Project Report

Fall 2024

CSE-310 Control Systems

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Engr. Waseem Ullah Khan

Date:

8th January 2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Contents

1 Problem which is considered **3**

2 Solution **4**
2.1 State-space Representation of the System 4
2.2 Stability analysis of the system 4
2.2.1 Eigen Values 4
2.2.2 Poles 4
2.2.3 Step Response 4
2.2.4 Pole-Zero Map 5
2.2.5 Root Locus 5
2.3 Controllability analysis of the system 5
2.4 Observability analysis of the system 6
2.5 Controller Design for the system 6
2.5.1 Steady-State Error 6
2.5.2 PID Controller Design 8
2.6 Controller Design for given System 9
2.6.1 Step Input ($R(s) = \frac{1}{s}$) 9
2.6.2 Ramp Input ($R(s) = \frac{1}{s^2}$) 9
2.6.3 Parabolic Input ($R(s) = \frac{1}{s^3}$) 10

3 MATLAB Code **10**

4 Results and Discussions **14**
4.1 SSE for Step Response 15
4.1.1 SSE Calculation before PID Design 15
4.1.2 SSE Calculation after PID Design 15
4.2 SSE for Ramp Response 15
4.2.1 SSE Calculation before PID Design 15
4.2.2 SSE Calculation after PID Design 16
4.3 SSE for Parabolic Response 16
4.3.1 SSE Calculation before PID Design 16
4.3.2 SSE Calculation after PID Design 16

5 Conclusion **19**

1 Problem which is considered

Perform the following for Problem 22 at Page 148:
a. Consider the state-space of Problem 22, Page 148 of Norman Nise Book Edition 5.
b. Check the stability of the system using all the methods that you know.
c. Compute the controllability and observability for the system. If the system is unstable, design a suitable controller for it.
d. Simulate the system using the controller that you design and show all the responses.
e. Design a PID Controller and show the response of the system using PID Controller. Compare the results obtained in part d and e.
f. Compute the steady state errors before and after designing controller.

The Problem 22 at Page 148 is qouted as: "' In the past, Type-1 diabetes patients had to inject themselves with insulin three to four times a day. New delayed-action insulin analogues such as insulin Glargine require a single daily dose. A similar procedure to the one described in the Pharmaceutical Drug Absorption case study of this chapter is used to find a model for the concentration-time evolution of plasma for insulin Glargine. For a specific patient, state-space model matrices are given by (Tarín, 2007)"'

$$A = \begin{bmatrix} -0.435 & 0.209 & 0.02 \\ 0.268 & -0.394 & 0 \\ 0.227 & 0 & -0.02 \end{bmatrix} B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.003 & 0 & 0 \end{bmatrix} D = 0$$

where the state vector is given by

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

The state variables are:

- x_1: insulin amount in the plasma compartment,
- x_2: insulin amount in the liver compartment,
- x_3: insulin amount in the interstitial (body tissue) compartment.

The system input is u: external insulin flow.
The system output is y: plasma insulin concentration.

2 Solution

In this report, we address the the above problem and explain each subproblem in detail.

2.1 State-space Representation of the System

The state-space representation of the system can be written as follows:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} -0.435 & 0.209 & 0.02 \\ 0.268 & -0.394 & 0 \\ 0.227 & 0 & -0.02 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(t) \tag{1}$$

$$y = \begin{bmatrix} 0.003 & 0 & 0 \end{bmatrix} x \tag{2}$$

2.2 Stability analysis of the system

In this section, we analyze the stability of the system. The stability can be checked using different ways namely eigen values, step response, poles, root locus and RH-stability criteria.

2.2.1 Eigen Values

For our case, the system is of 3rd order and therefore there will be three eigen values. Let λ_1 , λ_2 and λ_3 denote the eigen values of the system. The values of eigen values can be written as follows:

$$\lambda_1 = -0.6560, \lambda_2 = -0.1889, \lambda_3 = -0.0042 \tag{3}$$

As we can see all of the eigen values are negative, which indicates the system is stable.

2.2.2 Poles

Next, we verify the same fact by observing the poles of the system. Let p_1, p_2 and p_3 denote the poles of the system. The values for poles are as follows:

$$p_1 = -0.6560, p_2 = -0.1889, p_3 = -0.0042 \tag{4}$$

We observe here again that all of the poles are positive, which indicates the system is stable.

2.2.3 Step Response

Next, we verify the same fact by seeing the step-response of the system. The step-response of open-loop system is shown in Figure 2.

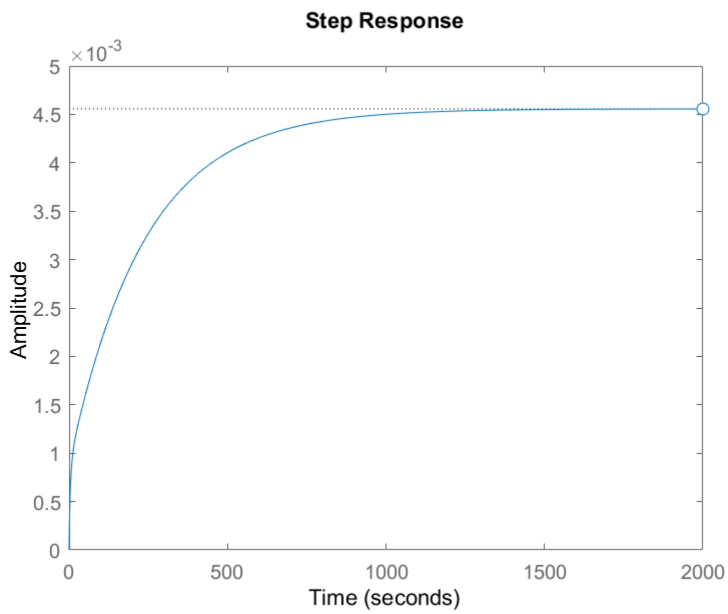

Figure 1: Plot of step response in MATLAB.

From Figure 2, we can do the following analysis:

$$\%OS = 0\%$$
$$T_r = 496s$$
$$T_s = 882s$$
$$PeakValue = 0.00456$$
$$FinalValue = 0.00456$$

As there are no sign changes in the first column, the system is stable.

2.2.4 Pole-Zero Map

Next, I plotted the system's pole-zero map to analyze stability. All poles are in the left-half plane, as shown in Figure 3. This confirms the system is stable.

2.2.5 Root Locus

I performed a root locus analysis to observe pole movement as the gain K changes. Figure 4 shows all poles stay in the left-half plane, verifying the system's stability.

2.3 Controllability analysis of the system

As the system is stable, there's no need for controllability analysis.

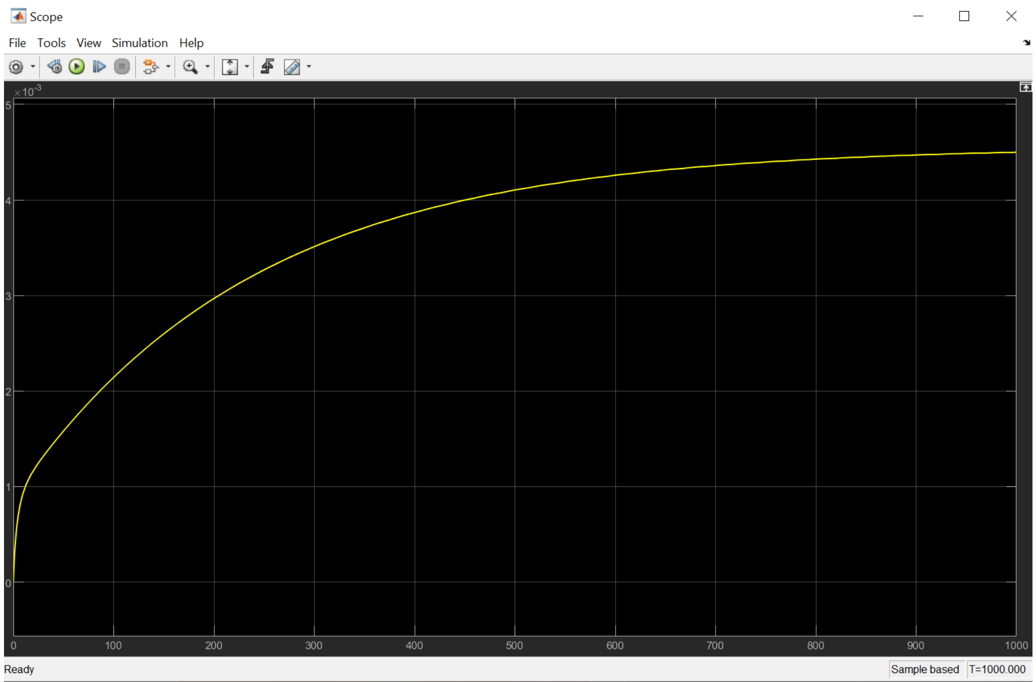

Figure 2: Plot of step response in Simulink.

2.4 Observability analysis of the system

As the system is stable, there's no need for observability analysis.

2.5 Controller Design for the system

As the system is stable, there's no need to design Full-State Feedback Controller or Observer-based Controller. However, to reduce the steady-state error, we can design a PID Controller for it.

2.5.1 Steady-State Error

The steady-state error (SSE) is an important measure of a control system's performance, indicating how accurately the system tracks the reference input in the steady state.

For this system in a unity feedback configuration, the steady-state error depends on the system type and the input signal. Using the *Final Value Theorem*, the steady-state error is calculated as:

$$\text{SSE} = \lim_{s \to 0} s \cdot E(s)$$

where $E(s)$ is the Laplace transform of the error signal, $E(s) = R(s) - T(s) \cdot$

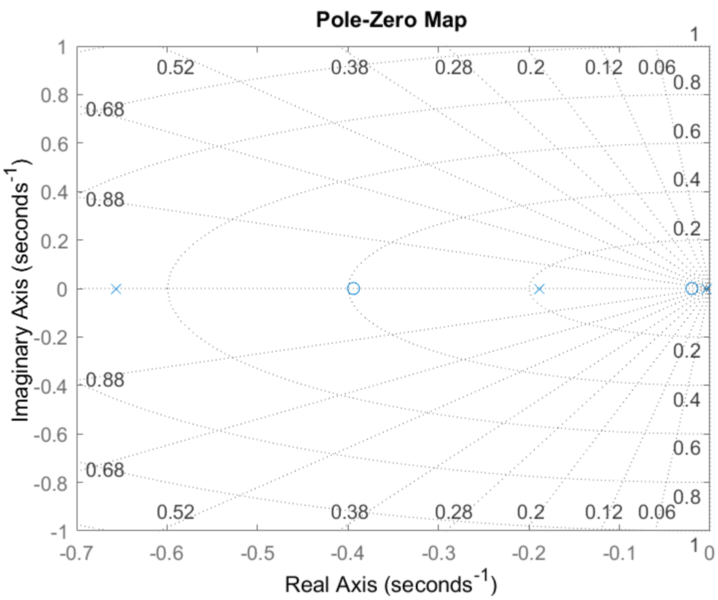

Figure 3: Pole Zero Map of the System

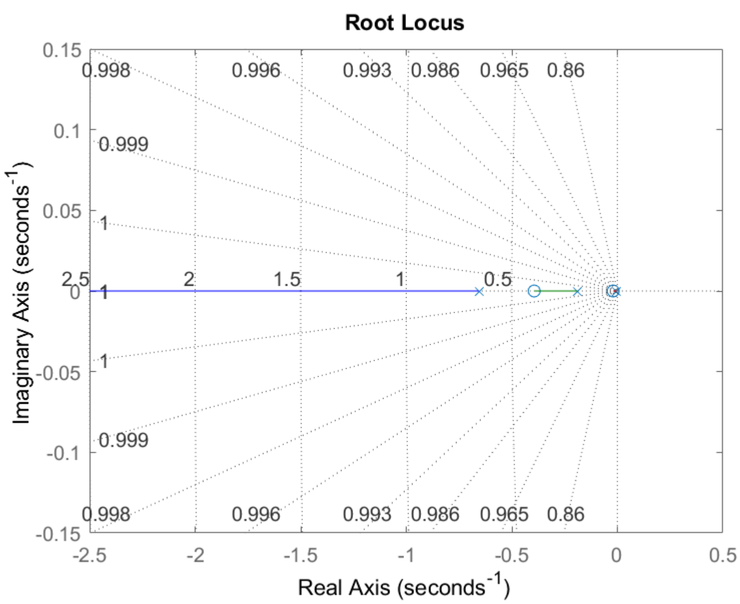

Figure 4: Root Locus Plot of System.

$R(s)$. For a unity feedback system, $E(s) = \frac{R(s)}{1+G(s)}$, where $G(s)$ is the open-loop transfer function.

Step Input $\left(R(s) = \frac{1}{s}\right)$ For a step input, the steady-state error is given by:

$$\text{SSE} = \frac{1}{1 + K_p}$$

where $K_p = \lim_{s \to 0} G(s)$ is the *position error constant.*

Ramp Input $\left(R(s) = \frac{1}{s^2}\right)$ For a ramp input, the steady-state error is given by:

$$\text{SSE} = \frac{1}{K_v}$$

where $K_v = \lim_{s \to 0} s \cdot G(s)$ is the *velocity error constant.*

Parabolic Input $\left(R(s) = \frac{1}{s^3}\right)$ For a parabolic input, the steady-state error is:

$$\text{SSE} = \frac{1}{K_a}$$

where $K_a = \lim_{s \to 0} s^2 \cdot G(s)$ is the *acceleration error constant.*

—

2.5.2 PID Controller Design

To reduce the steady-state error, we design a PID controller for the system. The transfer function of a PID controller is:

$$C(s) = K_p + \frac{K_i}{s} + K_d \cdot s$$

where:

- K_p is the proportional gain, which reduces the rise time and improves the transient response.
- K_i is the integral gain, which eliminates the steady-state error by adding a pole at the origin.
- K_d is the derivative gain, which improves system stability and reduces overshoot.

The overall closed-loop transfer function with the PID controller becomes:

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)}$$

2.6 Controller Design for given System

To analyze the steady-state error for different types of input signals, we apply the *Final Value Theorem*, which states:

$$\text{SSE} = \lim_{s \to 0} s \cdot E(s)$$

where $E(s)$ is the Laplace transform of the error signal. For a unity feedback system, $E(s) = \frac{R(s)}{1+G(s)}$, and $G(s)$ is the open-loop transfer function.

Given the system:

$$G(s) = \frac{0.0003s^2 + 0.0001242s + 2.364 \times 10^{-6}}{s^3 + 0.849s^2 + 0.1274s + 0.0005188}$$

we calculate the steady-state error for different input types.

2.6.1 Step Input ($R(s) = \frac{1}{s}$)

For a step input, the steady-state error is given by:

$$\text{SSE} = \lim_{s \to 0} s \cdot \frac{\frac{1}{s}}{1 + G(s)} = \frac{1}{1 + K_p}$$

where $K_p = \lim_{s \to 0} G(s)$ is the *position error constant.*

For the given system:

$$G(s)\Big|_{s=0} = \frac{2.364 \times 10^{-6}}{0.0005188} \approx 0.00456$$

Thus:

$$\text{SSE} = \frac{1}{1 + 0.00456} \approx 0.9955$$

2.6.2 Ramp Input ($R(s) = \frac{1}{s^2}$)

For a ramp input, the steady-state error is given by:

$$\text{SSE} = \lim_{s \to 0} s \cdot \frac{\frac{1}{s^2}}{1 + G(s)} = \frac{1}{K_v}$$

where $K_v = \lim_{s \to 0} s \cdot G(s)$ is the *velocity error constant.*

For the given system:

$$K_v = \lim_{s \to 0} s \cdot \frac{0.0003s^2 + 0.0001242s + 2.364 \times 10^{-6}}{s^3 + 0.849s^2 + 0.1274s + 0.0005188} = 0$$

Thus:

$$\text{SSE} = \frac{1}{0} = \infty$$

2.6.3 Parabolic Input ($R(s) = \frac{1}{s^3}$)

For a parabolic input, the steady-state error is given by:

$$\text{SSE} = \lim_{s \to 0} s \cdot \frac{\frac{1}{s^3}}{1 + G(s)} = \frac{1}{K_a}$$

where $K_a = \lim_{s \to 0} s^2 \cdot G(s)$ is the *acceleration error constant.*

For the given system:

$$K_a = \lim_{s \to 0} s^2 \cdot \frac{0.0003s^2 + 0.0001242s + 2.364 \times 10^{-6}}{s^3 + 0.849s^2 + 0.1274s + 0.0005188} = 0$$

Thus:

$$\text{SSE} = \frac{1}{0} = \infty$$

3 MATLAB Code

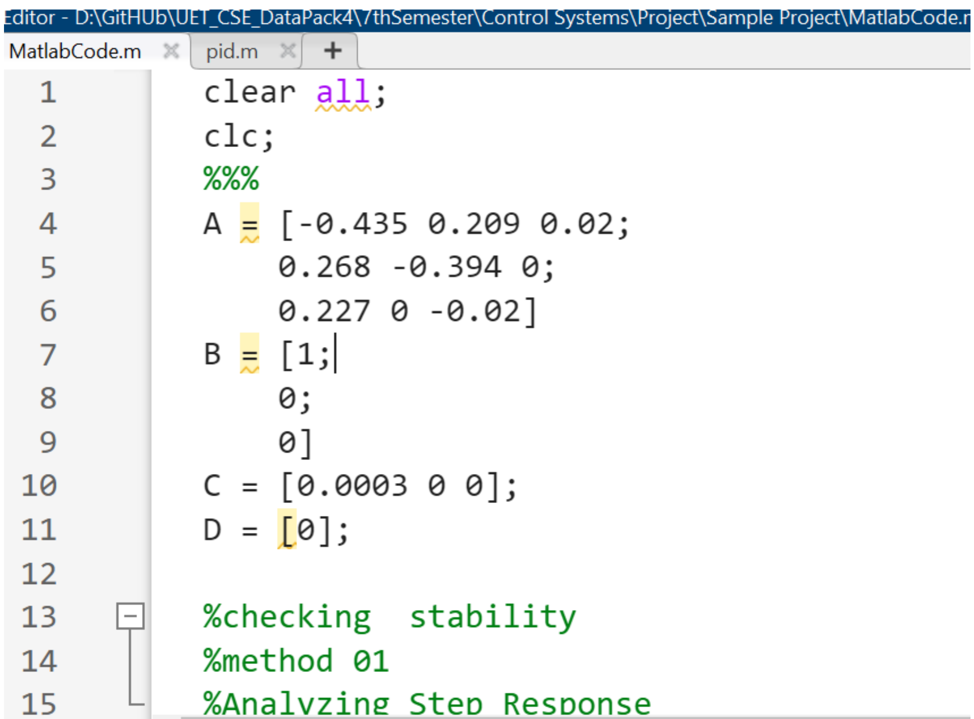

Figure 5: MATLAB Code Part 1

MatlabCode.m | pid.m

```matlab
%method 01
%Analyzing Step Response
figure
step(A,B,C,D);

%method 02
%Displaying A matrix and its eigenvalues
disp('Matrix A = ')
disp(A)
disp('The eigenvalues of matrix A are:-')
eigen_values = eig(A)

%method 03
%Poles of Transfer function
[num1 , denum1] = ss2tf(A,B,C,D);
```

Figure 6: MATLAB Code Part 2

MatlabCode.m | pid.m

```matlab
[num1 , denum1] = ss2tf(A,B,C,D);
disp('The transfer function of input 1 is:-')
Sys1=tf(num1,denum1)
disp('The poles for input 1 are:')
Poles_of_input_1 = roots(denum1)
%%%
%method4 RH Method
% Method 5: Pole-Zero Map
%disp('--- Method 5: pzmap ---');
figure;
pzmap(Sys1);
title('Pole-Zero Map');
grid on;
```

Figure 7: MATLAB Code Part 3

MatlabCode.m pid.m

```matlab
% Method 6: Root Locus
%disp('--- Method 6: Root Locus ---');
figure;
rlocus(Sys1);
title('Root Locus');
grid on;

%%
sys = feedback(Sys1,1);
step(sys)
info = stepinfo(sys);
disp(info);
hold on
steady_state_value = info.SettlingMin;  % Approximate stead
reference_value = 1;  % For step input, reference is 1
```

Figure 8: MATLAB Code Part 2

MatlabCode.m pid.m

```matlab
reference_value = 1;  % For step input, reference is 1
sse = abs(reference_value - steady_state_value);
disp(['Steady-State Error (Step Input): ', num2str(sse)]);

%%

%Kp = 249.004887914577;
%Ki = 3.01944271063754;
%Kd = -7758.2518873453;
%p = pid(Kp,Ki,Kd);
p = pidtune(Sys1, 'pid');
sys_new = feedback(p*Sys1,1);
step(sys_new)
info1 = stepinfo(sys_new);
disp(info1);
```

Figure 9: MATLAB Code Part 5

```
MatlabCode.m   pid.m
disp(info1);
steady_state_value = info1.SettlingMin;  % Approximate stea
reference_value = 1;  % For step input, reference is 1
sse = abs(reference_value - steady_state_value);
disp(['Steady-State Error (Step Input): ', num2str(sse)]);

%%
% Ramp and parabolic response
figure;

% Time vector
t = 0:0.01:2000;  % Simulation time (adjust as needed)

% Ramp Input
ramp_input = t;
```

Figure 10: MATLAB Code Part 6

```
MatlabCode.m
ramp_input = t;
[response_ramp, t_ramp] = lsim(sys_new, ramp_input, t);
subplot(2,1,1);
plot(t_ramp, response_ramp, 'b', 'LineWidth', 1.5);
hold on;
plot(t_ramp, ramp_input, 'r--', 'LineWidth', 1); % Reference ramp
title('Ramp Response');
xlabel('Time (s)');
ylabel('Output');
legend('System Response', 'Ramp Input');
grid on;

% Parabolic Input
parabolic_input = 0.5 * t.^2;
[response_parabolic, t_parabolic] = lsim(sys_new, parabolic_input, t);
```

Figure 11: MATLAB Code Part 7

4 Results and Discussions

We simulated the above system. The schematic for simulation (using Simulink) is shown in Figure 13 and the values for Kp, Ki, Kd are shown in Figure 13.

```
[response_parabolic, t_parabolic] = lsim(sys_new, parabolic_input, t)
subplot(2,1,2);
plot(t_parabolic, response_parabolic, 'g', 'LineWidth', 1.5);
hold on;
plot(t_parabolic, parabolic_input, 'r--', 'LineWidth', 1); % Referenc
title('Parabolic Response');
xlabel('Time (s)');
ylabel('Output');
legend('System Response', 'Parabolic Input');
grid on;
% Analyzing the steady-state errors for ramp and parabolic inputs
steady_state_error_ramp = abs(ramp_input(end) - response_ramp(end));
disp(['Steady-State Error for Ramp Input: ', num2str(steady_state_err
steady_state_error_parabolic = abs(parabolic_input(end) - response_pa
disp(['Steady-State Error for Parabolic Input: ', num2str(steady_stat
```

Figure 12: MATLAB Code Part 8

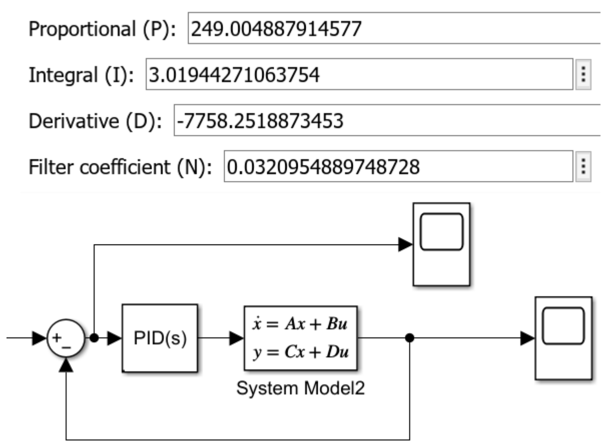

Figure 13: PID Controller in Simulink

4.1 SSE for Step Response

This section describes the SSE computation before and after PID Design using Step Signal as input.

4.1.1 SSE Calculation before PID Design

The steady-state error was calculated by making a unity closed loop system. The sketch can be seen in the figure 14. The value of SSE is same as calculated in 2.6.1

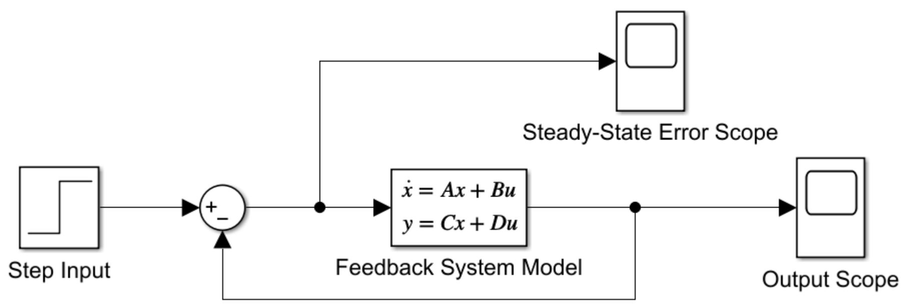

Figure 14: Unity Feedback System With Step as Input

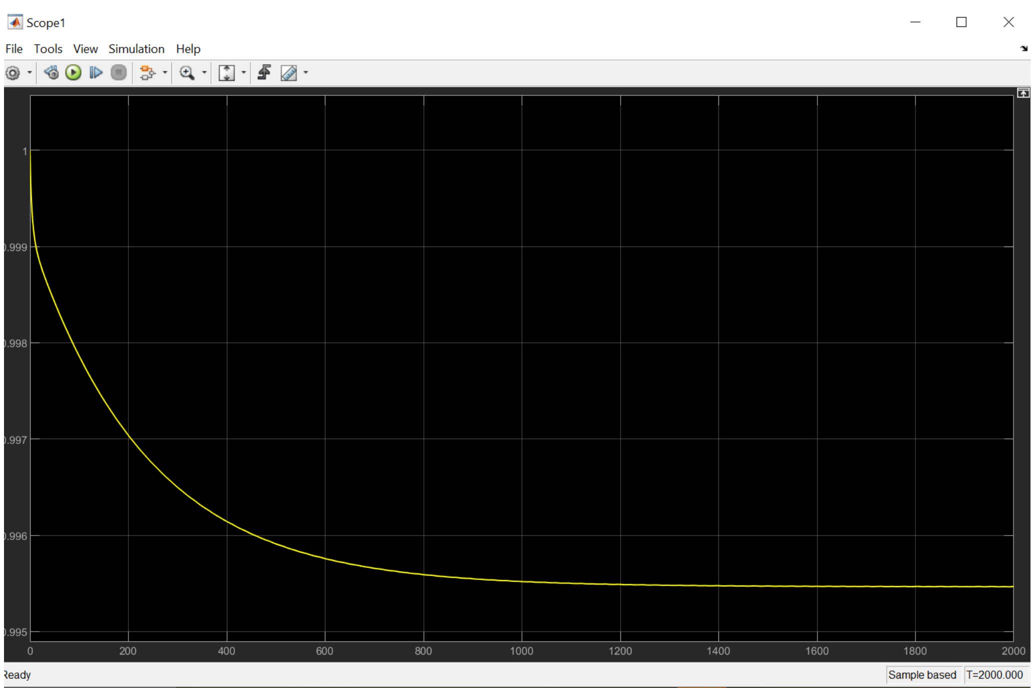

Figure 15: SSE Calculation before PID in Simulink

4.1.2 SSE Calculation after PID Design

After connecting the PID Controller in series as shown in Figure 13, the SSE reaches almost to zero and we get a stable response at 1. The step response is shown in figure 16 (MATLAB) and Figure 17 (Simulink). The SSE values can be verified from Figure 18 (Simulink). A comparison of before and after SSE can also be seen in Figure 19.

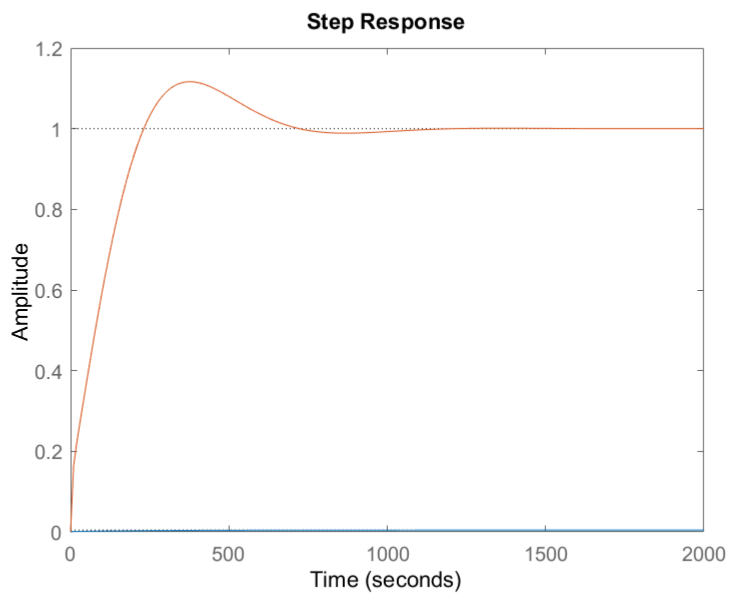

Figure 16: Step Response after PID in MATLAB

4.2 SSE for Ramp Response

This section describes the SSE computation before and after PID Design using Ramp Signal as input.

4.2.1 SSE Calculation before PID Design

The procedure is same as described in previous section 4.1 but with Ramp as input signal. The sketch for computing SSE for Ramp is shown in Figure 20.

4.2.2 SSE Calculation after PID Design

After connecting the PID Controller in series as shown in Figure 13, the SSE reaches almost to 75 and the response is still inifinite. The ramp response is shown in figure 22 (MATLAB) and Figure 23 (Simulink). The SSE value from MATLAB is 57.5251 for Ramp Input.

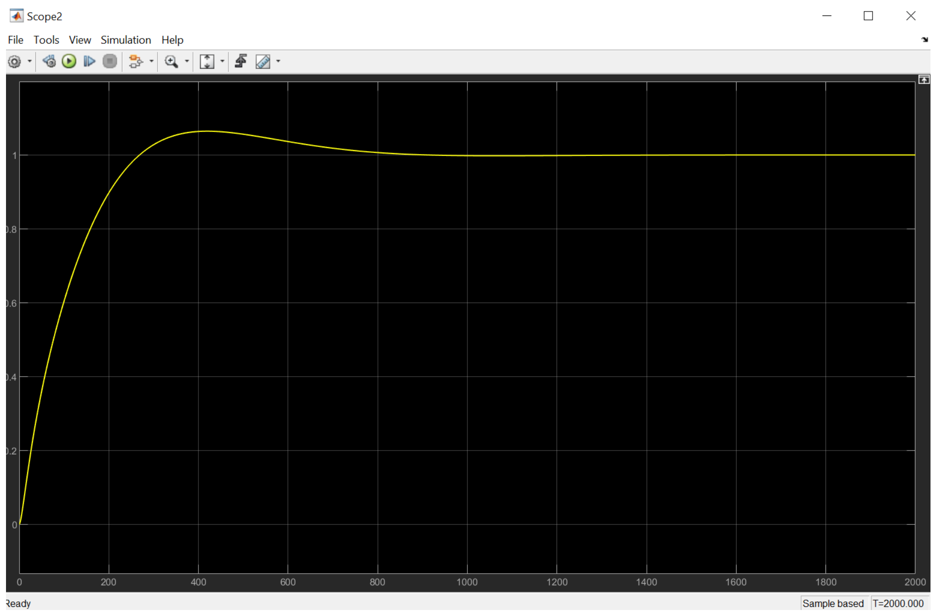

Figure 17: Step Response after PID in Simulink

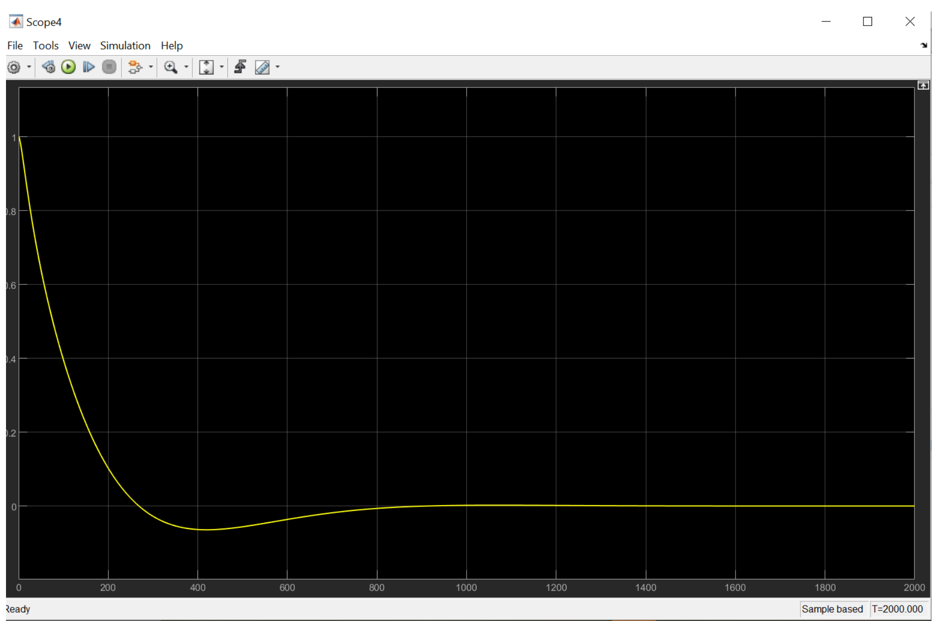

Figure 18: SSE Calculation after PID in Simulink

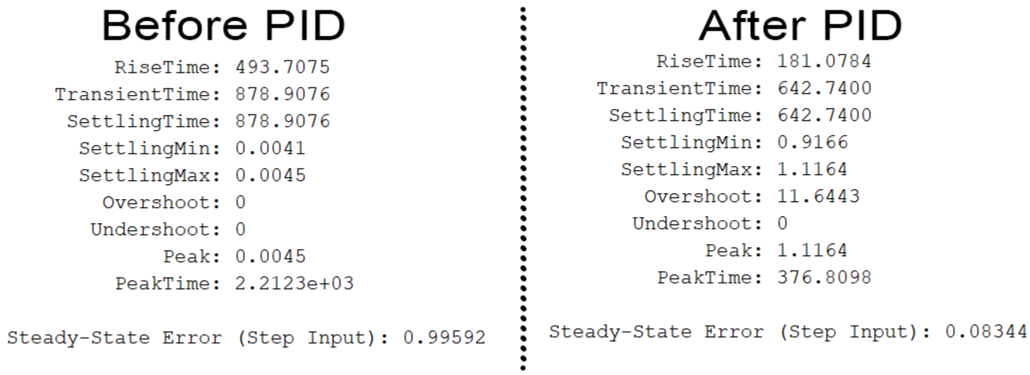

Figure 19: SSE Calculation before and After PID in MATLAB

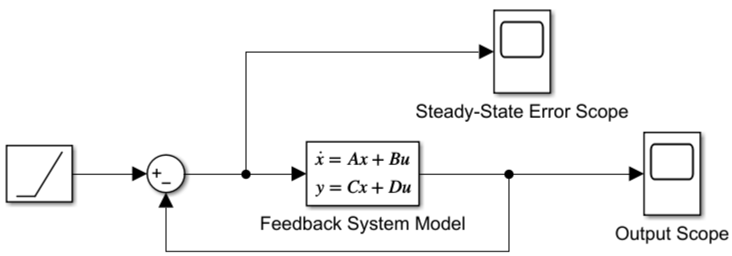

Figure 20: Unity Feedback System With Ramp as Input

4.3 SSE for Parabolic Response

This section describes the SSE computation before and after PID Design using Parabolic Signal as input.

4.3.1 SSE Calculation before PID Design

The procedure is same as described in previous section 4.1 but with Parabolic as input signal. The sketch for computing SSE for Parabolic is shown in Figure 24.

4.3.2 SSE Calculation after PID Design

After connecting the PID Controller in series as shown in Figure 13, the SSE reaches to infinity and the response is also inifinite. The Parabolic response is shown in figure 22 (MATLAB) and Figure 26 (Simulink).

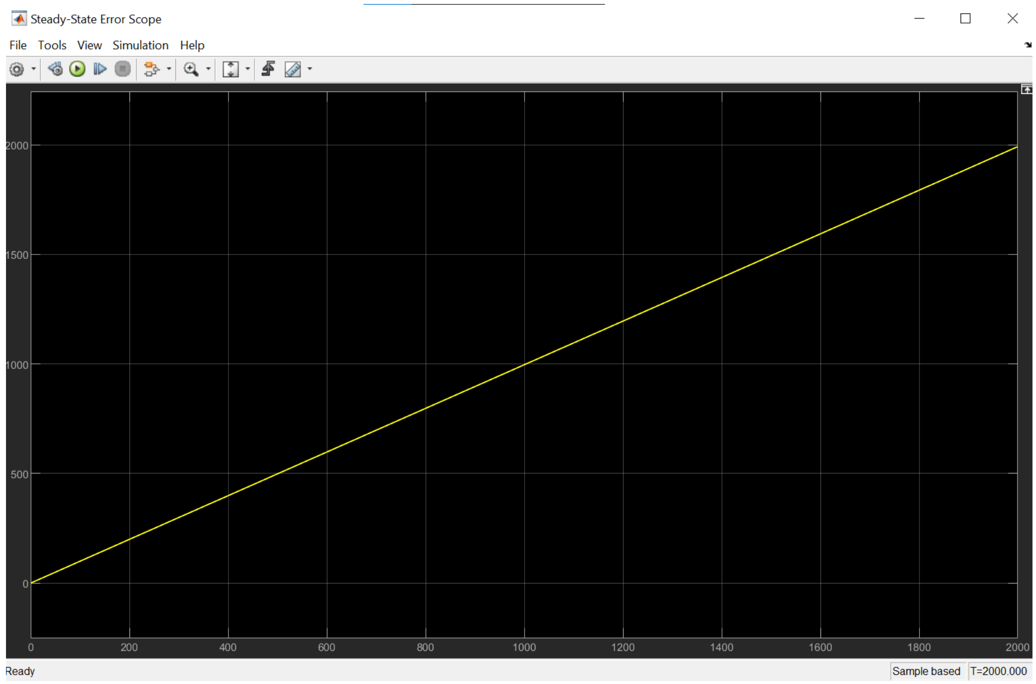

Figure 21: SSE Calculation before PID in Simulink for Ramp

5 Conclusion

In this project, we analyzed and designed a control system for a state-space model derived from Problem 22 in the Norman Nise book. The system stability was verified using eigenvalues, poles, and step response analysis, confirming that the system is inherently stable. Simulation results for the PID controller offered a significant reduction in steady-state error while maintaining acceptable transient behavior. This project highlights the importance of system analysis and appropriate controller design in achieving optimal performance for control systems.

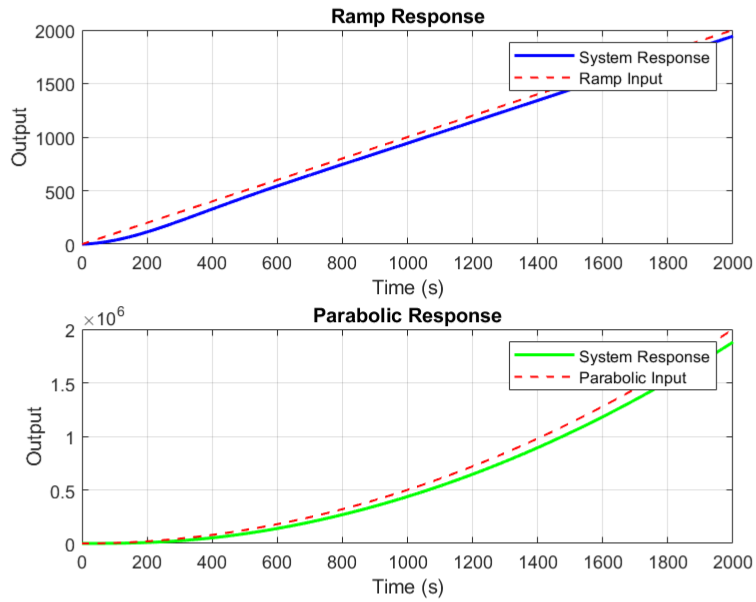

Figure 22: Ramp Response after PID in MATLAB

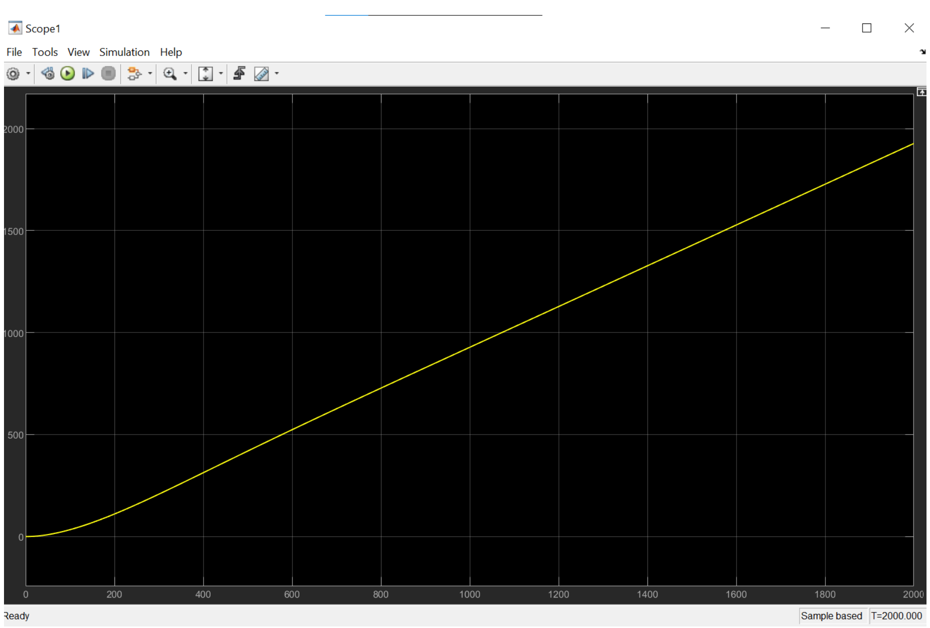

Figure 23: Ramp Response after PID in Simulink

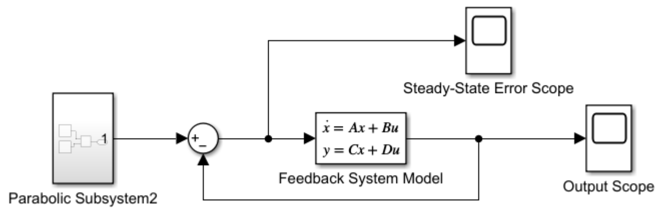

Figure 24: Unity Feedback System With Parabolic as Input

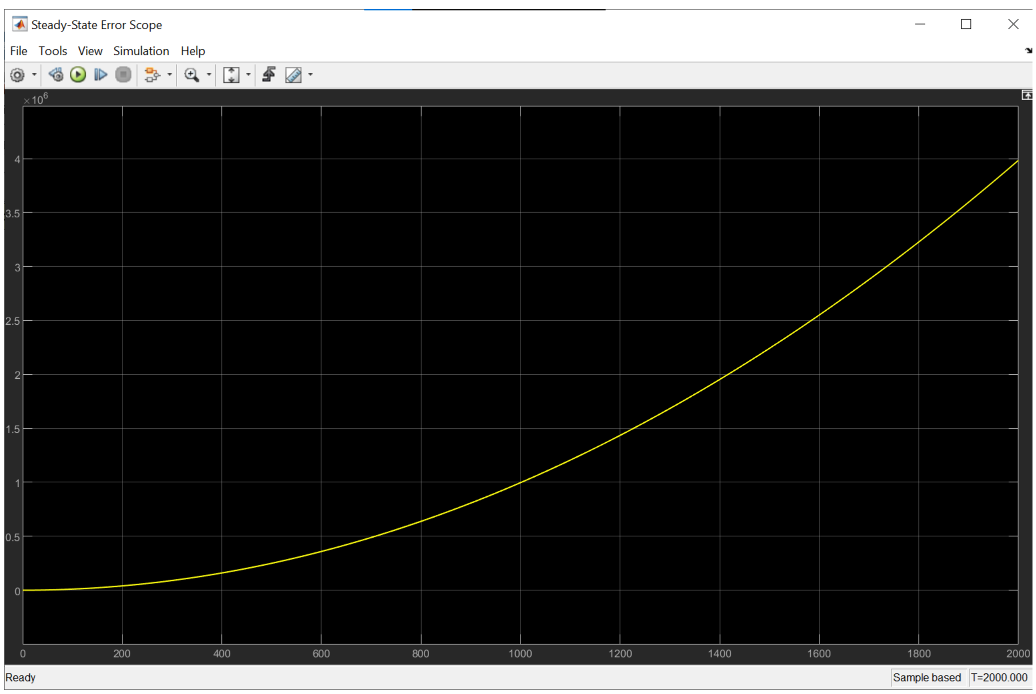

Figure 25: SSE Calculation before PID in Simulink for Parabolic

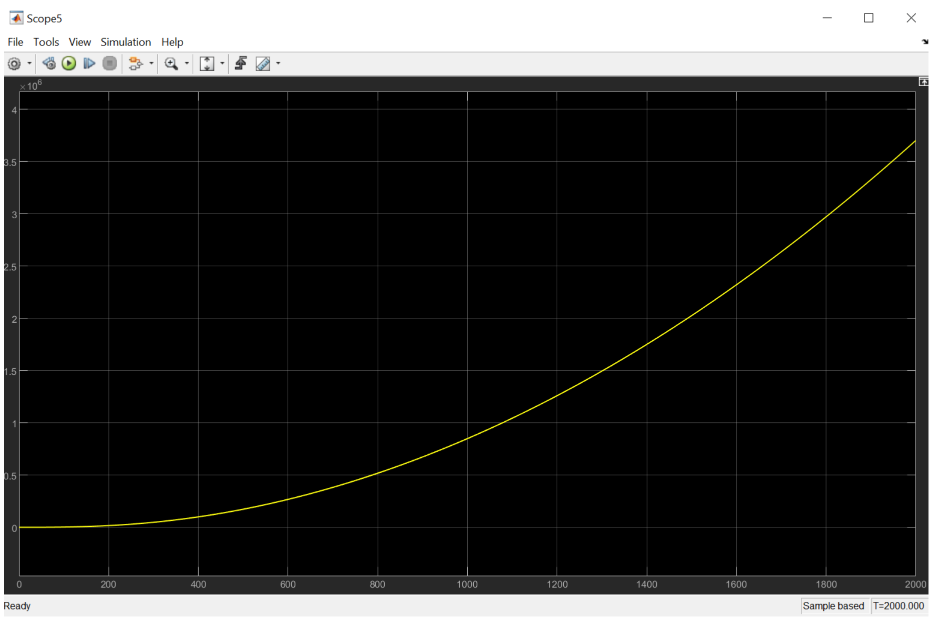

Figure 26: Parabolic Response after PID in Simulink