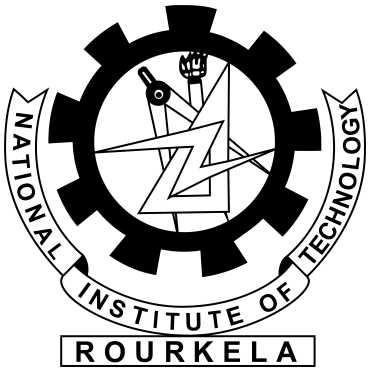
NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA

Department of Electrical Engineering

Design and Analysis of an Autopilot Control System for a Riderless Bicycle



Course:Advanced Control Systems [EE3302]

Spring semester 2024-25

Submitted by: Medipally Saivivek
Roll Number: 122EE0378
Under the guidance of: Prof. Manas Kumar Bera

Abstract

This project investigates the development of an autopilot system for a riderless bicycle to maintain vertical stability at three specific velocities: 0 m/s (stationary), 3.5 m/s, and 5 m/s. A fourth-order linear parameter-varying (LPV) model is employed, with the system states comprising roll angle, roll rate, steer angle, and steer rate. The system's input is steering torque, and the output is roll angle.

For each velocity, detailed analysis is conducted to derive state-space models and transfer functions, compute poles and zeros, analyze system eigenvalues, and assess stability characteristics (asymptotic, marginal, and BIBO). Time-domain responses are simulated under zero-input and unit step input conditions to understand the system's dynamic behavior. The project demonstrates that velocity significantly affects bicycle stability—unstable at 0 m/s and 5 m/s, but inherently stable at 3.5 m/s.

Based on controllability analysis confirming the system's controllability at all velocities, feedback controllers are designed with specific performance criteria using SISO tool in matlab. The controllers successfully stabilize the bicycle at all three velocities, though with varying levels of difficulty. The most challenging control scenarios occur at 0 m/s (stationary) and 5 m/s (high speed) due to right-half-plane poles. MATLAB's control system tools are used for analysis and controller design, with simulation results validating the effectiveness of the proposed control strategies.

Contents

Chapter 1

Introduction

1.1 Background

Riderless bicycles present unique control challenges due to their inherent instability at certain speeds and their nonlinear dynamics. The control of autonomous bicycles has applications in various fields, including personal transportation, last-mile delivery systems, and research platforms for studying bicycle dynamics. The stability of a bicycle is particularly interesting as it varies significantly with forward velocity—a characteristic that makes it a compelling subject for advanced control system design.

1.2 Problem Statement

This project focuses on designing an autopilot system for a riderless bicycle to maintain vertical upright stability at three fixed velocities: 0 m/s, 3.5 m/s, and 5 m/s. Using a fourth-order linear parameter-varying (LPV) model with states comprising roll angle, roll rate, steer angle, and steer rate, the system takes steering torque as input and produces roll angle as output. The project involves:

1. Deriving state-space models and transfer functions for each velocity, computing poles,zeros, and analyzing system eigenvalues.
2. Simulating time responses under zero-input and unit step input conditions with assumed non-zero initial states.
3. Analysing Asymptotic, marginal, BIBO stability at each velocity.
4. Checking the possibility of control and Designing velocity-specific feedback controllers to stabilize the bicycle, meeting specific performance criteria related to steady-state error, damping ratio, and settling time.

5. Validating controller performance through closed-loop step response simulations using SISOtool in MATLAB.

1.3 Modeling Approach

The bicycle is modeled using a linear parameter-varying (LPV) approach, where the system parameters depend on the forward velocity. This approach allows us to examine the system behavior at different operating velocities (0 m/s, 3.5 m/s,5 m/s) by linearizing around these points.

The state variables are :

$$x = \begin{bmatrix} \phi & \dot{\phi} & \delta & \dot{\delta} \end{bmatrix}^T \tag{1.1}$$

where:

- ϕ - Roll angle (bicycle's leaning angle from vertical)
- $\dot{\phi}$ - Roll rate
- δ - Steer angle
- $\dot{\delta}$ - Steer rate

The input u is the steering torque applied to the handlebars.
The output y is the roll angle ϕ, representing the bicycle's deviation from the vertical upright position.

Chapter 2

State-Space Representation and Transfer Functions

For each velocity, a state-space model is derived in the form:

$$\dot{x} = Ax + Bu \tag{2.1}$$

$$y = Cx + Du \tag{2.2}$$

Based on given velocity, three distinct models are obtained.In generalized matrix, output matrix are all state variables but according to our problem statement ,we consider only roll angle as output.

$$A(v) =$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 13.67 & 0.225 - 1.319v^2(t) & -0.164v(t) & -0.552v(t) \\ 4.857 & 10.81 - 1.125v^2(t) & 3.621v(t) & -2.388v(t) \end{bmatrix},$$

$$B = \begin{bmatrix} 0 \\ 0 \\ -0.339 \\ 7.457 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Figure 2.1: General matrix of LPV model

2.1 Model at V = 0 m/s

Matalab code:

```
A1 = [0, 0, 1, 0;
      0, 0, 0, 1;
      13.67, 0.225, 0, 0;
      4.857, 10.81, 0, 0];
disp('Matrix A1:');
disp(A1);

% Define the 1x4 matrix C and 4x1 matrix B using the variable 'a'
C = [1, 0, 0, 0];    % 1x4 row vector
B = [0; 0; -0.339; 7.457];    % 4x1 column vector

% Display the matrices
disp('Matrix C:');
disp(C);

disp('Matrix B:');
disp(B);


syms s

% Define the 4x4 matrix using the symbolic variable 'a'
% X= S*I - A1
X = [s, 0, -1, 0;
      0, s, 0, -1;
      -13.67, -0.225, s, 0;
      -4.857, -10.81, 0, s];

% Display the matrix
disp('Matrix X:');
disp(X);

% finding and to display the inverse
X_inv = inv(X);
disp('Inverse of Matrix X:');
disp(X_inv);
% Transfer function(Tf1) is: (C * inv(X) * B)
Tf1 = C * inv(X) * B;  % (1x4) * (4x4) * (4x1)
% Finding eigenvalues of A1
eigenvalues = eig(A1);

% to display the eigenvalues
disp('Eigenvalues of the matrix A1:');
disp(eigenvalues);
% to display the result
disp('Result of C * inv(X) * B:');
disp(Tf1);
 %now to find zero and poles of this tansfer function
 %  the numerator and denominator coefficients from obtained transfer
 %  function
num = [0,0, -67800,  0, 800631];  % Expand numerator as needed
den = 5 * [8000, 0, -195840, 0, 1173439];   % 5 times denominator coefficients

% Create the transfer function
sys = tf(num, den);
```

```
% Find zeros and poles
z = zero(sys);
p = pole(sys);

% Display results
disp('Zeros:');
disp(z);
disp('Poles:');
disp(p);
% Plot pole-zero map
figure;
pzmap(sys);
title('Pole-Zero Map');
xlabel('Real Axis');
ylabel('Imaginary Axis');
grid on;
% Plot eigne values
z = eig(A1); %
plot(real(z), imag(z), 'o')
axis equal
grid on
xlabel('Re(\lambda)')
ylabel('Im(\lambda)')
title('Eigenvalues in the Complex Plane')
```

Figure 2.2: MATLAB Code

$$\frac{67113}{5(8000s^4 - 195840s^2 + 1173439)} - \frac{678(100s^2 - 1081)}{25(8000s^4 - 195840s^2 + 1173439)}$$

To combine and simplify, get a common denominator:

$$H(s) = \frac{67113 \cdot 5 - 678 \cdot (100s^2 - 1081)}{5(8000s^4 - 195840s^2 + 1173439)}$$

Figure 2.3: Transfer function

```
Matrix A1:
         0         0    1.0000         0
         0         0         0    1.0000
   13.6700    0.2250         0         0
    4.8570   10.8100         0         0

Matrix C:
     1     0     0     0

Matrix B:
         0
         0
   -0.3390
    7.4570

Matrix X:
[         s,         0, -1,  0]
[         0,         s,  0, -1]
[ -1367/100,     -9/40,  s,  0]
[-4857/1000, -1081/100,  0,  s]
```

Figure 2.4: output of matlab code

```
Eigenvalues of the matrix A1:
   -3.7432
   -3.2355
    3.7432
    3.2355

Result of C * inv(X) * B:
67113/(5*(8000*s^4 - 195840*s^2 + 1173439)) - (678*(100*s^2 - 1081))/(25*(8000*s^4 - 195840*s^2 + 1173439))

Zeros:
    3.4364
   -3.4364

Poles:
   -3.7432
   -3.2355
    3.7432
    3.2355
```

Figure 2.5: output of above matlab code

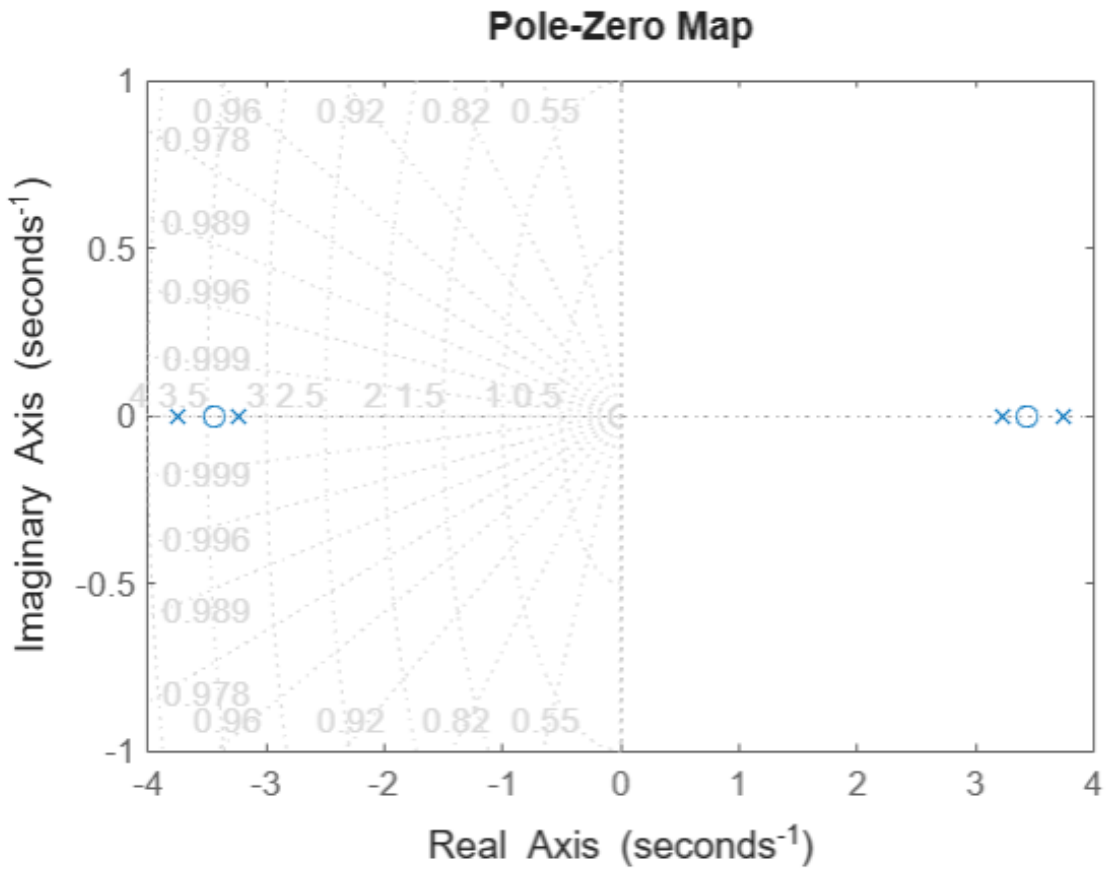

Pole-Zero Map
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
1
0.5
0
-0.5
-1
-4
-3
-2
-1
0
1
2
3
4
0.96
0.92
0.82
0.55
0.978
0.989
0.996
0.999
4
3.5
3
2.5
2
1.5
1
0.5

Figure 2.6

2.2 Model at V = 3.5 m/s

At V = 3.5 m/s, the state-space model has the following matrices:

```
A2 = [0, 0, 1, 0;
     0, 0, 0, 1;
     13.67, -15.93275, -0.574, -1.932;
     4.857, -2.97125, 12.6735, -8.358];
disp('Matrix A2:');
disp(A2);

% Define the 1x4 matrix C and 4x1 matrix B using the variable 'a'
C = [1, 0, 0, 0];    % 1x4 row vector
B = [0; 0; -0.339; 7.457];    % 4x1 column vector

% Display the matrices
disp('Matrix C:');
disp(C);

disp('Matrix B:');
disp(B);

% s is the eigen value
syms s

% Define the 4x4 matrix using the symbolic variable 'a'
% X= S*I - A2
X = [s, 0, -1, 0;
     0, s, 0, -1;
     -13.67, 15.93275, s+0.574, 1.932;
     -4.857, 2.97125,-12.6735, s+8.358];


% Display the matrix
disp('Matrix X:');
disp(X);

% finding and to display the inverse
X_inv = inv(X);
disp('Inverse of Matrix X:');
disp(X_inv);
% Transfer function(Tf1) is: (C * inv(X) * B)
Tf1 = C * inv(X) * B;  % (1x4) * (4x4) * (4x1)
% Finding eigenvalues of A1
eigenvalues = eig(A2);

% to display the eigenvalues
disp('Eigenvalues of the matrix A2:');
disp(eigenvalues);
% to display the result
disp('Result of C * inv(X) * B:');
disp(Tf1);
 %now to find zero and poles of this tansfer function
 %  the numerator and denominator coefficients from obtained transfer
 %  function
num = [0, 0, -2712000, -137873768, -958583464];
den = [8000000, 71456000, 148671552, 790072549, 294147034];

% Create the transfer function
sys = tf(num, den);
```

Figure 2.7: Matlab code

$$H(s) = \frac{-2712000s^2 - 137922288s - 958542164}{8000000s^4 + 71456000s^3 + 148671552s^2 + 790072549s + 294147034}$$

Figure 2.8: Transfer function for V=3.5 m/s

```
>> marix2
Matrix A2:
         0         0    1.0000         0
         0         0         0    1.0000
   13.6700  -15.9328   -0.5740   -1.9320
    4.8570   -2.9712   12.6735   -8.3580

Matrix C:
     1     0     0     0

Matrix B:
         0
         0
   -0.3390
    7.4570

Matrix X:
[         s,          0,          -1,            0]
[         0,          s,           0,           -1]
[ -1367/100, 63731/4000, s + 287/500,      483/250]
[-4857/1000,   2377/800, -25347/2000, s + 4179/500]
```

Figure 2.9: output of above matlab code

```
Eigenvalues of the matrix A2:
  -8.0753 + 0.0000i
  -0.2301 + 3.3809i
  -0.2301 - 3.3809i
  -0.3965 + 0.0000i
```

Figure 2.10: Eigen values when V=3.5 m/s

```
Zeros:
   -42.5270
    -8.3114

Poles:
   -8.0753 + 0.0000i
   -0.2301 + 3.3809i
   -0.2301 - 3.3809i
   -0.3965 + 0.0000i
```

Figure 2.11: poles ,zeros when V=3.5 m/s

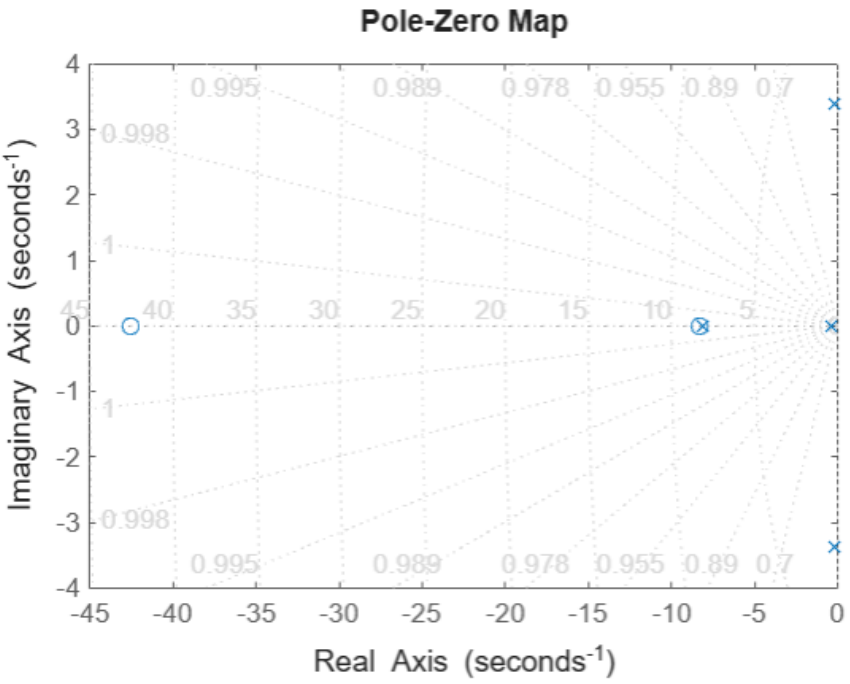


Figure 2.12

2.3 Model at V = 5 m/s

At V = 5 m/s, the state-space model has the following matrices:

```
A3 = [0, 0, 1, 0;
      0, 0, 0, 1;
      13.67, -32.75, -0.82, -2.76;
      4.857, -17.315, 18.105, -11.94];
disp('Matrix A3:');
disp(A3);

% Define the 1x4 matrix C and 4x1 matrix B using the variable 'a'
C = [1, 0, 0, 0];    % 1x4 row vector
B = [0; 0; -0.339; 7.457];    % 4x1 column vector

% Display the matrices
disp('Matrix C:');
disp(C);

disp('Matrix B:');
disp(B);

% s is the eigen value
syms s

% Define the 4x4 matrix using the symbolic variable 'a'
% X= S*I - A3
X = [s, 0, -1, 0;
     0, s, 0, -1;
     -13.67, 32.75, s+0.82, 2.76;
     -4.857, 17.315,-18.105, s+11.94];

% Display the matrix
disp('Matrix X:');
disp(X);

% finding and to display the inverse
X_inv = inv(X);
disp('Inverse of Matrix X:');
disp(X_inv);
% Transfer function(Tf1) is: (C * inv(X) * B)
Tf3 = C * inv(X) * B;  % (1x4) * (4x4) * (4x1)
% Finding eigenvalues of A1
eigenvalues = eig(A3);

% to display the eigenvalues
disp('Eigenvalues of the matrix A3:');
disp(eigenvalues);
% to display the result
disp('Result of C * inv(X) * B:');
disp(Tf3);
 %now to find zero and poles of this tansfer function
 %  the numerator and denominator coefficients from obtained transfer
 %  function
num = [0, 0, -33900, -2462898, -25008653.5];
den = [100000, 1276000, 6340560, 45732257, -7762930];

% Create the transfer function
sys = tf(num, den);
```

```
% Find zeros and poles
z = zero(sys);
p = pole(sys);

% Display results
disp('Zeros:');
disp(z);
disp('Poles:');
disp(p);
% Plot pole-zero map
figure;
pzmap(sys);
title('Pole-Zero Map');
xlabel('Real Axis');
ylabel('Imaginary Axis');
grid on;
% Plot eigne values
z = eig(A3); %
plot(real(z), imag(z), 'o')
axis equal
grid on
xlabel('Re(\lambda)')
ylabel('Im(\lambda)')
title('Eigenvalues in the Complex Plane')
```

Figure 2.13: Matlab code

```
>> matrix3
Matrix A3:
         0         0    1.0000         0
         0         0         0    1.0000
   13.6700  -32.7500   -0.8200   -2.7600
    4.8570  -17.3150   18.1050  -11.9400

Matrix C:
     1     0     0     0

Matrix B:
         0
         0
   -0.3390
    7.4570
```

Figure 2.14: output of above matlab code

```
Matrix X:
[          s,         0,         -1,           0]
[          0,         s,          0,          -1]
[ -1367/100,     131/4, s + 41/50,       69/25]
[-4857/1000, 3463/200, -3621/200, s + 597/50]
```

Figure 2.15: inv (S*I - A3) matrix

$$H(s) = -\frac{7457(276s+3275)}{100000s^4+1276000s^3+6340560s^2+45732257s-7762930} - \frac{339(200s^2+2388s+3463)}{2(100000s^4+1276000s^3+6340560s^2+45732257s-7762930)}$$

$$H(s) = \frac{-7457(276s+3275) - 339(200s^2+2388s+3463)/2}{100000s^4+1276000s^3+6340560s^2+45732257s-7762930}$$

Or, equivalently:

$$\text{Numerator: } N(s) = -7457 \cdot (276s+3275) - \frac{339}{2} \cdot (200s^2+2388s+3463)$$

$$\text{Denominator: } D(s) = 100000s^4+1276000s^3+6340560s^2+45732257s-7762930$$

Figure 2.16: Transfer function for V= 5 m/s

```
Eigenvalues of the matrix A3:
 -10.8598 + 0.0000i
  -1.0330 + 6.4842i
  -1.0330 - 6.4842i
   0.1658 + 0.0000i
```

Figure 2.17

```
Zeros:
  -60.4476
  -12.2043

Poles:
 -10.8598 + 0.0000i
  -1.0330 + 6.4842i
  -1.0330 - 6.4842i
   0.1658 + 0.0000i
```

Figure 2.18

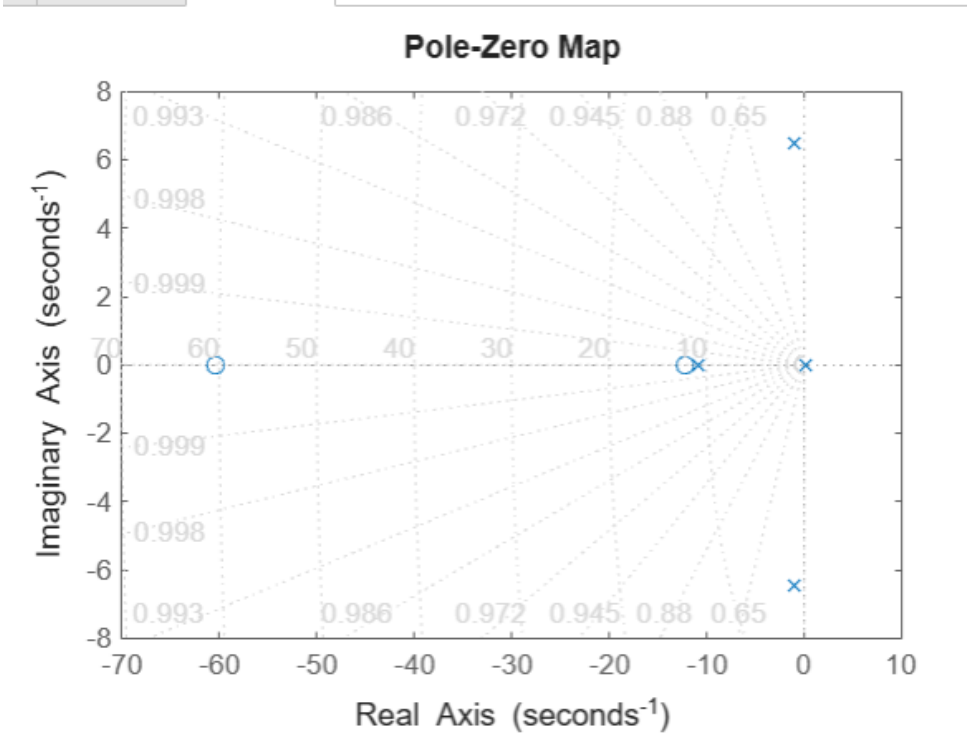


Figure 2.19

Chapter 3

Time Response Analysis

The time responses of the bicycle system are analyzed for each velocity under two conditions:

1. Zero-input response with initial state $x_0 = [2, 2, 0, 0]^T$
2. Response to a unit step input

3.1 Time Response at V = 0 m/s

3.1.1

For the stationary bicycle (V = 0 m/s), the zero-input response with initial condition $x_0 = [2, 2, 0, 0]^T$ shows divergent behavior. The roll angle and roll rate grow unbounded over time, indicating that the system is unstable at this velocity.

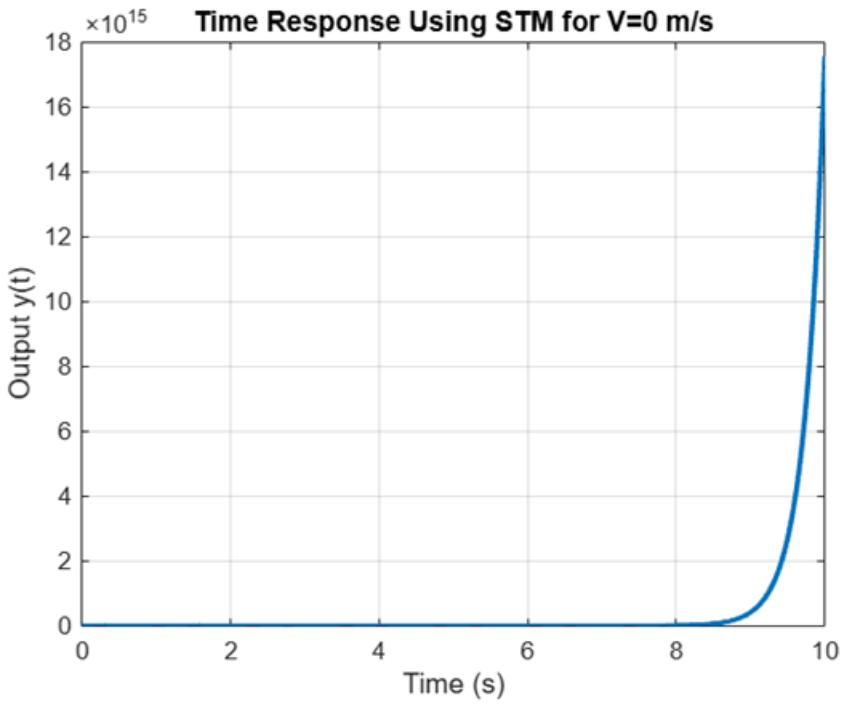


Figure 3.1: Zero-input response at V = 0 m/s

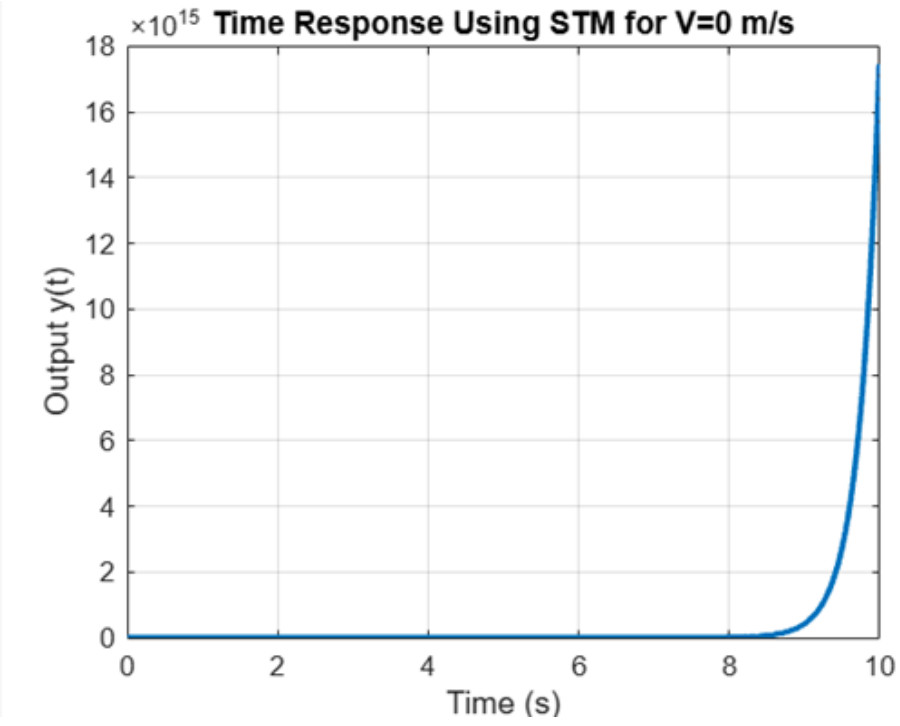


Figure 3.2: response to unit step at V = 0 m/s

3.2 Time Response at V = 3.5 m/s

3.2.1

At V = 3.5 m/s, the zero-input response with the same initial condition shows that the system states eventually return to equilibrium, indicating stability at this velocity.

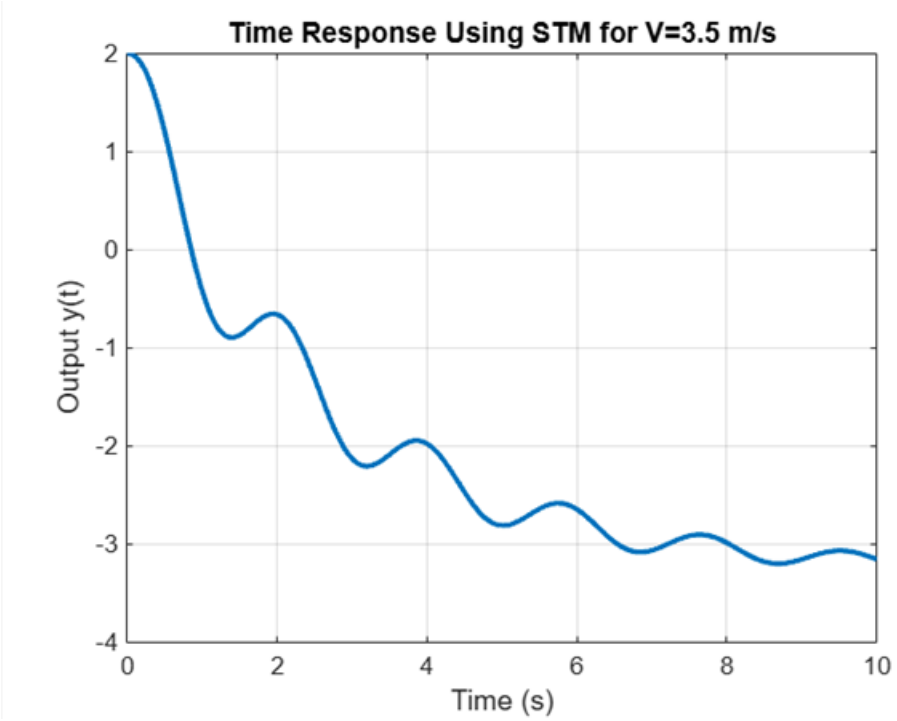


Figure 3.3: Zero-input response at V = 3.5 m/s

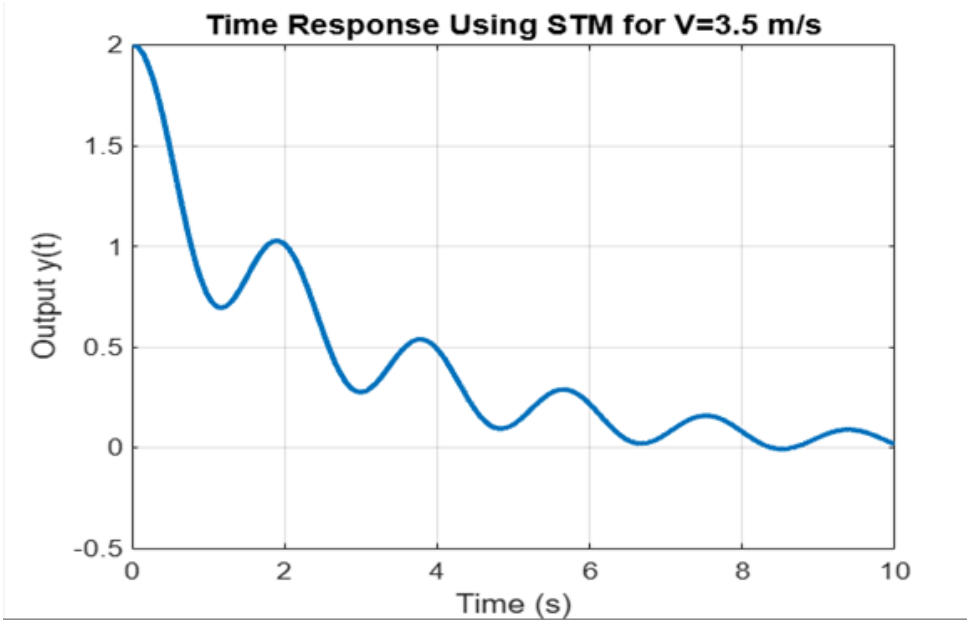


Figure 3.4: response to unit step at V = 3.5 m/s

3.3 Time Response at V = 5 m/s

3.3.1

At V = 5 m/s, the zero-input response again shows divergent behavior, indicating that the system is unstable at this higher velocity as well.

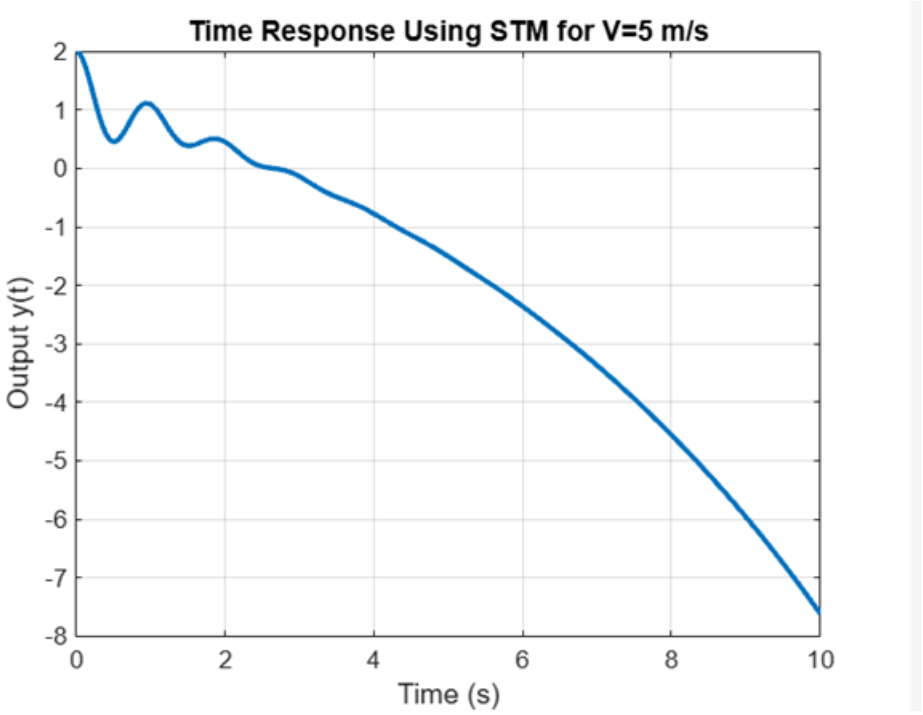


Figure 3.5: Zero-input response at V = 5 m/s

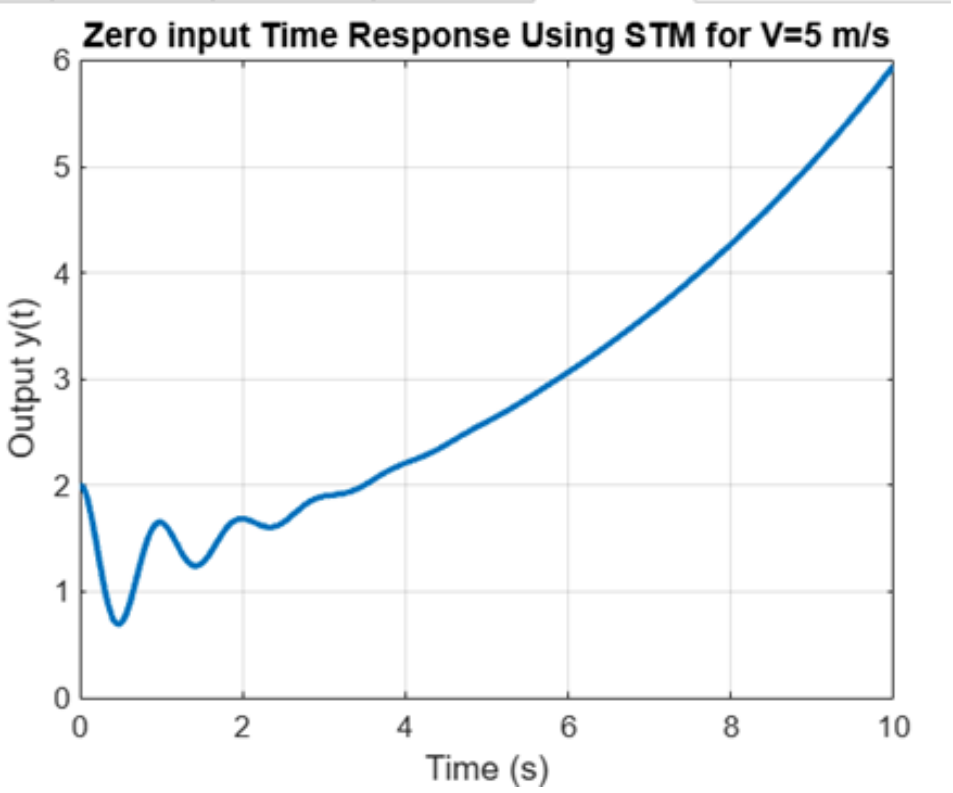


Figure 3.6: response to unit step at V = 5 m/s

Chapter 4

Stability Analysis

Three types of stability are analyzed for each velocity:

- Asymptotic stability: In the absence of input, all state trajectories converge to the equilibrium as time goes to infinity. This requires all eigenvalues to be in the left half-plane.
- Marginal stability:In the frozen LTI model,no eigenvalue lies in the right half-plane, but at least one pair lies on the imaginary axis.
- BIBO (Bounded-Input, Bounded-Output) stability: if the zero-input response is bounded,then LPV system is BIBO stable.

4.1 Stability at V = 0 m/s

Eigen values= -3.7432, -3.2355, 3.7432,3.2355.

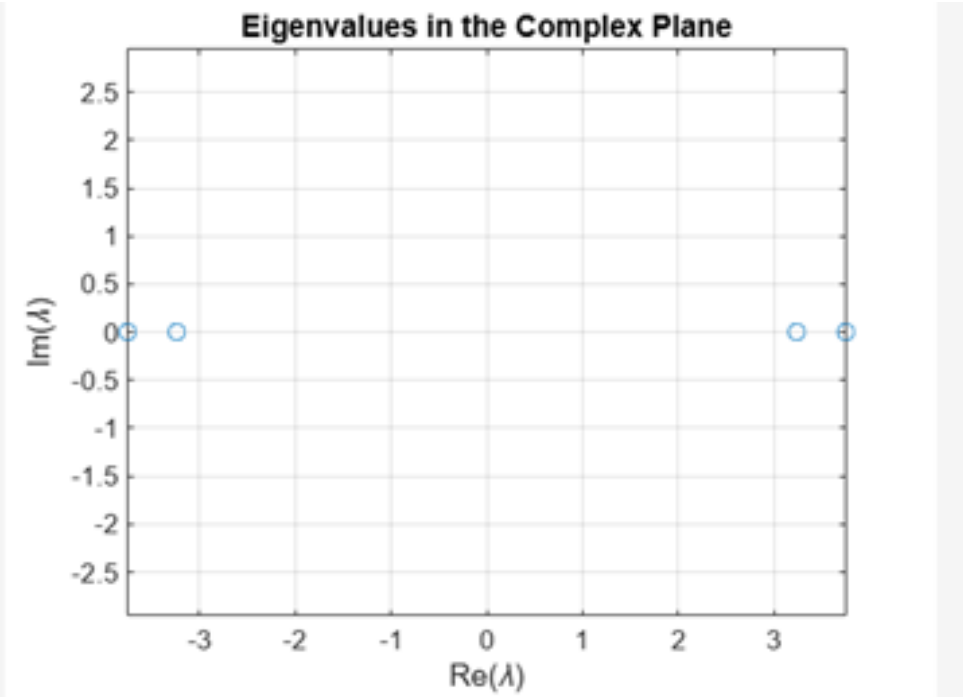


Figure 4.1

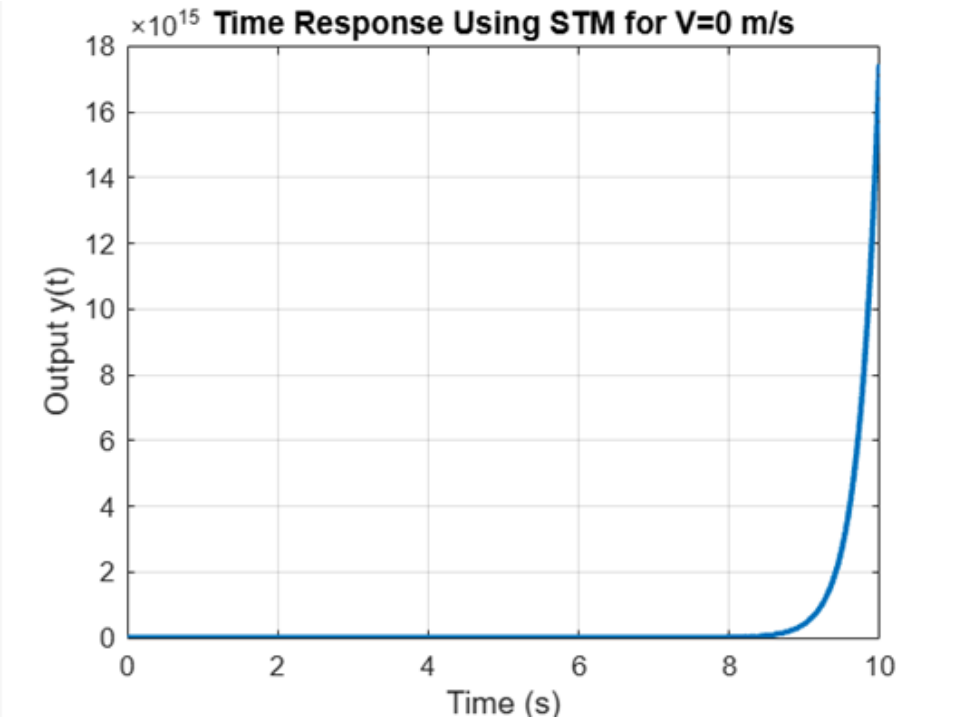


Figure 4.2

- **Asymptotic stability**: No, because two eigenvalues lie on the right half-plane.Hence system is not asymptotically stable
- **Marginal stability**: No, not all eigenvalues lie on the left half-plane, and there is no eigenvalue pair exclusively on the imaginary axis.Hence system is not Marginally

stable

- **BIBO stability**: No, the zero-input response is unbounded.Hence not BIBO stable

4.2 Stability at V = 3.5 m/s

Eigen values= -8.0753 , -0.3965 , -0.2301+ 3.3809i , -0.2301 - 3.3809i.

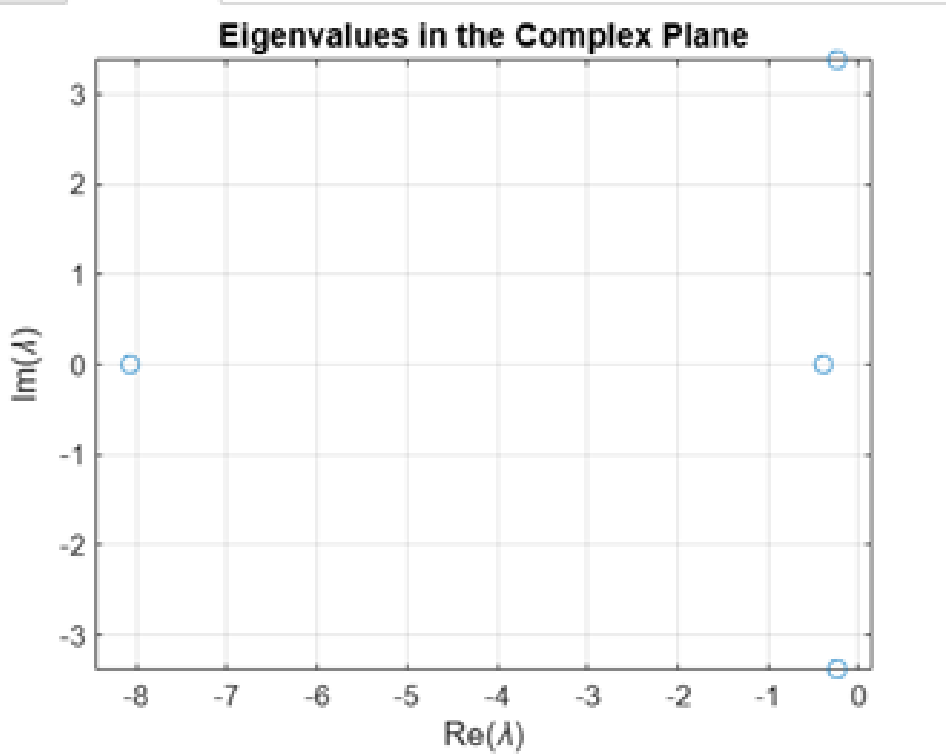


Figure 4.3

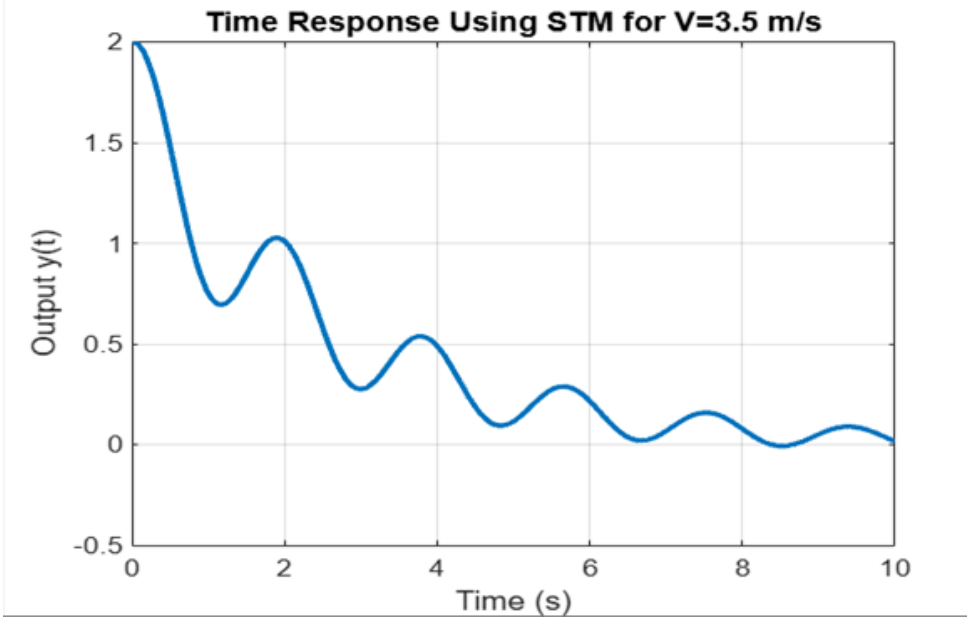


Figure 4.4

- **Asymptotic stability**: Yes, because all eigenvalues lie on the left half-plane.
- **Marginal stability**: No, no eigenvalue pair lies on the imaginary axis.
- **BIBO stability**: Yes, the zero-input response is bounded.

4.3 Stability at V = 5 m/s

Eigen values= -10.8598 ,-1.033+ 6.4842i ,-1.033- 6.4842i , 0.1658.

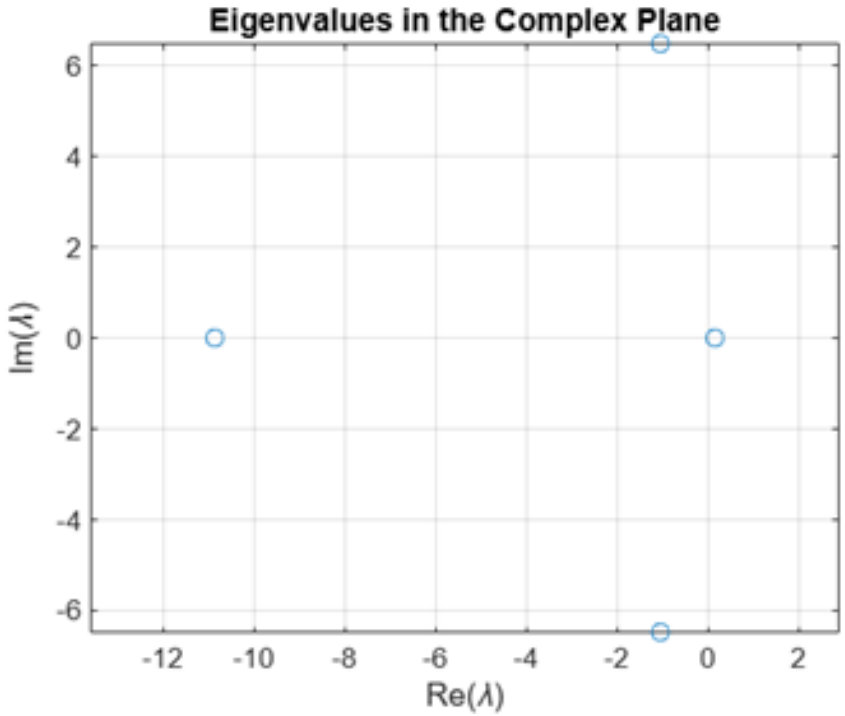


Figure 4.5

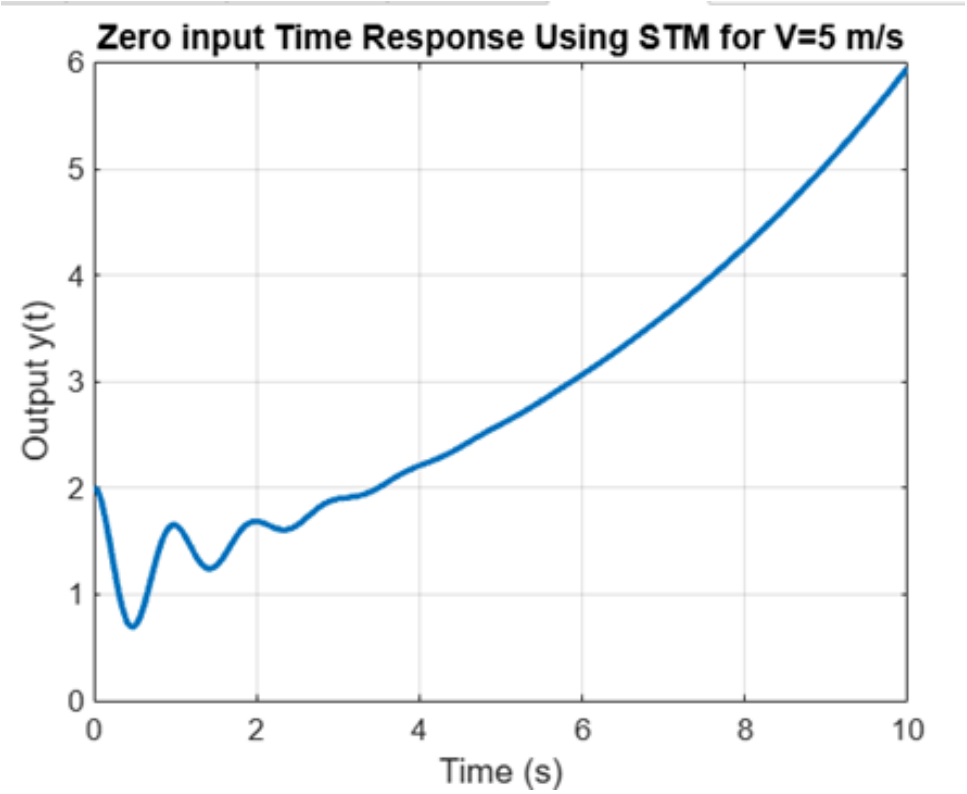


Figure 4.6

- **Asymptotic stability**: No, because one eigenvalue (0.1658) lies on the right half-plane.

- **Marginal stability**: No, no eigenvalue pair lies on the imaginary axis.
- **BIBO stability**: No, the zero-input response is unbounded.

Chapter 5

Controller Design

5.1 Controllability Analysis

Before designing controllers, the controllability of the system is verified for each velocity. The controllability matrix is computed as:

$$\mathcal{C} = [B \; AB \; A^2B \; A^3B] \tag{5.1}$$

```
% Define your system matrices A and B
A1 = [0, 0, 1, 0;
     0, 0, 0, 1;
     13.67, 0.225, 0, 0;
     4.857, 10.81, 0, 0];
B = [0; 0; 0; 1];

% Calculate the controllability matrix
ControllabilityMatrix = [B, A1*B, (A1^2)*B, (A1^3)*B];

% Display the controllability matrix
disp('Controllability Matrix:');
disp(ControllabilityMatrix);

% Calculate and display the determinant
determinant = det(ControllabilityMatrix);
disp(['Determinant of the controllability matrix: ', num2str(determinant)]);

```

Figure 5.1: Matlab code to compute controllability matrix

For all three velocities, corresponding A matrix is changed.

5.1.1 Controllability at V = 0 m/s

```
Controllability Matrix:
         0         0         0    0.2250
         0    1.0000         0   10.8100
         0         0    0.2250         0
    1.0000         0   10.8100         0

Determinant of the controllability matrix: -0.050625
```

Figure 5.2: controllability matrix for V=0 m/s

$$\det(\mathcal{C}) \neq 0 \tag{5.2}$$

Therefore, the system is controllable at V = 0 m/s.

5.1.2 Controllability at V = 3.5 m/s

```
>> controllability2_check
Controllability Matrix:
         0         0   -1.9320    1.3239
         0    1.0000   -8.3580   42.3997
         0   -1.9320    1.3239   24.0793
    1.0000   -8.3580   42.3997 -322.1487

Determinant of the controllability matrix: -185.1586
```

Figure 5.3: controllability matrix for V=3.5 m/s

$$\det(\mathcal{C}) \neq 0 \tag{5.3}$$

Therefore, the system is controllable at V = 3.5 m/s.

5.1.3 Controllability at V = 5 m/s

```
>> controllability3_check
Controllability Matrix:
         0         0   -2.7600    2.4676
         0    1.0000  -11.9400   75.2788
         0   -2.7600    2.4676  143.5129
    1.0000  -11.9400   75.2788 -660.8172

Determinant of the controllability matrix: -894.3101
```

Figure 5.4: controllability matrix for V=5 m/s

$$\det(\mathcal{C}) \neq 0 \tag{5.4}$$

Therefore, the system is controllable at V = 5 m/s.

5.2 Controller Design for V = 0 m/s

Controller is designed with the help of SISO tool im Matlab. The following is the matlab code to open it.We need to enter zeros,poles and gain of transfer function correspoing to the velocity.

```
1  a=zpk([-42.5270 -8.3114],[-8.0753 -0.3965 -0.2301+ 3.3809i  -0.2301-3.3809i],[1]);
2  sisotool(a)
3
```

Figure 5.5: Matlab code for opening siso tool

Design specifications:

- Settling time: 10 seconds
- Damping ratio: 0.75

A state feedback controller of the form $u = -Kx$ is designed using pole placement techniques. The controller gain vector K is determined to place the closed-loop poles at desired locations that meet the design specifications.

The uncontrolled system is unstable with poles in the right half-plane. After applying the designed controller, the closed-loop system has all poles in the left half-plane, meeting the specified settling time of 10 seconds.

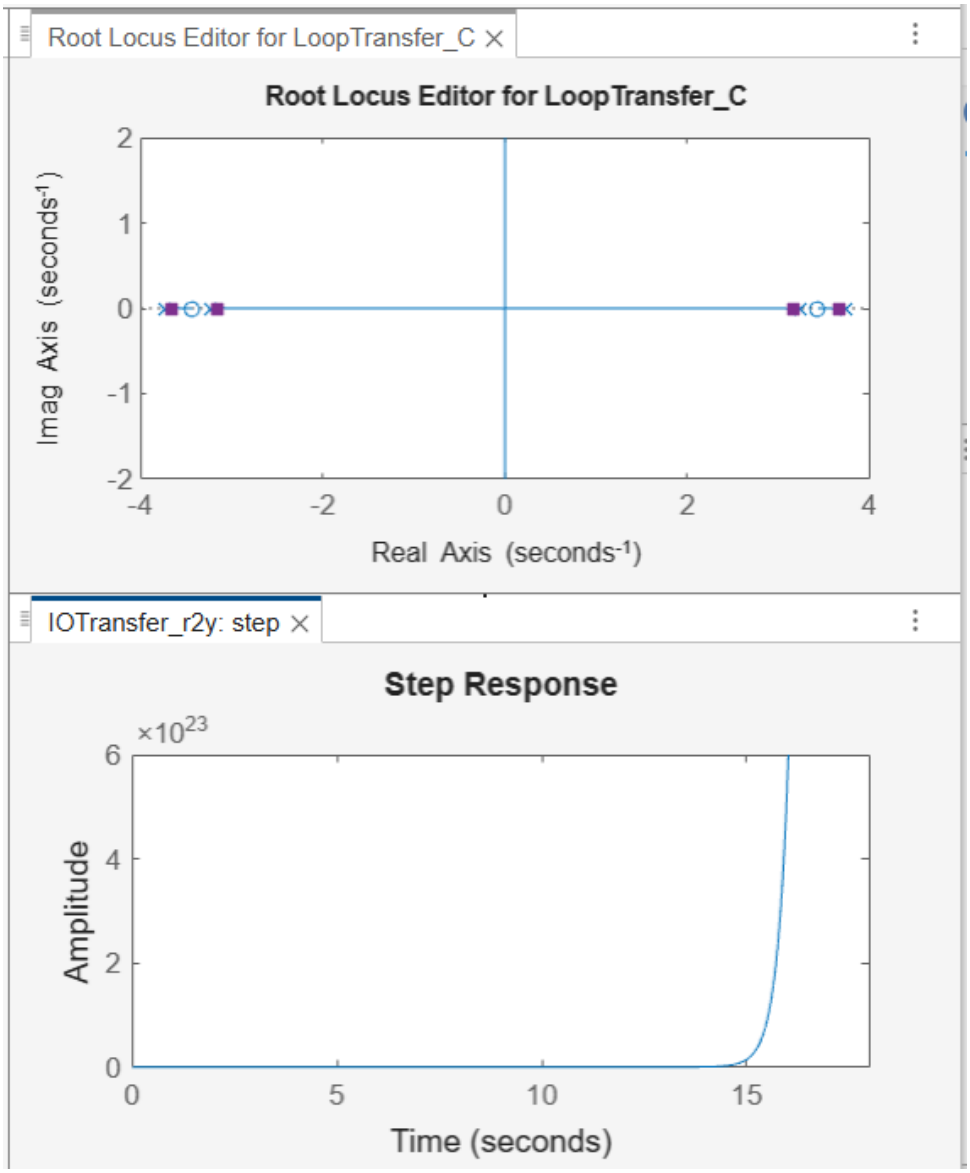


Figure 5.6: uncontrolled response at V = 0 m/s

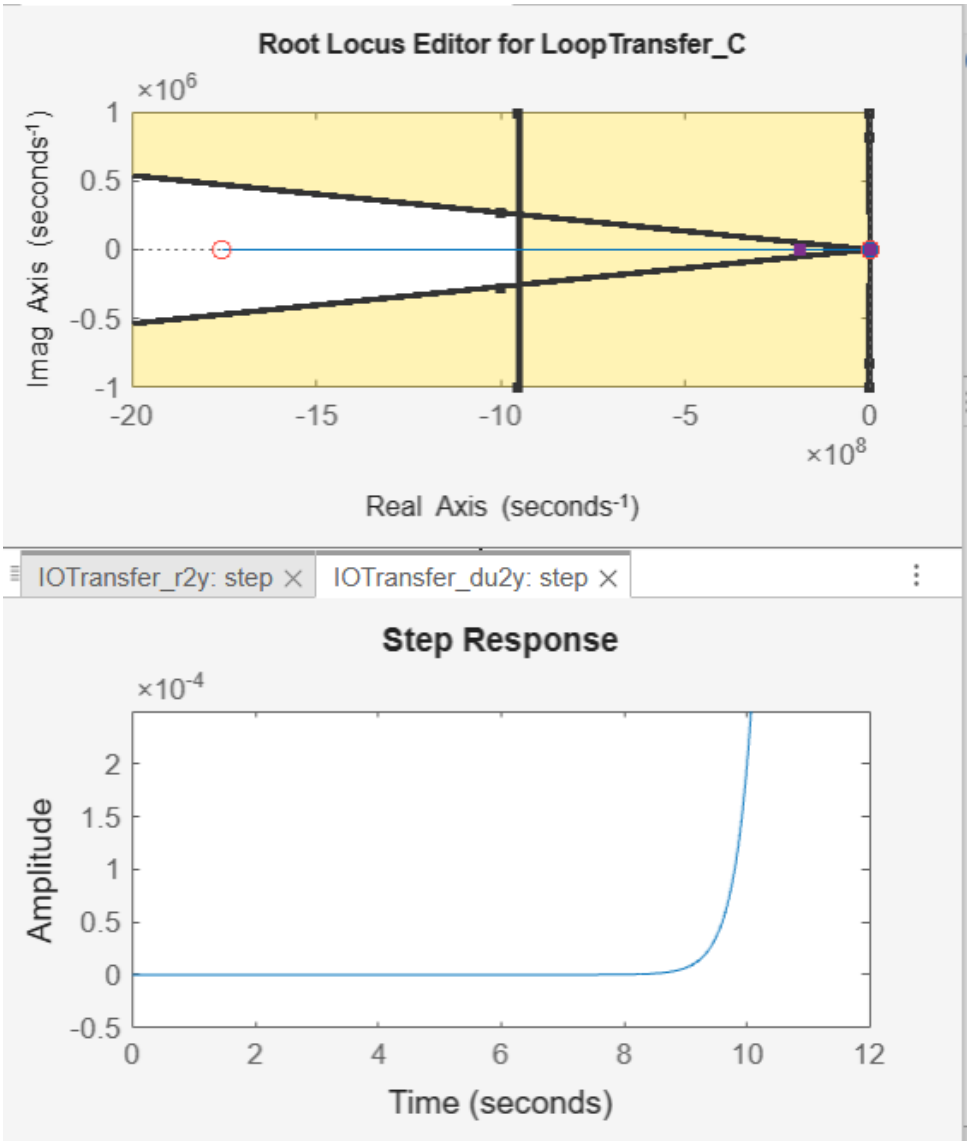


Figure 5.7: controlled response at V = 0 m/s

5.3 Controller Design for V = 3.5 m/s

For V = 3.5 m/s, where the system is inherently stable, a controller is designed to improve performance characteristics such as settling time and overshoot.

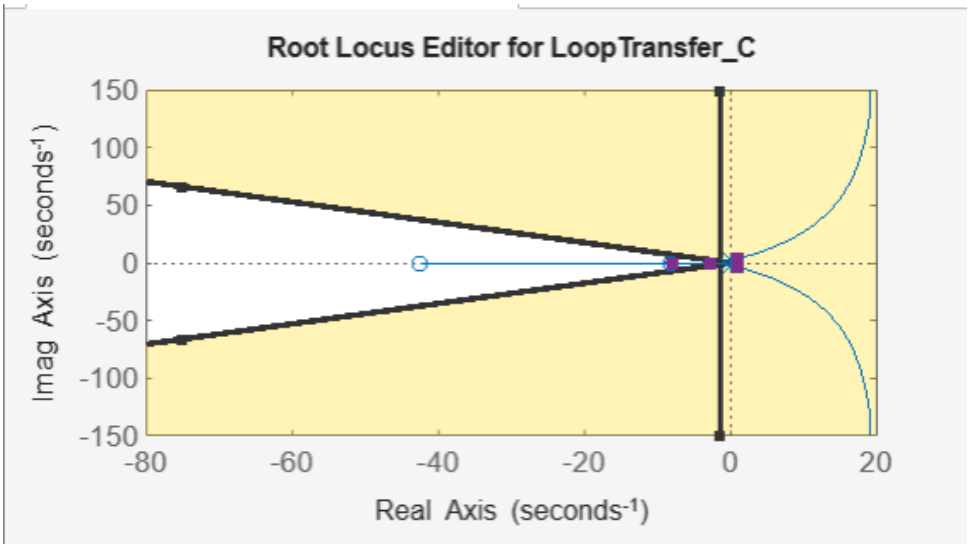


Figure 5.8: uncontrolled response at V = 3.5 m/s

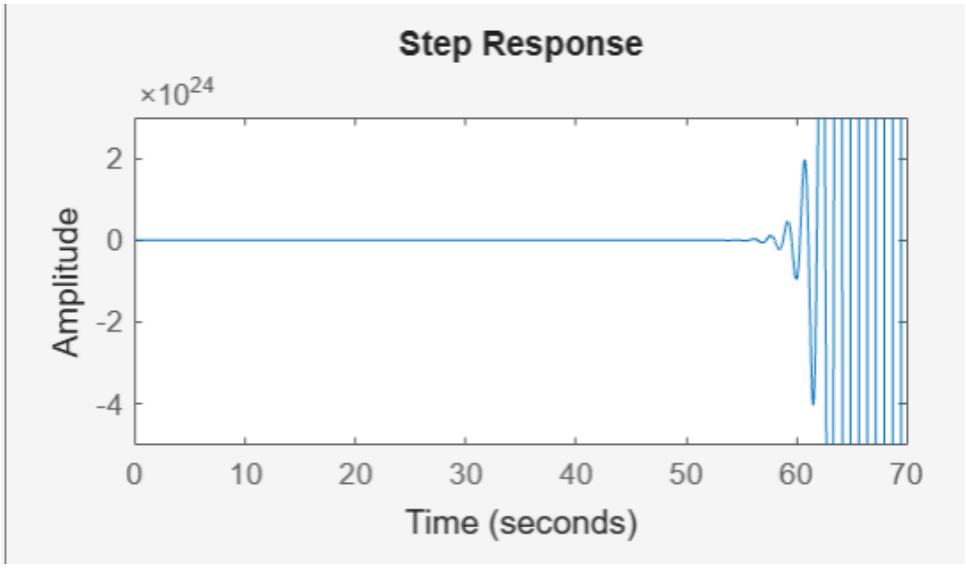


Figure 5.9: uncontrolled response at V = 3.5 m/s

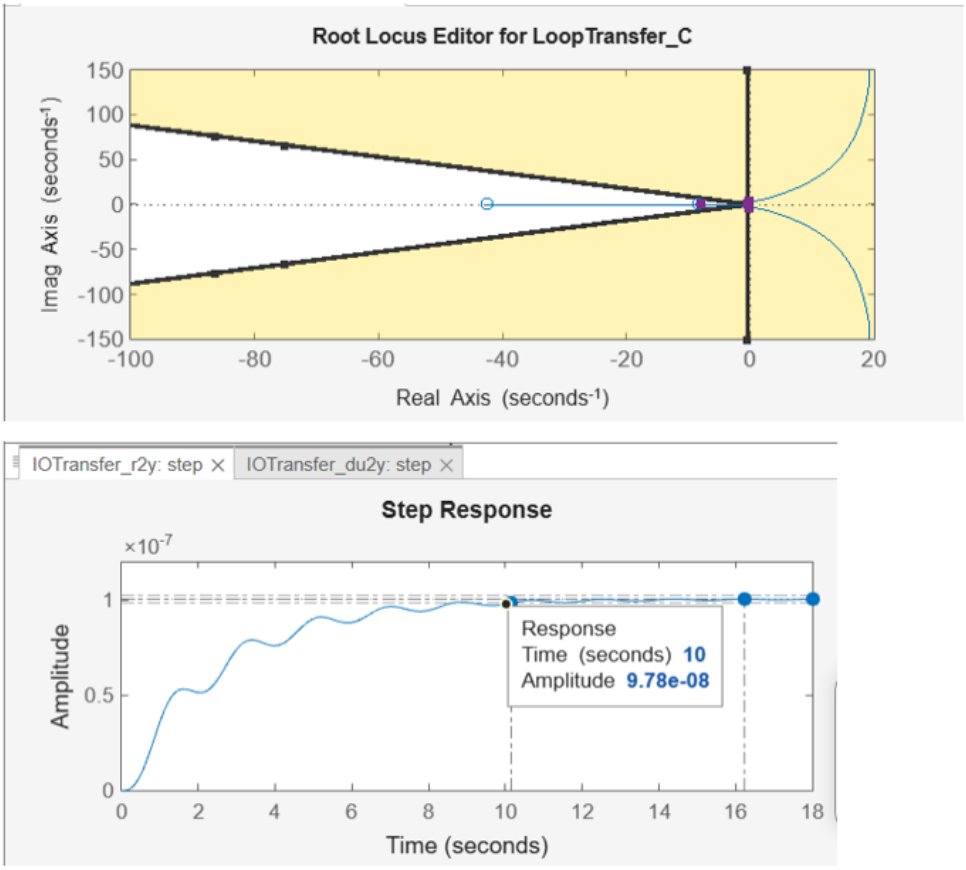


Figure 5.10: controlled response at V = 3.5 m/s

```
>> controller2_design
>> final_control2

ans =

  struct with fields:

        RiseTime: 4.4876
   TransientTime: 10.1480
    SettlingTime: 10.1480
     SettlingMin: 8.8046e-08
     SettlingMax: 1.0051e-07
       Overshoot: 0.1184
      Undershoot: 0
            Peak: 1.0051e-07
        PeakTime: 16.2238
```

Figure 5.11: controlled response at V = 3.5 m/s in output command

The controller successfully meets the desired specifications, with improved settling time and damping ratio compared to the uncontrolled system.

5.4 Controller Design for V = 5 m/s

At V = 5 m/s, the system has one unstable pole at 0.1658. A controller is designed to stabilize the system while meeting performance criteria.

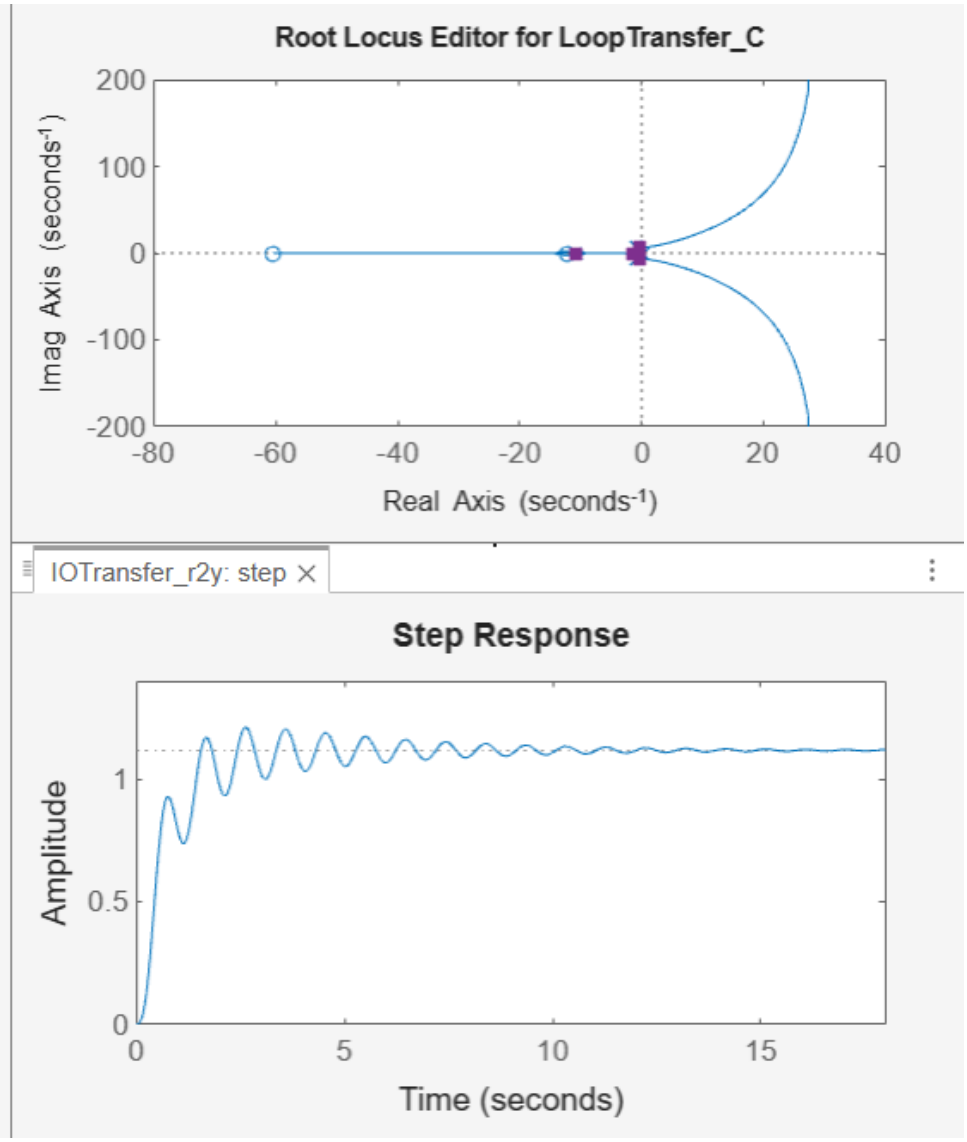


Figure 5.12: uncontrolled response at V = 5 m/s

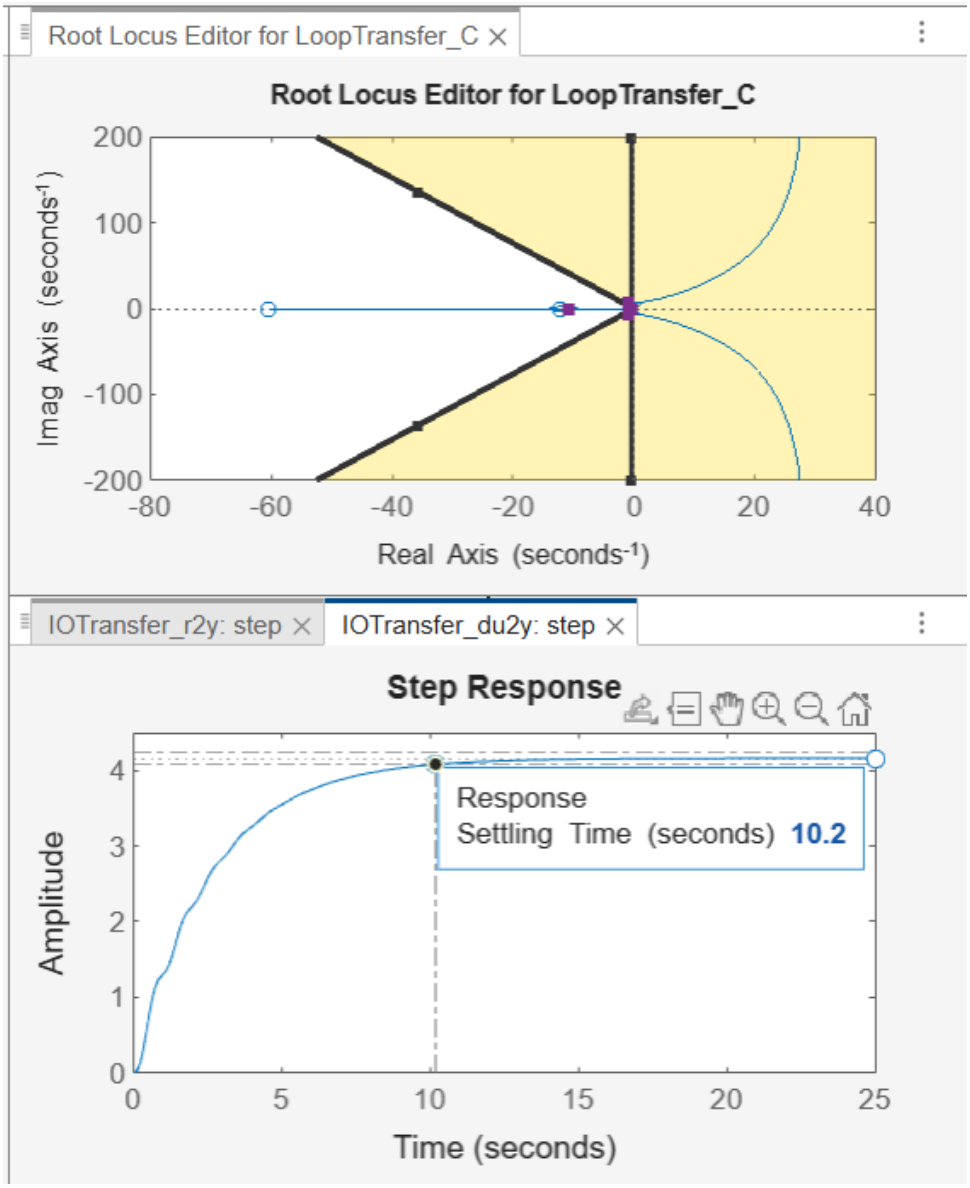


Figure 5.13: controlled response at V = 5 m/s

5.5 Controller Design for V = 3.5 m/s

The controlled system is stable with all poles in the left half-plane, demonstrating that the bicycle can be stabilized at this velocity with appropriate control action.

Chapter 6

Conclusions

Based on the comprehensive analysis and control design for the riderless bicycle system, the following conclusions can be drawn:

1. **Velocity-dependent stability**: The bicycle exhibits different stability characteristics at different velocities. At V = 0 m/s and V = 5 m/s, the system is inherently unstable, while at V = 3.5 m/s, it is naturally stable.

2. **Controllability**: Despite the varying stability characteristics, the bicycle system is controllable at all three velocities, enabling the design of stabilizing controllers.

3. **Controller effectiveness**: The designed controllers successfully stabilize the bicycle at all velocities, transforming unstable systems into stable ones.

4. **Control challenges**: Stabilizing the bicycle at V = 0 m/s (stationary) is particularly challenging due to the presence of right-half-plane poles.

5. **LPV modeling approach**: The linear parameter-varying (LPV) modeling approach effectively captures the velocity-dependent dynamics of the bicycle, facilitating the analysis and control design at different operating points.

This project demonstrates that an automatic control system can maintain a riderless bicycle in an upright position across different velocities, including the challenging cases of a stationary bicycle and high-speed operation.