 0.005; % Sampling time
plant_ss = ss(A, B, C, D);
plant_tf = c2d(tf(plant_ss), Ts, 'zoh');

% Bode Plot
figure;
bode(plant_tf);
grid on;
title('Bode Plot of Plant in Discrete Time');
```

Figure 28: Code for bode plot of original plant (similarly-structured code was also used to find the bode plot of the plant with the added controller)