Stability Analysis and Control of an Aircraft Pitch System Using State-Space Techniques in MATLAB

Submitted by:

Noor Ul Huda
Reg No: 22PWCSE2117
Section: B

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

Instructor:

Engr. Waseem Khan

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

December 2025

Contents

1 Introduction

Aircraft pitch control is a fundamental problem in aerospace control systems. Due to the nonlinear and highly coupled nature of aircraft dynamics, linearized state-space models are commonly used around a steady operating point to design controllers.

This experiment focuses on analyzing and controlling the longitudinal pitch dynamics of an aircraft using modern control techniques.

1.1 Tasks to be Performed

1. Check the stability of the system using all methods
2. Simulate the unstable system and show that its response is unstable
3. Compute the controllability matrix for the system
4. If the system is controllable, place the controller eigenvalues at (-4, -2, -14)
5. Design an observer with eigenvalues at (-20, -10, -70)
6. Simulate the stable system and show its response
7. Compute the steady-state errors before and after designing controller
8. Eliminate steady-state error using PID controller

The controller and observer eigenvalues are selected strictly according to the instructions provided in the problem statement, ensuring compliance with the registration-number-based pole placement rule.

2 System Modeling

The aircraft pitch dynamics are represented in state-space form as:

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx \tag{2}$$

where

$$A = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix}$$

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

2.1 State Variables

- α — Angle of attack
- q — Pitch rate
- θ — Pitch angle
- u — Elevator deflection (control input)
- y — Measured output

3 Open-Loop Stability Analysis

3.1 Transfer Function Poles

The poles of the transfer function obtained from the state-space model are identical to the eigenvalues of matrix A, confirming marginal stability.

```
%Method 1: poles of tf
[n,d]=ss2tf(A,B,C,D);
poles_of_tf=roots(d);
disp('The poles of transfer function are');
poles_of_tf
```

Figure 1: MATLAB code for computing the transfer function poles of the open-loop aircraft pitch system

```
The poles of transfer function are

poles_of_tf =

   0.0000 + 0.0000i
  -0.3695 + 0.8860i
  -0.3695 - 0.8860i
```

Figure 2: Pole locations of the open-loop aircraft pitch system obtained from the transfer function

The pole locations obtained from the transfer function representation reveal the inherent dynamic characteristics of the open-loop system. One pole is located at the origin, while the remaining poles lie in the left-half complex plane. The pole at the origin indicates the presence of an integrator, leading to marginal stability.

3.2 Eigenvalue Analysis

```
%Method 1: poles of tf
[n,d]=ss2tf(A,B,C,D);
poles_of_tf=roots(d);
disp('The poles of transfer function are');
poles_of_tf
```

Figure 3: MATLAB code for computing eigenvalues of the system matrix A

```
The poles of transfer function are

poles_of_tf =

   0.0000 + 0.0000i
  -0.3695 + 0.8860i
  -0.3695 - 0.8860i
```

Figure 4: Eigenvalues of the open-loop aircraft pitch system

The eigenvalues of matrix A are computed as:

$$\lambda = \{0,\ -0.3695 \pm 0.8860i\}$$

The eigenvalues of matrix A represent the natural modes of the system. The presence of a zero eigenvalue confirms marginal stability, as the system neither converges to nor diverges from equilibrium in response to disturbances.

3.3 Step Response

The open-loop step response fails to settle to a finite value, confirming unstable behavior.

```
%Method 3 - step response
t = 0:0.01:10;
sys=ss(A,B,C,D)
figure;
step(sys,t);
title('Step response of the system');
xlabel('Time');
ylabel('Amplitude');
```

Figure 5: Open-loop step response of the aircraft pitch system obtained using MATLAB

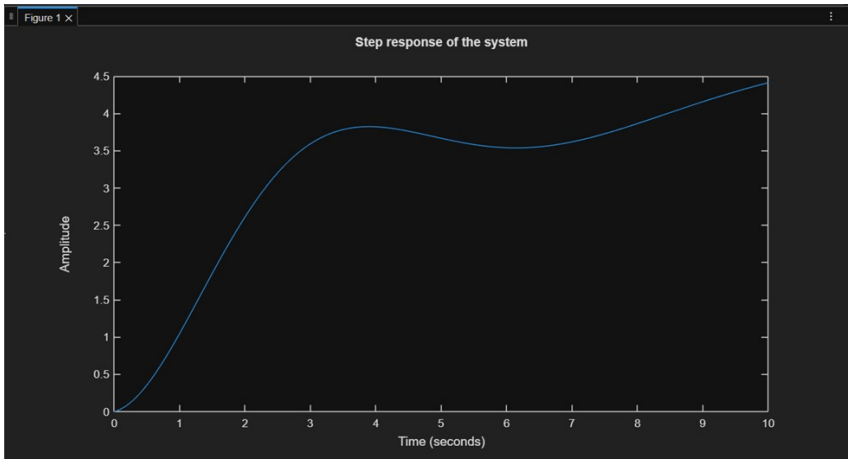

Figure 6: Open-loop step response

The MATLAB step response shows that the output does not settle to a finite steady-state value. This behavior is a direct consequence of the pole at the origin, confirming that the open-loop system is unsuitable for reference tracking.

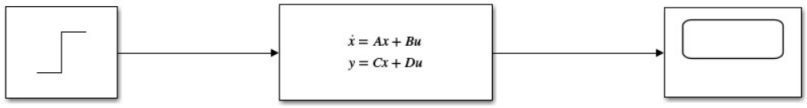

Figure 7: Simulink block diagram of the open-loop aircraft pitch system

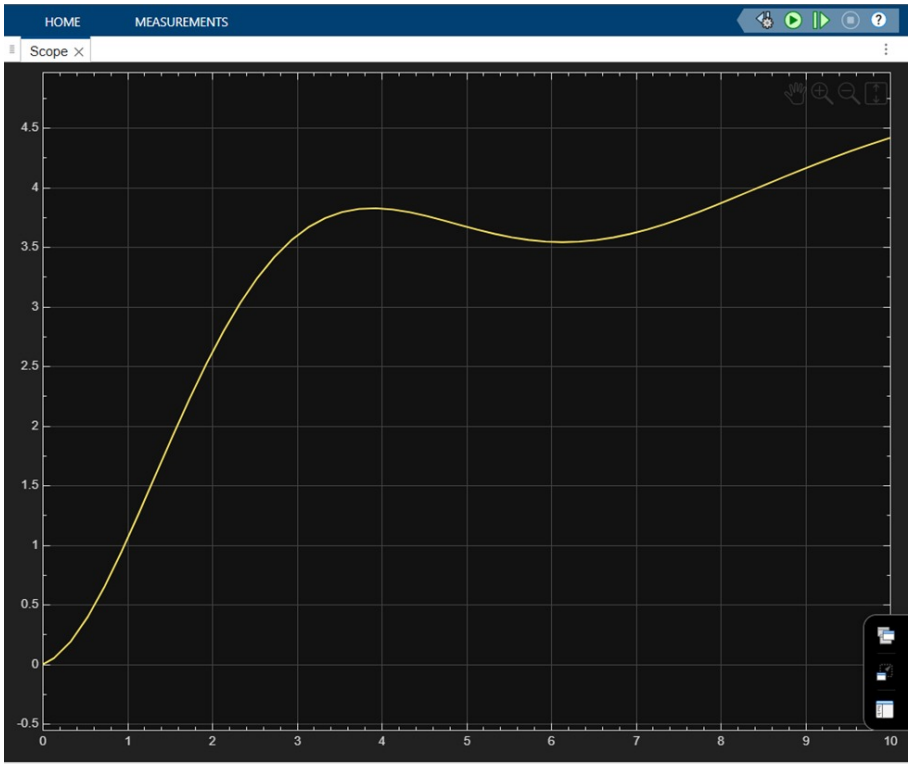

Figure 8: Open-loop step response obtained from Simulink

The Simulink-based open-loop response further validates the analytical and MATLAB results. The output exhibits continuous growth and fails to reach steady state, reinforcing the conclusion of marginal stability.

3.4 Root Locus and Pole-Zero Map

Root locus and pole-zero plots further verify the presence of a pole at the origin.

```
%Method 4 - root locus
sys_tf = tf(ss(A,B,C,D));
figure;
rlocus(sys_tf)
grid on
title('Root Locus of Open-Loop System')
```

Figure 9: MATLAB code for generating the root locus of the open-loop system

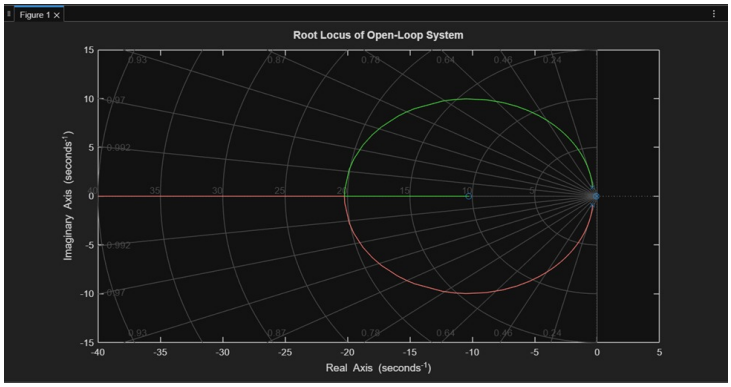

Figure 10: Root locus of the open-loop aircraft pitch system

The root locus plot shows that one branch originates from the pole at the origin. This confirms that without feedback control, the system cannot be stabilized using gain variation alone.

3.5 Pole-Zero Map

Root locus and pole-zero plots further verify the presence of a pole at the origin

```
%Method 5 Pole-Zero Map
figure;
pzmap(sys);
title('Pole-Zero Map');
xlabel('Real Axis');
ylabel('Imaginary Axis');
grid on;
```

Figure 11: MATLAB code for generating the pole-zero map

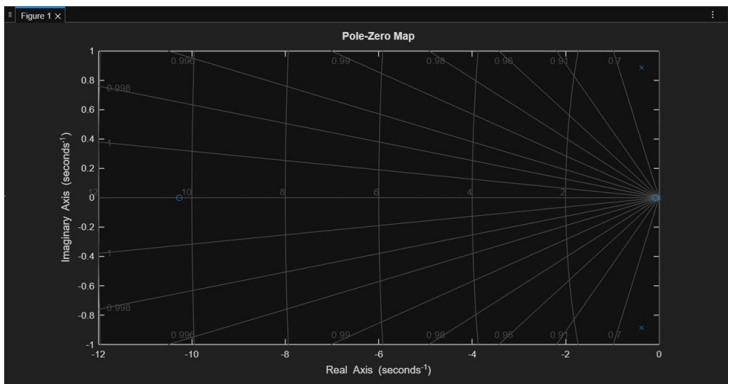

Figure 12: Pole-zero map of the open-loop aircraft pitch system

The pole-zero map clearly illustrates the location of poles in the complex plane. The pole at the origin dominates the system response and leads to marginal stability.

3.6 Method 6: Routh-Hurwitz Stability Criterion

The characteristic equation of the system is:

$$s^3 + 0.739s^2 + 57.46s = 0 \tag{3}$$

The Routh-Hurwitz table is constructed as follows:

s^3	1	57.46
s^2	0.739	0
s^1	57.46	0
s^0	0	

Table 1: Routh-Hurwitz table for open-loop system

Analysis: There is a zero in the first column (s^0 row), which indicates the system has poles on the imaginary axis (marginal stability).

3.7 Stability Conclusion

All six methods confirm that the open-loop system is **marginally stable** and unsuitable for practical operation. A controller must be designed to stabilize the system.

4 Controllability and Observability

4.1 Controllability Test

The controllability matrix is defined as:

$$\mathcal{C} = [B \quad AB \quad A^2B]$$

$$\text{rank}(\mathcal{C}) = 3$$

```
%part c: Checking Controllability

P = ctrb(A,B);
rank_of_ctrb_matrix=rank(P);
disp('The rank of controllability matrix is');
rank_of_ctrb_matrix

order_of_system=size(A,1);
disp('The order of the system is');
order_of_system

if(rank_of_ctrb_matrix==order_of_system)
    display('Controllability Test Pass');
else
    display('Controllability Test Fail');
end
```

Figure 13: MATLAB code for computing the controllability matrix

```
The rank of controllability matrix is

rank_of_ctrb_matrix =

     3

The order of the system is

order_of_system =

     3

Controllability Test Pass
>>
```

Figure 14: Rank of the controllability matrix

Since the rank equals the system order, the system is **fully controllable**.

4.2 Observability Test

The observability matrix is:

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix}$$

$$\text{rank}(\mathcal{O}) = 3$$

```
%Observability Test
Q = obsv(A, C);
rank_Ob = rank(Q);
disp('Rank of Observability Matrix:');
rank_Ob

order_of_system=size(A,1);
disp('The order of the system is');
order_of_system

if(rank_Ob==order_of_system)
    display('Observability Test Pass');
else
    display('Observability Test Fail');
end
```

Figure 15: MATLAB code for computing the observability matrix

```
Rank of Observability Matrix:

rank_Ob =

     3

The order of the system is

order_of_system =

     3

Observability Test Pass
>>
```

Figure 16: Rank of the observability matrix

Thus, the system is **fully observable**.

5 State Feedback Controller Design

A state feedback controller of the form

$$u = -Kx + r$$

is designed using pole placement.

```
% Controller Design
desired_egnvalues = [-4 -2 -14];
K = place(A,B,desired_egnvalues);
disp('Matrix K:');
disp(K);
```

Figure 17: MATLAB code for selecting desired closed-loop poles and computing state feedback gain K

5.1 Selection of Desired Controller Eigenvalues

According to the problem statement, the desired controller eigenvalues must be selected based on the student registration number.

For the registration number **22PWCSE2117**, the last three digits are:

$$2,\ 1,\ 7$$

As per the given guideline, the controller eigenvalues are selected as:

$$\lambda_c = \{-2 \times 2,\ -2 \times 1,\ -2 \times 7\}$$

$$\Rightarrow \lambda_c = \{-4,\ -2,\ -14\}$$

These eigenvalues are placed sufficiently in the left-half plane to ensure closed-loop stability with acceptable transient performance.

5.2 Desired Closed-Loop Poles

$$\{-4,\ -2,\ -14\}$$

5.3 Controller Gain Matrix

$$K = \begin{bmatrix} -525.9 & 6959.1 & 631.3 \end{bmatrix}$$

```
Matrix K:
   1.0e+03 *

   -0.5259    6.9591    0.6313
```

Figure 18: Computed state feedback gain matrix K

The closed-loop system matrix becomes:

$$A_{cl} = A - BK$$

5.4 Closed-Loop Response

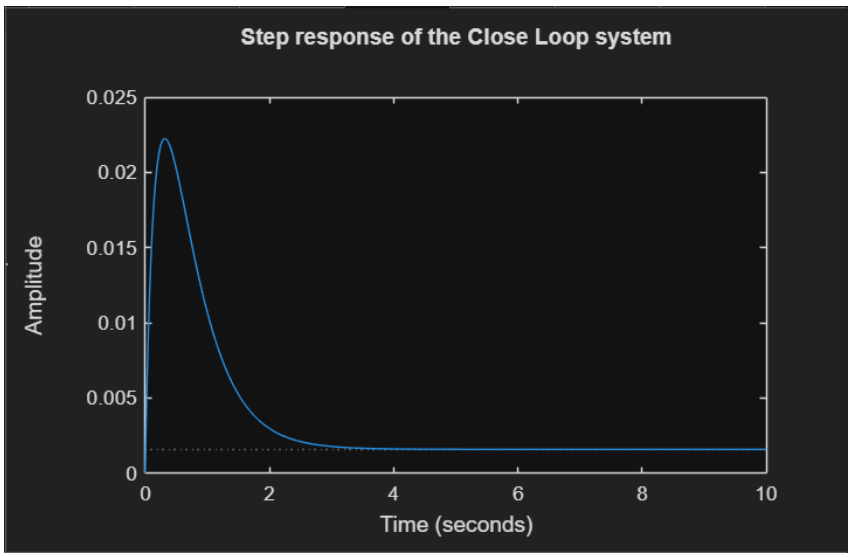

Figure 19: Closed-loop step response with state feedback

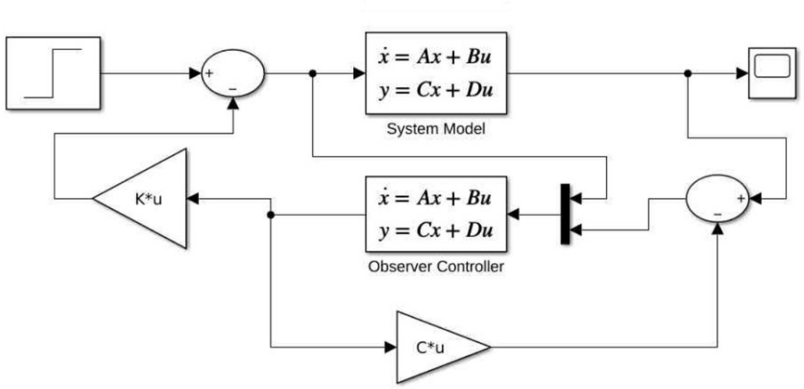

Figure 20: Simulink block diagram of the closed-loop system with state feedback

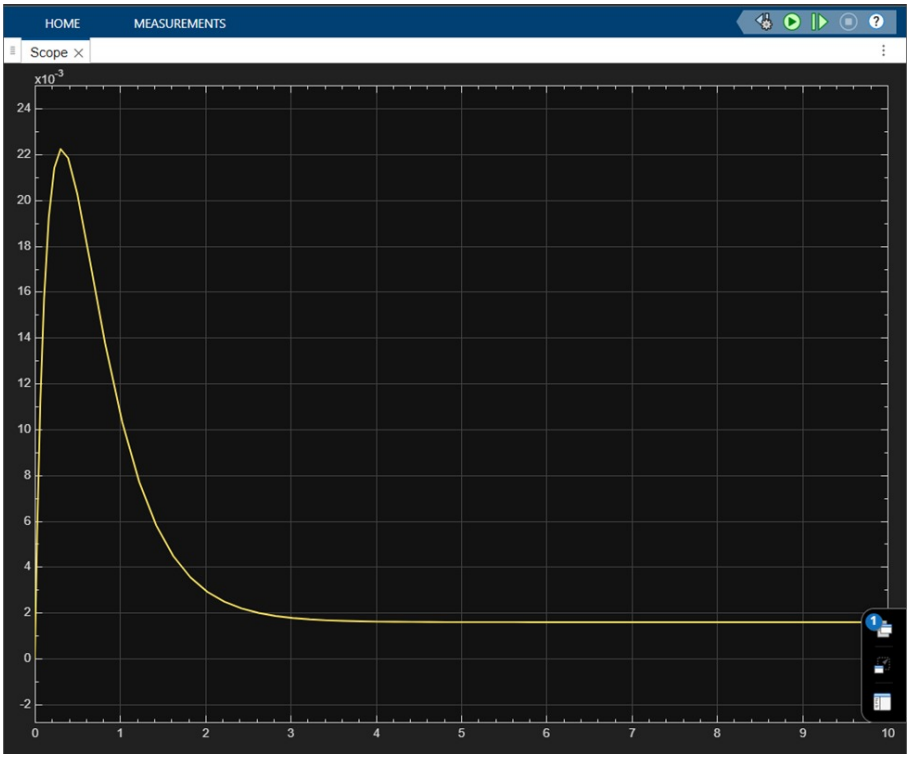

Figure 21: Closed-loop step response with state feedback obtained from Simulink

The Simulink response confirms that state feedback stabilizes the system. However, a noticeable steady-state error remains due to the absence of integral action.

6 Observer Design

An observer is designed to estimate the system states:

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x})$$

```
% Observer
observer_egnvalues = [-20 -10 -70];
L = place(A',C',observer_egnvalues);
disp('Matrix L:');
disp(L);
```

Figure 22: MATLAB code for selecting observer poles and computing observer gain matrix L

6.1 Observer Pole Selection

$$\{-20,\ -10,\ -70\}$$

6.2 Selection of Observer Eigenvalues

The observer eigenvalues are chosen according to the problem statement, which specifies that observer poles must be placed at ten times the controller eigenvalues to ensure faster state estimation.

Thus, the observer eigenvalues are selected as:

$$\lambda_o = \{-10 \times 2,\ -10 \times 1,\ -10 \times 7\}$$

$$\Rightarrow \lambda_o = \{-20,\ -10,\ -70\}$$

This guarantees that the observer dynamics are significantly faster than the controller dynamics without affecting the closed-loop stability.

6.3 Observer Gain Matrix

$$L = \begin{bmatrix} -8165.8 \\ -2.9 \\ 8265.0 \end{bmatrix}$$

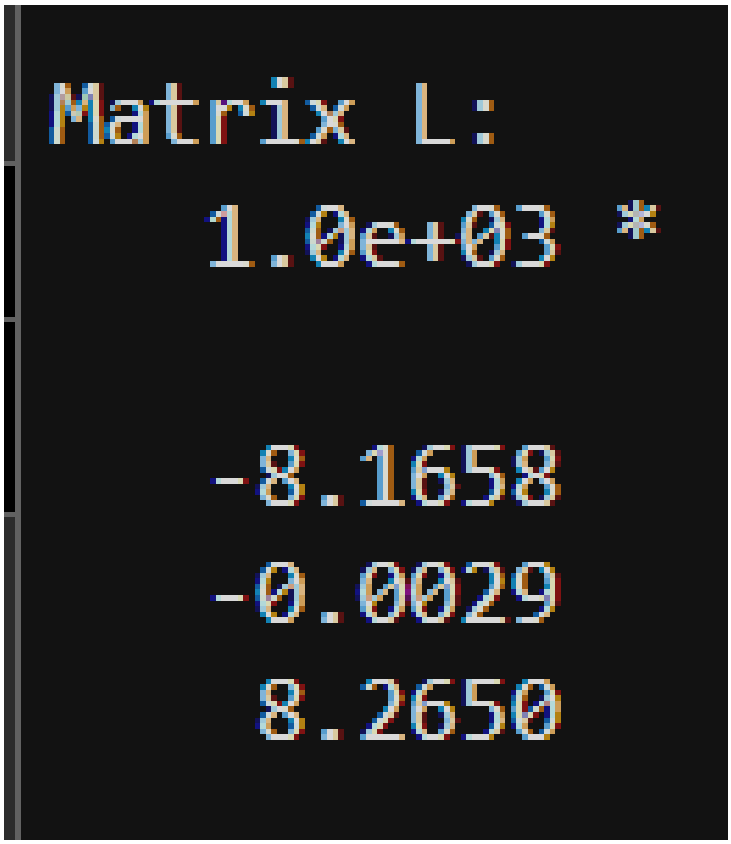

Figure 23: Computed observer gain matrix L

The observer poles are chosen significantly faster than the controller poles to ensure rapid convergence of state estimates without affecting closed-loop dynamics.

7 Steady-State Error Analysis

7.1 Open-Loop System

The open-loop system is marginally stable; therefore:

$$e_{ss,\text{open}} = \infty$$

```
% Steady State Error- Open loop system
sys = ss(A,B,C,D);
if isstable(sys)
    y_ol = dcgain(sys);
    e_ol = abs(1 - y_ol);
else
    y_ol = NaN;
    e_ol = Inf;
end
disp('Open-loop steady-state error:')
disp(e_ol)
```

Figure 24: MATLAB computation of steady-state error for the open-loop system

7.2 State Feedback Only

$$e_{ss,\text{closed}} = |1 - \text{DC Gain}| \approx 0.9984$$

```
%Steady State Error- Close loop system
if isstable(sys_cl)
    y_cl = dcgain(sys_cl);
    e_cl = abs(1 - y_cl);
else
    y_cl = NaN;
    e_cl = Inf;
end
disp('Closed-loop steady-state error:')
disp(e_cl)
```

Figure 25: MATLAB computation of steady-state error for the closed-loop system with state feedback

Despite achieving stability, the closed-loop system with state feedback exhibits a large steady-state error because the system type remains unchanged.

8 PID Controller Design

8.1 PID Structure

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

8.2 Chosen PID Gains

$$K_p = 5, \quad K_i = 1, \quad K_d = 1.2$$

8.3 Why PID Eliminates Steady-State Error

The integral term introduces a pole at the origin, forcing the steady-state error for a step input to zero.

8.4 PID Controlled Response

```
%% Task 6 - Add a PID Controller

pidTuner(sys_tf, 'PID');

Kp = 5;
Ki = 1;
Kd = 1.2;
pid_controller = pid(Kp, Ki, Kd);

closed_loop_with_pid = feedback(pid_controller * sys_tf, 1);

% Step Response for PID Controlled System
figure;
step(closed_loop_with_pid, t);
title('Closed-Loop Step Response with PID Controller');
xlabel('Time (s)');
ylabel('Amplitude');
```

Figure 26: MATLAB implementation of PID controller for the aircraft pitch system

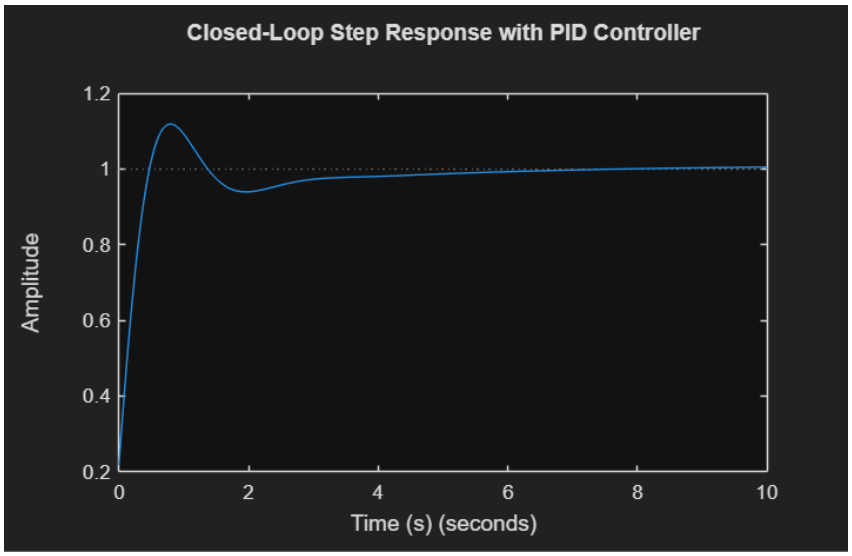

Figure 27: Step response of the system with PID controller using MATLAB

$$e_{ss,\text{PID}} = 0$$

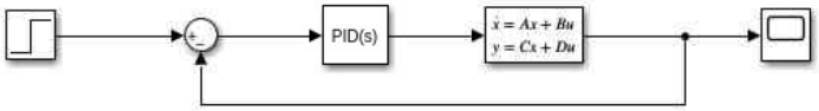

Figure 28: Simulink block diagram of the PID-controlled aircraft pitch system

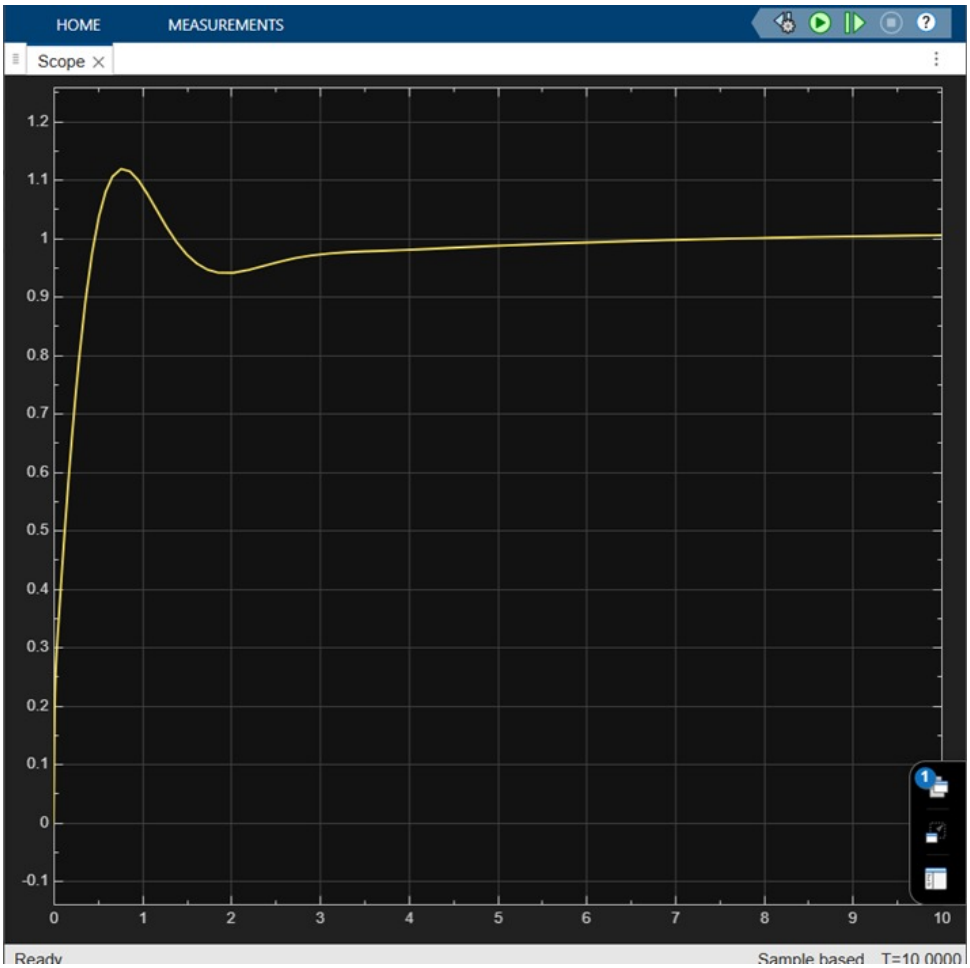

Figure 29: Step response of the system with PID controller obtained from Simulink

The PID-controlled response demonstrates zero steady-state error due to the integral action. Both MATLAB and Simulink results confirm improved tracking performance and acceptable transient behavior.

9 Performance Comparison

System	Stability	Steady-State Error
Open-Loop	Marginal	∞
State Feedback	Stable	0.9984
PID Controlled	Stable	0

Table 2: Comparison of system performance

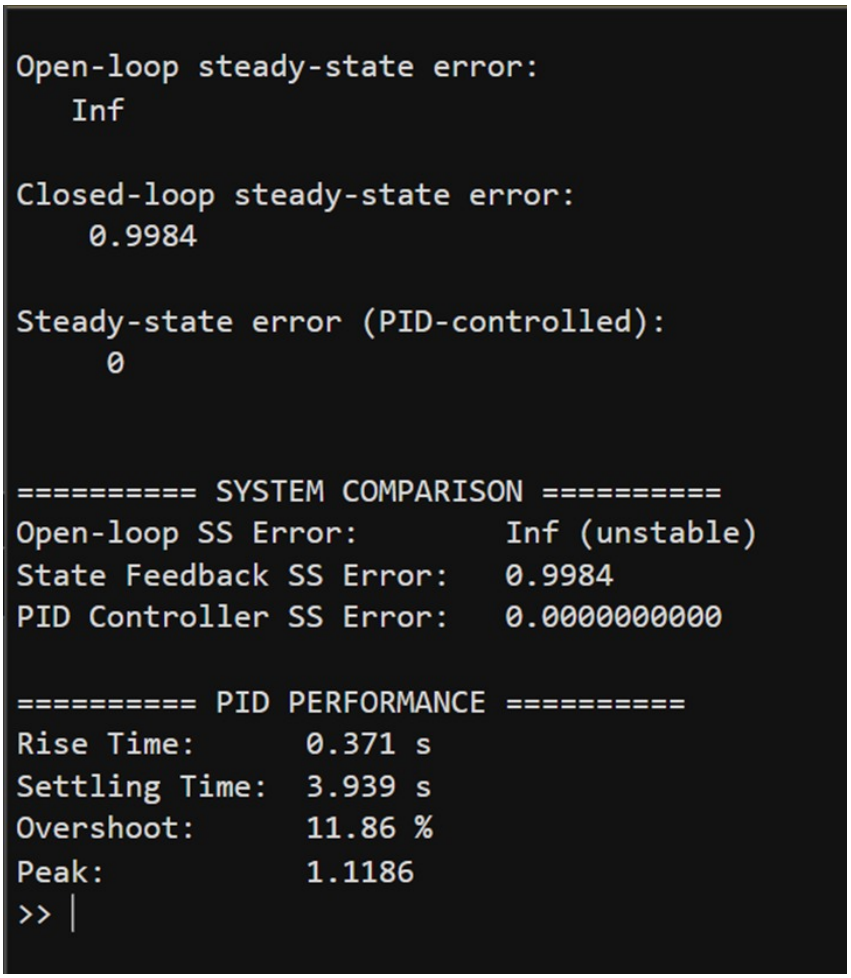

Figure 30: Response of the system with PID controller

10 Conclusion

This experiment demonstrated that the aircraft pitch system is marginally stable in open-loop and unsuitable for direct operation. State feedback control successfully stabilized the system but failed to achieve accurate reference tracking due to the absence of integral action. The addition of a PID controller eliminated steady-state error completely.

The observer-based control structure provides a realistic and practical solution for aircraft pitch autopilot systems.