Aircraft Pitch System Control

Course Project

Final Report

May 16, 2025

Team Members:

Walid Sherif	202200702
Abdelrhman Alnaqeeb	202200281

Supervisors:

Dr. Mohamed L.Shaltot
Eng. Somia Talat

Contents

Abstract

This project investigates the design and implementation of an aircraft pitch control system utilizing classical control theory principles. The research begins with the development of a mathematical model for aircraft pitch dynamics through linearized equations of motion, resulting in a third-order transfer function representation. A comprehensive open-loop analysis is conducted employing pole-zero mapping, Routh-Hurwitz stability criterion, and root locus techniques to characterize the system's inherent behavior and stability properties.

The control system design phase focuses on meeting stringent performance requirements including constrained overshoot, accelerated rise time, limited settling time, and minimal steady-state error. A systematic approach to PID controller synthesis is implemented, involving progressive parameter tuning and performance evaluation. Multiple controller configurations are examined to determine the optimal gain parameters that satisfy all design specifications simultaneously.

The designed control system is implemented and validated using MATLAB and Simulink environments, with simulation results demonstrating the effectiveness of the proposed control strategy.

Chapter 1

Introduction

1.1 Aircraft Pitch Motion

Aircraft pitch refers to the rotation of an aircraft about its lateral axis, which affects the angle between the aircraft's longitudinal axis and the horizon. Controlling pitch is crucial for maintaining the desired flight altitude, executing takeoffs and landings, and ensuring passenger comfort. The primary control surface used for pitch control is the elevator, located on the horizontal stabilizer at the tail of the aircraft.

In the context of flight dynamics, pitch motion is part of the aircraft's longitudinal dynamics, along with changes in forward speed and vertical position. The pitch angle θ and the rate of change of this angle q are key variables that define the aircraft's orientation in the vertical plane. Another important parameter is the angle of attack α, which is the angle between the aircraft's reference line and the oncoming airflow.

When the pilot or autopilot system commands a pitch change, the elevator deflection δ creates an aerodynamic moment that rotates the aircraft about its lateral axis. This change in orientation affects the lift forces and ultimately determines the aircraft's flight path.

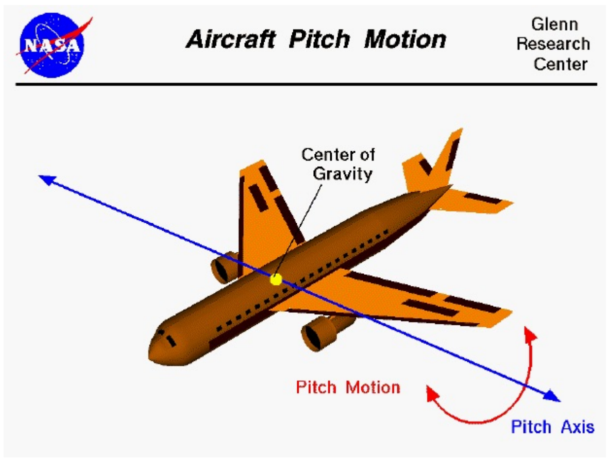


Figure 1.1: Aircraft Pitch Motion (Source: NASA Glenn Research Center)

1.2 Project Objectives

The main objectives of this project are:

1. Develop a mathematical model of aircraft pitch dynamics using transfer function representation.
2. Analyze the open-loop system characteristics using classical control techniques.
3. Design a PID controller to stabilize the pitch angle and achieve specified performance criteria.
4. Compare the performance of different controllers under various operating conditions.
5. Implement and simulate the control system using MATLAB and Simulink.

Chapter 2

System Modeling

We will assume that the aircraft is in steady-cruise at constant altitude and velocity; thus, the thrust, drag, weight and lift forces balance each other in the x- and y-directions. We will also assume that a change in pitch angle will not change the speed of the aircraft under any circumstance (unrealistic but simplifies the problem a bit). Under these assumptions, the longitudinal equations of motion for the aircraft can be written as follows.

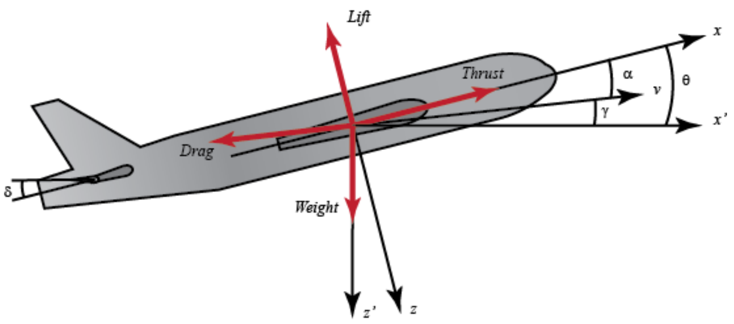


Figure 2.1: Aircraft Forces and Coordinate System (Source: University of Michigan Control Tutorials)

$$\dot{\alpha} = \mu\Omega\sigma[-(C_L + C_D)\alpha + \frac{1}{(\mu - C_L)}q - (C_W \sin\gamma)\theta + C_L] \tag{2.1}$$

$$\dot{\theta} = \Omega q \tag{2.2}$$

Please refer to any aircraft-related textbooks for the explanation of how to derive these equations. You may also refer to the Extras: Aircraft Pitch System Variables page to see a further explanation of what each variable represents.

For this system, the input will be the elevator deflection angle δ and the output will be the pitch angle θ of the aircraft.

2.0.1 Transfer Function

Before finding the transfer function and state-space models, let's plug in some numerical values to simplify the modeling equations shown above:

$$\dot{\alpha} = -0.313\alpha + 56.7q + 0.232\delta \tag{2.3}$$

$$\dot{q} = -0.0139\alpha - 0.426q + 0.0203\delta \tag{2.4}$$

$$\dot{\theta} = 56.7q \tag{2.5}$$

These values are taken from the data from one of Boeing's commercial aircraft.

Transfer Function

To find the transfer function of the above system, we need to take the Laplace transform of the above modeling equations. Recall that when finding a transfer function, zero initial conditions must be assumed. The Laplace transform of the above equations are shown below.

$$sA(s) = -0.313A(s) + 56.7Q(s) + 0.232\Delta(s) \tag{2.6}$$

$$sQ(s) = -0.0139A(s) - 0.426Q(s) + 0.0203\Delta(s) \tag{2.7}$$

$$s\Theta(s) = 56.7Q(s) \tag{2.8}$$

After few steps of algebra, you should obtain the following open loop transfer function.

$$P(s) = \frac{\Theta(s)}{\Delta(s)} = \frac{1.151s + 0.1774}{s^3 + 0.739s^2 + 0.921s} \tag{2.9}$$

Chapter 3

Classical Control Analysis

3.1 Pole-Zero Analysis

Zeros: One zero at $s = -0.1541$

Poles: At $s = 0$, $s = -0.3695 \pm j0.8861$

Interpretation: The pole at $s = 0$ indicates marginal stability; complex poles indicate damped oscillatory behavior.

3.2 Stability Analysis (Routh-Hurwitz)

To analyze the stability of the aircraft pitch system using the Routh-Hurwitz criterion, we examine the characteristic equation derived from the closed-loop transfer function denominator:

$$s^3 + 0.739s^2 + 2.072s + 0.1774 = 0$$

We construct the Routh array step by step as follows:

s^3	1.000	2.072
s^2	0.739	0.1774
s^1	1.832	0
s^0	0.1774	

The first column entries are all positive, indicating no sign changes.

Factoring the polynomial or analyzing roots confirms the system poles lie in the left half-plane.

Thus:

- No poles in the right half-plane $\rightarrow$ system is stable

The system is **stable** under unity feedback. This means any bounded input will produce a bounded output that settles to a steady state.

3.3 Root Locus Analysis

Open-loop transfer function:

$$G(s)H(s) = K \cdot \frac{1.151s + 0.1774}{s(s^2 + 0.739s + 0.921)}$$

```
    num = [1.151 0.1774];
den = [1 0.739 0.921 0];
sys = tf(num, den);
figure;
rlocus(sys);
title('Root Locus of Aircraft Pitch System');
grid on;
```

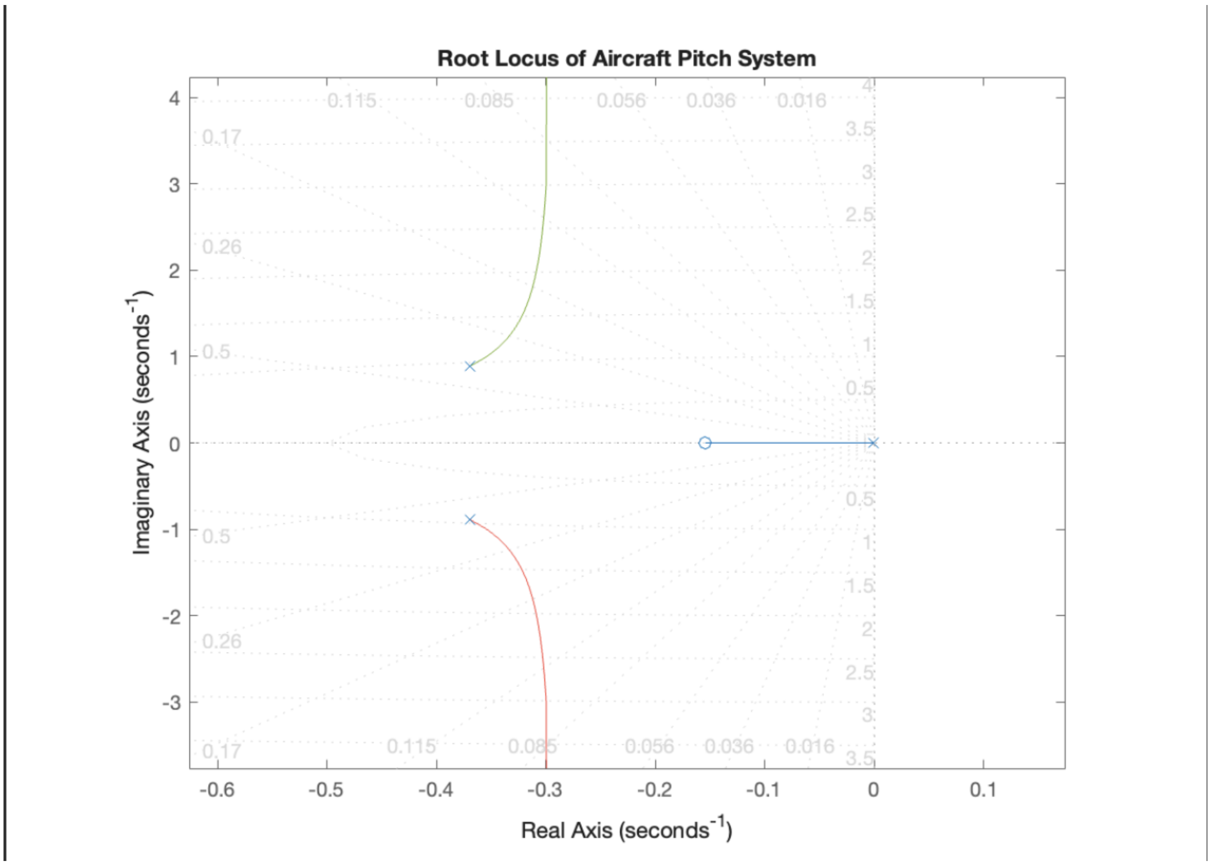


Figure 3.1: root locus

3.4 PID Tuning:

$$P(s) = \frac{\Theta(s)}{\Delta(s)} = \frac{1.151s + 0.1774}{s^3 + 0.739s^2 + 0.921s} \tag{3.1}$$

we have 2 inputs: first one is elevator deflection[δ] and other one is aircraft pich angle[θ] for our system we will use step reference of 0.2 radians

system design limitations:

- Overshoot less than 10%
- Rise time less than 2 seconds
- Settling time less than 10 seconds
- Steady-state error less than 2%

3.4.1 PID transfer function:

$$C_s = K_P + \frac{K_i}{s} + K_d s \tag{3.2}$$

We will implement combinations of proportional (K_p), integral (K_i), and derivative (K_d) control in the unity-feedback architecture shown below in order to achieve the desired system behavior.

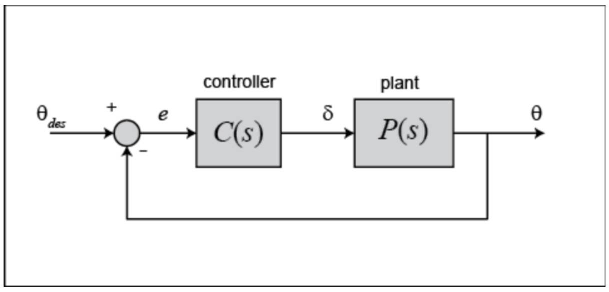


Figure 3.2: PID controller

3.4.2 Hand Tuning of PID Controller

Given the design specifications:

- Maximum overshoot: $OS\% \leq 10\%$ which corresponds to a damping ratio $\zeta = 0.59$,
- Settling time: $T_s \leq 10$ s,
- Desired rise time: $T_r = 2$ s.

Since the root locus does not intersect the line corresponding to $\zeta = 0.59$, we proceed to design a PD controller.

Step 1: PD Controller Design
The settling time is related to the damping ratio and natural frequency by:

$$T_s = \frac{4}{\zeta\omega_n} \implies \omega_n = \frac{4}{\zeta T_s}.$$

Using $\zeta = 0.59$ and $T_s = 10$ s, we find:

$$\omega_n = 0.677 \text{ rad/s}.$$

The damped natural frequency is:

$$\omega_d = \omega_n\sqrt{1-\zeta^2} = 0.5466 \text{ rad/s}.$$

Thus, the desired dominant poles are located at:

$$s = -0.4 \pm j0.5466.$$

Calculating the gain k using the magnitude condition of the root locus:

$$k = \frac{\text{product of pole distances}}{\text{product of zero distances}} = \frac{0.6773 \times 1.432 \times 0.34}{0.6898} = 0.4780.$$

The PD controller zero is placed at $s = -0.4$, so the PD controller transfer function is:

$$G_{PD}(s) = s + 0.4.$$

The overall compensated system transfer function becomes:

$$G_C(s) = \frac{k(s+0.4)(1.151s+0.1774)}{s^3 + 0.739s^2 + 0.921s}.$$

From the intersection of the damping ratio line with the new root locus, the new dominant poles are:

$$s = -0.66 \pm j0.885,$$

with gain $k = 0.598$.

Checking performance:

$$T_s = 6.06 \text{ s}, \quad T_r = 3.33 \text{ s (unsatisfactory)}, \quad k_p = \infty, \quad e_\infty = 0.$$

Step 2: PI Controller Addition

To improve steady-state error and transient response, we add a PI controller with zero at -0.2:

$$G_{PI}(s) = \frac{s+0.2}{s}.$$

The compensated system becomes:

$$G_C(s) = \frac{k(s+0.2)(s+0.4)(1.151s+0.1774)}{s^4 + 0.739s^3 + 0.921s^2}.$$

New dominant poles and gain:

$$s = -0.71 \pm j0.9291, \quad k = 0.669.$$

Performance metrics:

$$e_\infty = 0, \quad T_s = 5.63 \text{ s}, \quad T_p = 3.38 \text{ s}, \quad T_r = 3.098 \text{ s (still above target)}.$$

Step 3: PID Controller Design for Desired Rise Time

The desired rise time is $T_r = 2$ s, which relates to the real part of the pole σ by:

$$T_r = \frac{2.2}{\sigma} \implies \sigma = \frac{2.2}{T_r} = 1.1.$$

Using $\zeta = 0.59$, the natural frequency is:

$$\omega_n = \frac{\sigma}{\zeta} = \frac{1.1}{0.59} = 1.864 \text{ rad/s}.$$

The damped natural frequency is:

$$\omega_d = \omega_n \sqrt{1 - \zeta^2} = 1.0505 \text{ rad/s}.$$

Hence, the desired dominant poles are:

$$s = -1.1 \pm j1.505.$$

Calculating the angle contributions from existing poles and zeros:

$$\theta = (120.87^\circ + 114.9^\circ + 122.14^\circ) - (-107.67^\circ + 107^\circ + 139.7^\circ) = 218.88^\circ.$$

The zero angle θ_{zc} satisfies:

$$\theta_{zc} + 218.88^\circ = 180^\circ \implies \theta_{zc} = -38.88^\circ = 141.12^\circ.$$

Using the tangent relation for the zero location z_c:

$$\tan(141.12^\circ) = \frac{1.505}{z_c + 1.1} \implies z_c = 0.767.$$

Thus, the PID controller transfer function is:

$$G_{PID}(s) = \frac{k(s + 0.767)(s + 0.4)(1.151s + 0.1774)}{s^4 + 0.739s^3 + 0.921s^2}.$$

Performance check:

$$T_s = 3.63 \text{ s}, \quad T_p = 2.87 \text{ s}, \quad T_r = 2 \text{ s}, \quad e_\infty = 0, \quad k_3 = 1.588.$$

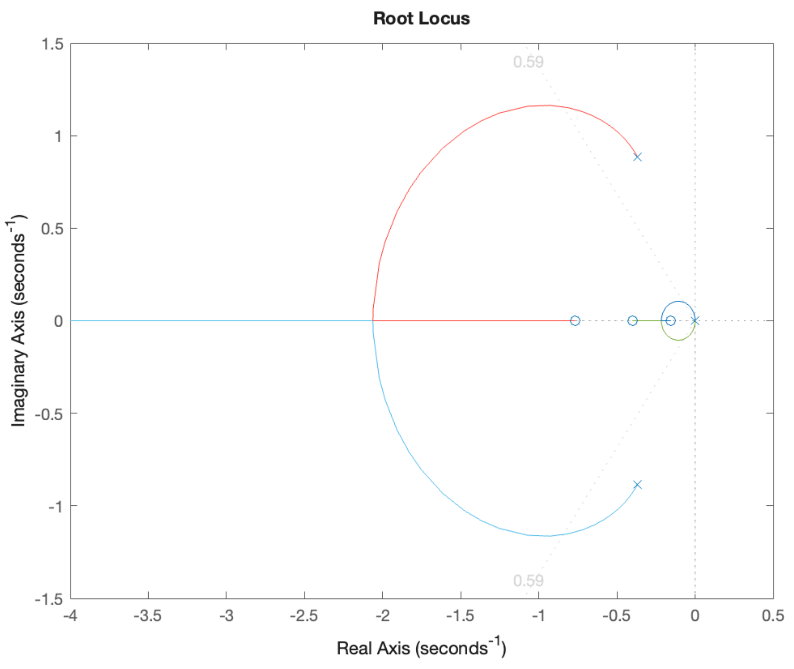


Figure 3.3: Root locus for tuned system

Step 4: PID Gains Determination
Expressing G_{PID} in the standard PID form:

$$G_{PID}(s) = k_1 + \frac{k_2}{s} + k_3 s = \frac{k_3\left(s^2 + \frac{k_1}{k_3}s + \frac{k_2}{k_3}\right)}{s}.$$

Matching coefficients yields:

$$G_{PID}(s) = \frac{1.588\,(s^2 + 1.167s + 0.3068)}{s}.$$

Therefore, the PID gains are:

$$\boxed{\begin{cases} k_1 = 1.85 & \text{(Proportional gain)} \\ k_2 = 0.487 & \text{(Integral gain)} \\ k_3 = 1.588 & \text{(Derivative gain)} \end{cases}}$$

3.4.3 Simulink:

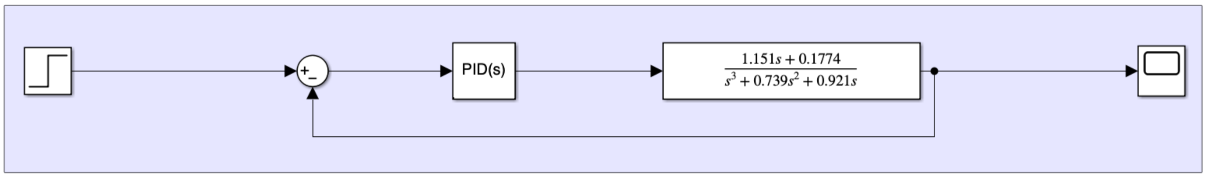
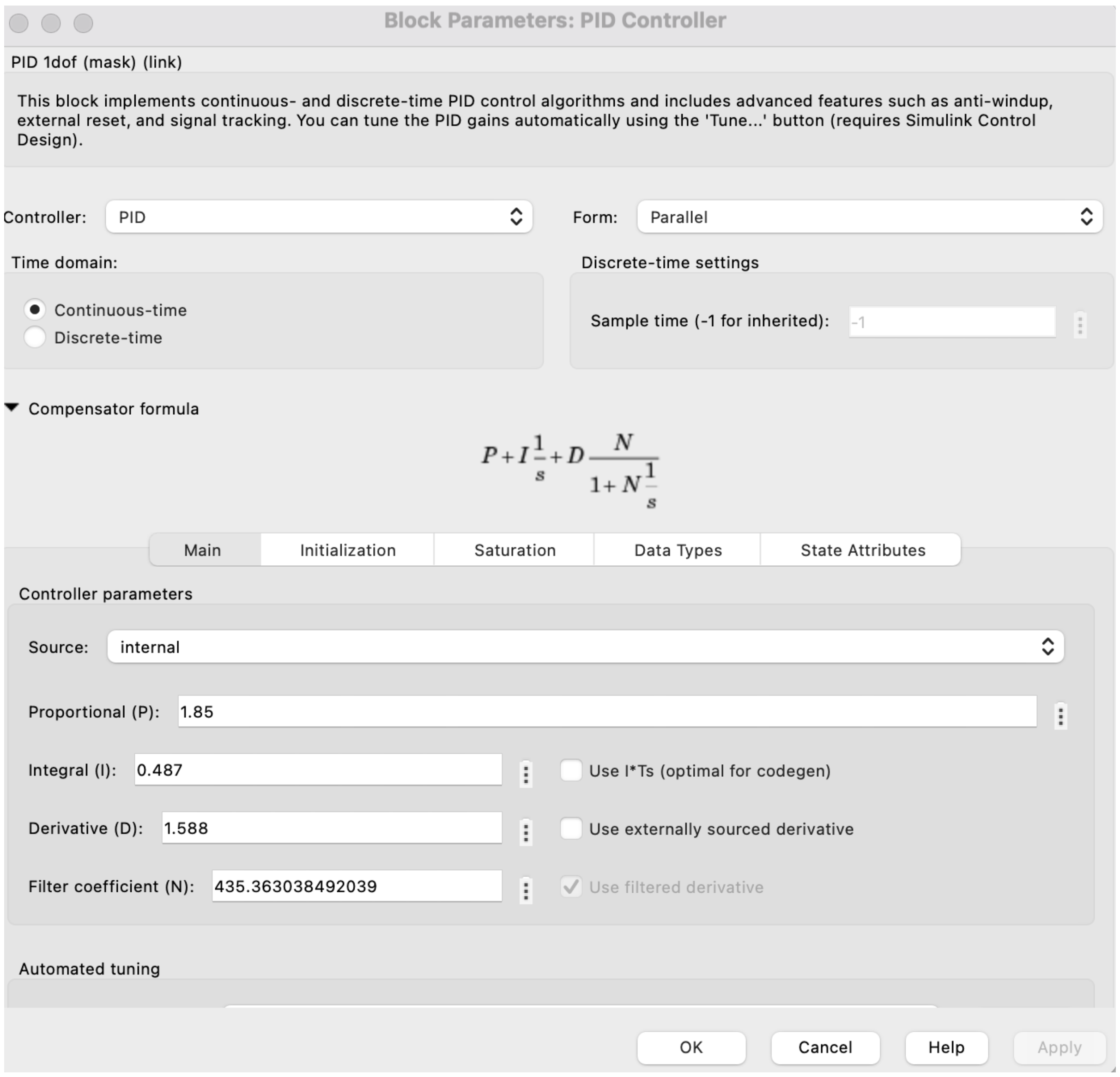


Figure 3.4: system model


Block Parameters: PID Controller

PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PID

Form: Parallel

Time domain:

● Continuous-time

○ Discrete-time

Discrete-time settings

Sample time (-1 for inherited): -1

Compensator formula

$$P+I\frac{1}{s}+D\frac{N}{1+N\frac{1}{s}}$$

Controller parameters

Source: internal

Proportional (P): 1.85

Integral (I): 0.487 ☐ Use I*Ts (optimal for codegen)

Derivative (D): 1.588 ☐ Use externally sourced derivative

Filter coefficient (N): 435.363038492039 ✓ Use filtered derivative

Automated tuning

Figure 3.5: PID block hand calculated factors

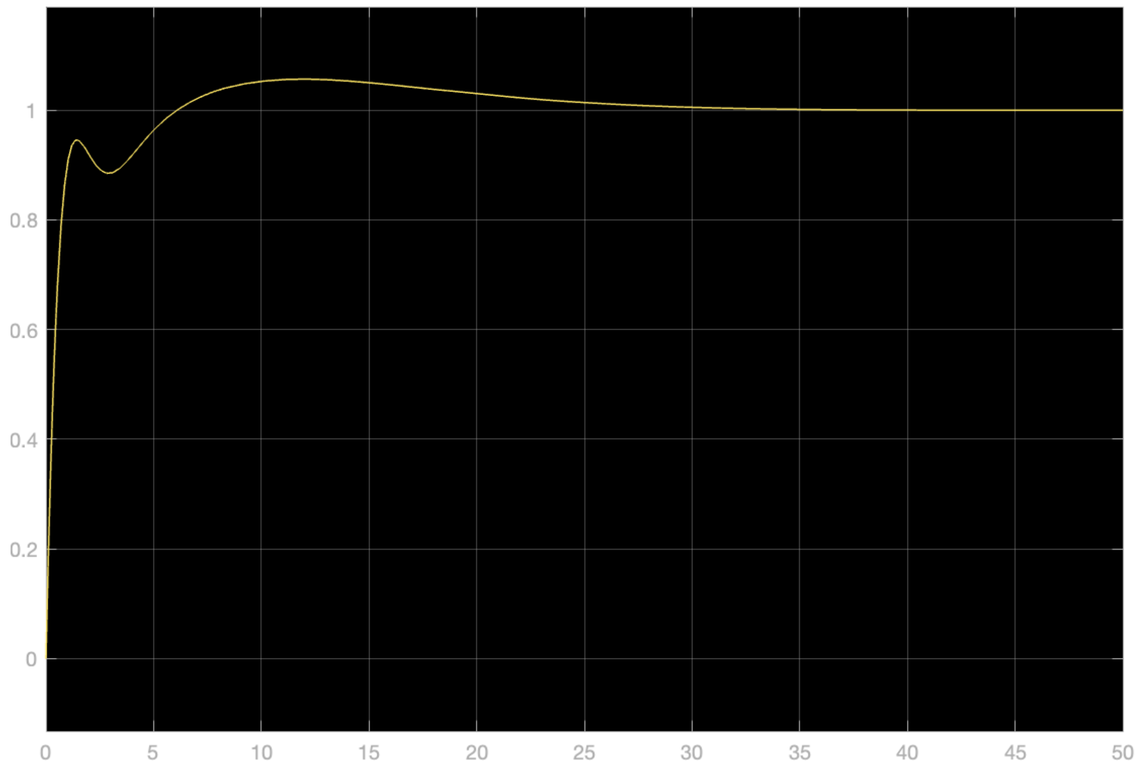
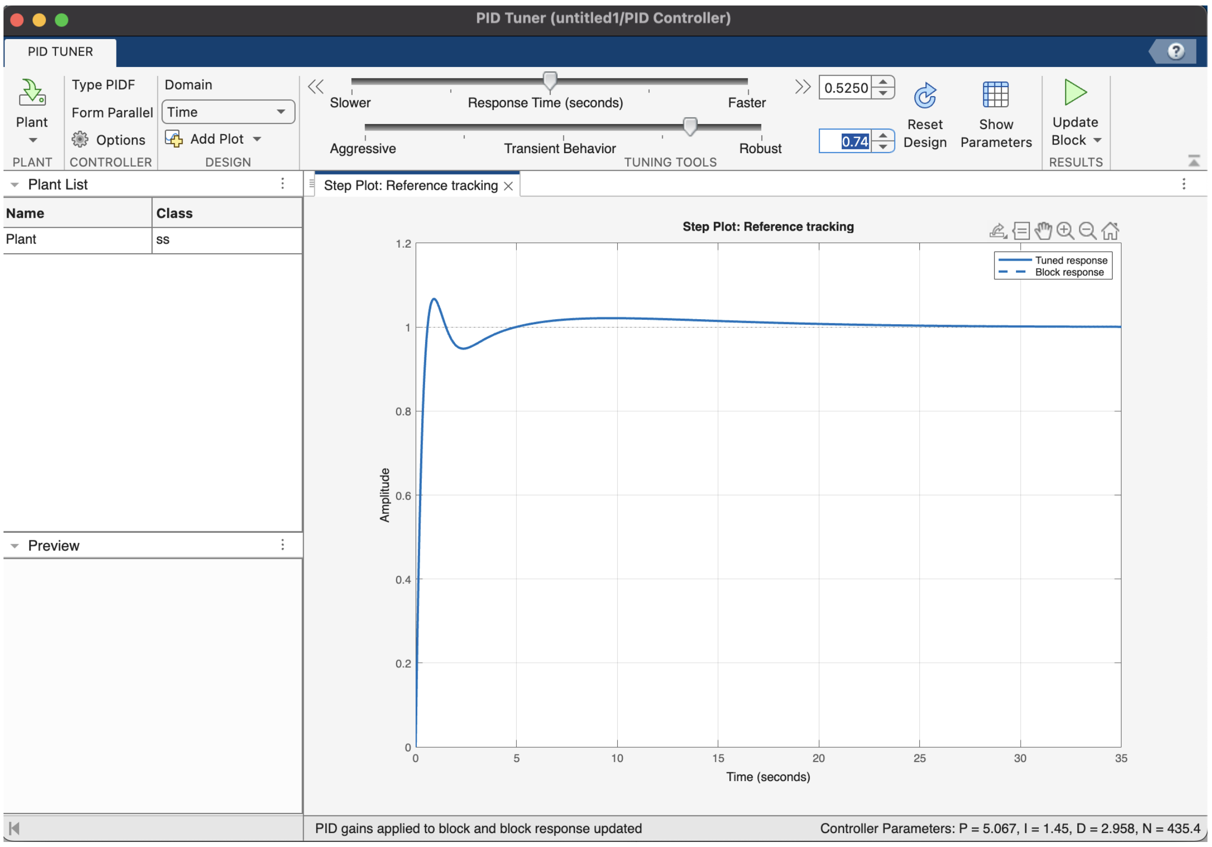


Figure 3.6: calculated Factors PID response

3.4.4 Simulink tuning:



PID Tuner (untitled1/PID Controller)

Type PIDF

Form Parallel

Domain: Time

Response Time (seconds): 0.5250

Transient Behavior: 0.74

Plant List

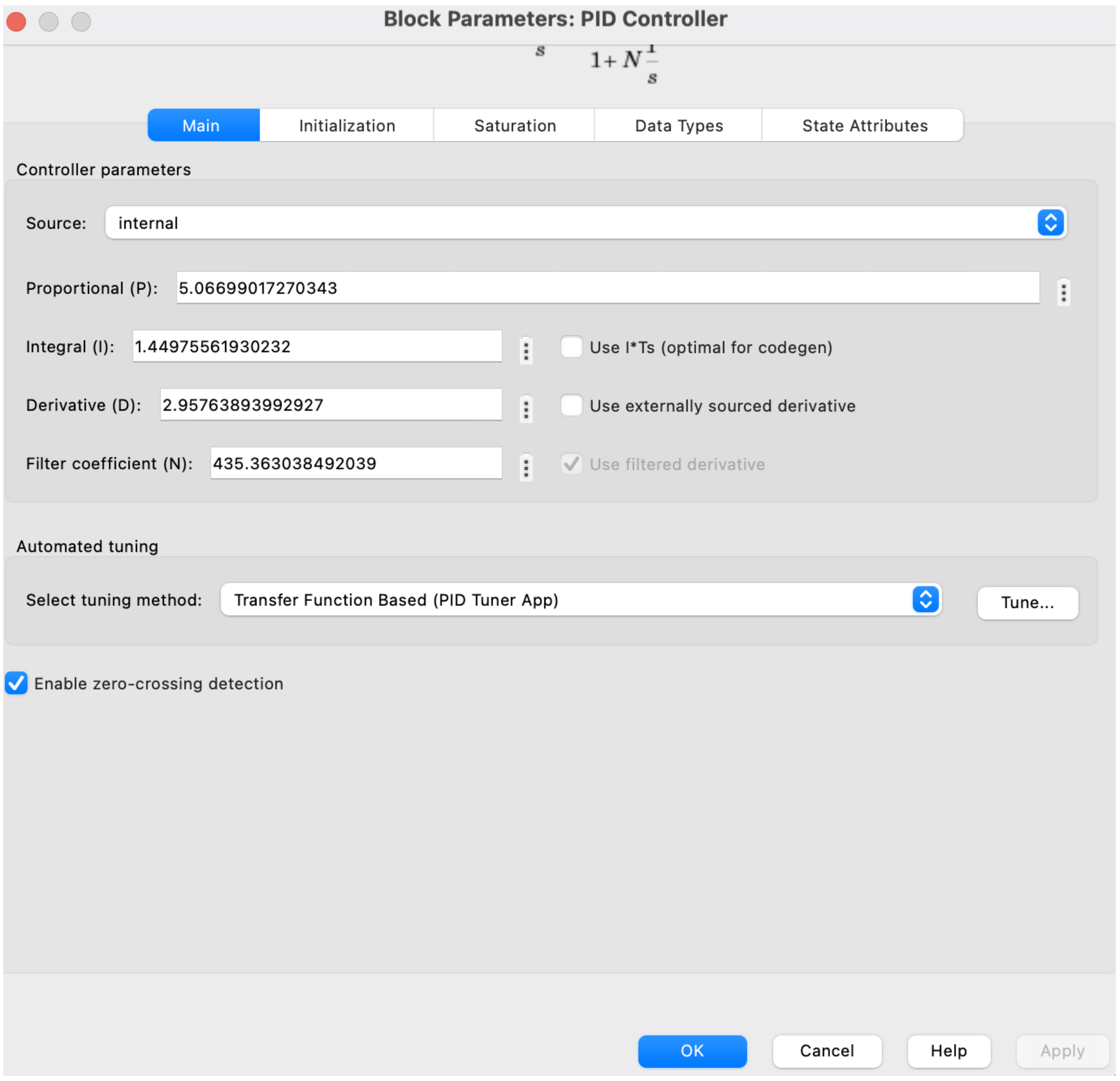
Name	Class
Plant	ss

PID gains applied to block and block response updated

Controller Parameters: P = 5.067, I = 1.45, D = 2.958, N = 435.4

Figure 3.7: tuning using PID tuner

set response time 0.525 and transient behavior =0.72


Block Parameters: PID Controller

$1+N\frac{1}{s}$

Main | Initialization | Saturation | Data Types | State Attributes

Controller parameters

Source: internal

Proportional (P): 5.06699017270343

Integral (I): 1.44975561930232 — Use I*Ts (optimal for codegen) [ ]

Derivative (D): 2.95763893992927 — Use externally sourced derivative [ ]

Filter coefficient (N): 435.363038492039 — Use filtered derivative [✓]

Automated tuning

Select tuning method: Transfer Function Based (PID Tuner App)

[✓] Enable zero-crossing detection

Figure 3.8: Tuned Factors

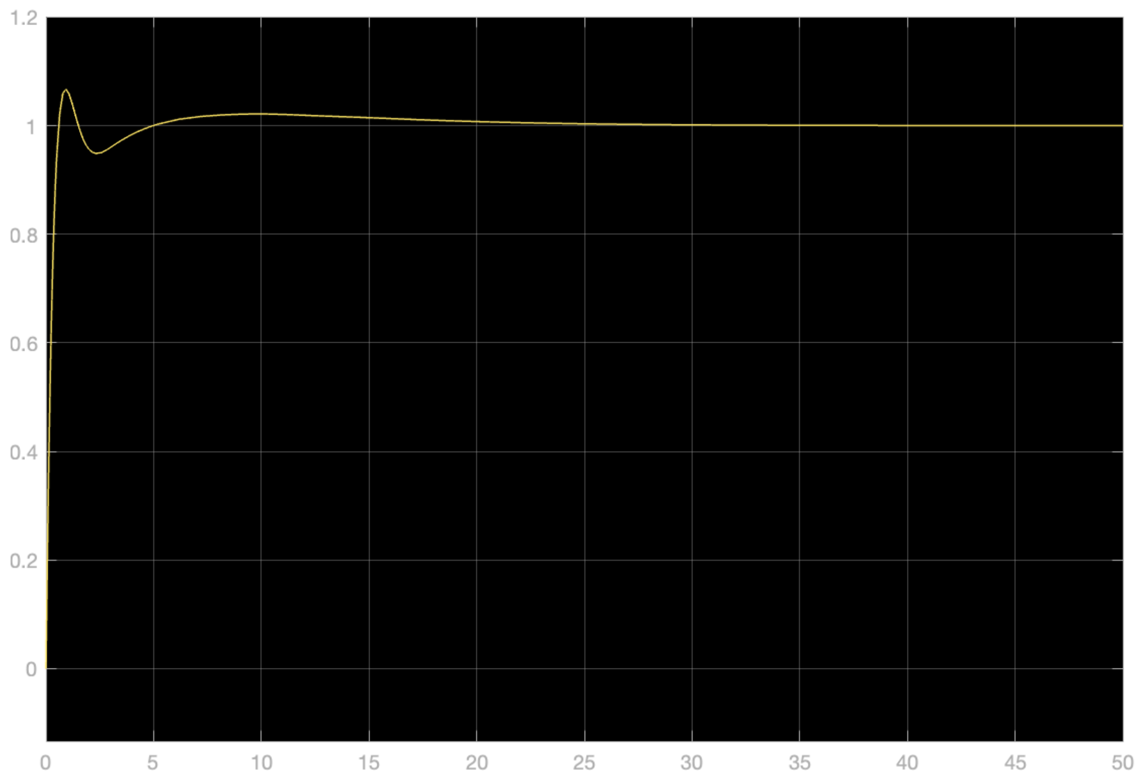


Figure 3.9: tuned response:

3.4.5 second method of Tuining

1. first we define our transfer function in matlab

```
        s = tf('s');
P_pitch = (1.151*s+0.1774)/(s^3+0.739*s^2+0.921*s);

```

2. then we use matlab Control System Designer to edit directly into PID parameters and notice response of each change

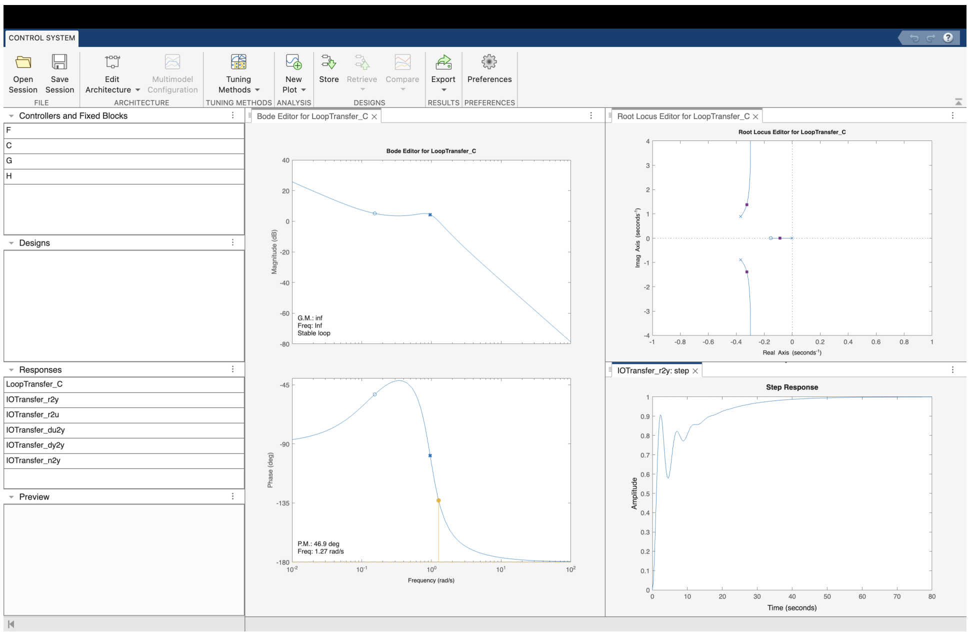
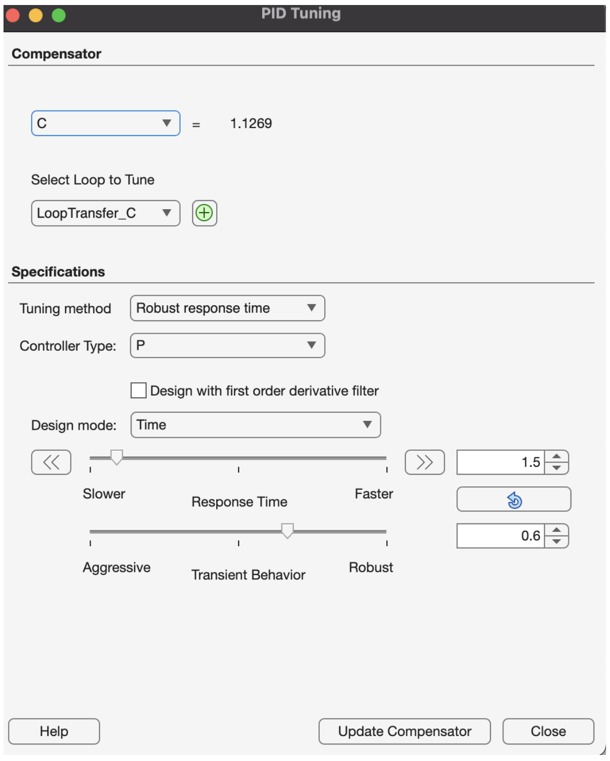


Figure 3.10: control system designer screen

3. then we set step to be 0.2 after that we start to tune PID controller at the beginning we try (P)controller by put response time to be 1.5 after this tuning we get $K_p = 1.1269$


PID Tuning

Compensator

C = 1.1269

Select Loop to Tune: LoopTransfer_C

Specifications

Tuning method: Robust response time

Controller Type: P

Design with first order derivative filter

Design mode: Time

Response Time: 1.5

Transient Behavior: 0.6

Figure 3.11: P Tuning

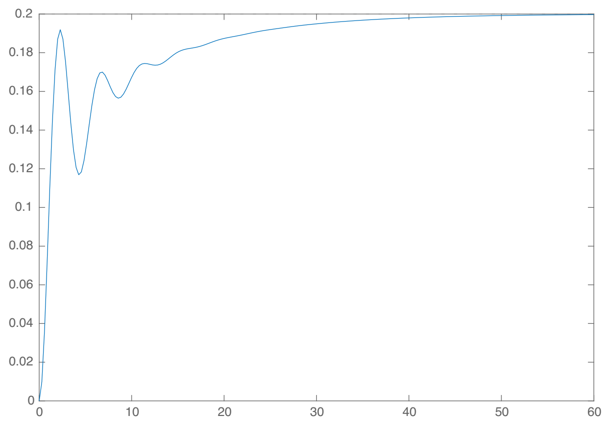
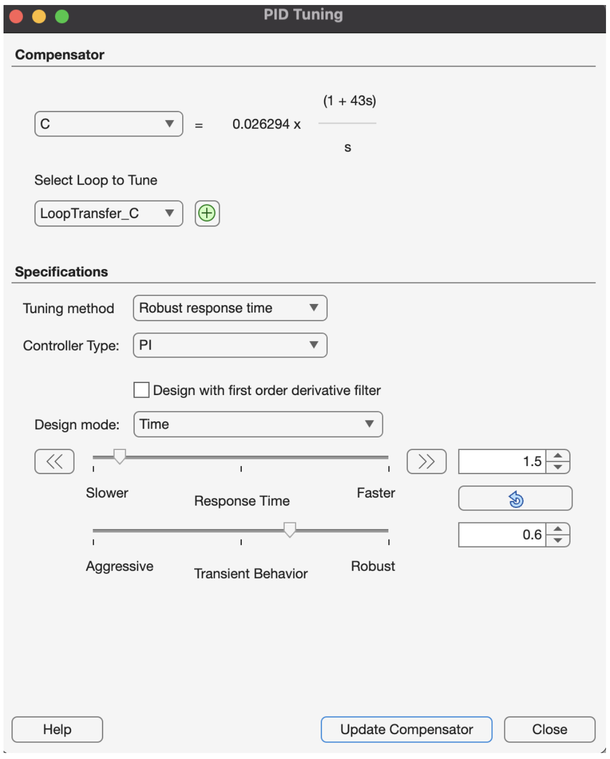


Figure 3.12: P controller response

as we see This controller meets the rise time requirement, but the settle time is much too large. The proportional controller does not provide us a sufficient degree of freedom in our tuning, we need to add integral and/or derivative terms to our controller in order to meet the given requirements.

4. then we try tuning PI controller without any change in previos change



PID Tuning

Compensator

C = 0.026294 x (1 + 43s) / s

Select Loop to Tune: LoopTransfer_C

Specifications

Tuning method: Robust response time

Controller Type: PI

Design with first order derivative filter

Design mode: Time

Response Time: 1.5

Transient Behavior: 0.6

Figure 3.13: PI tuning

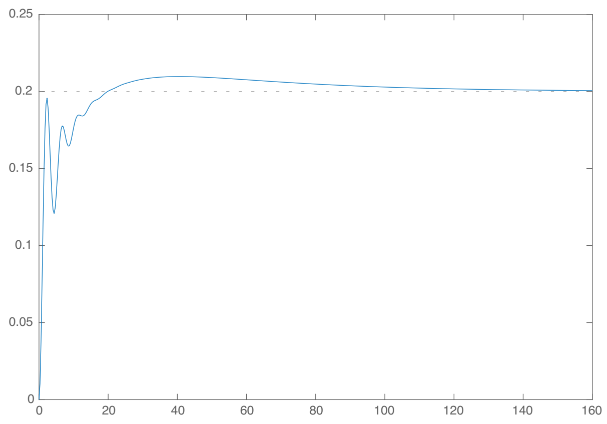


Figure 3.14: PI response

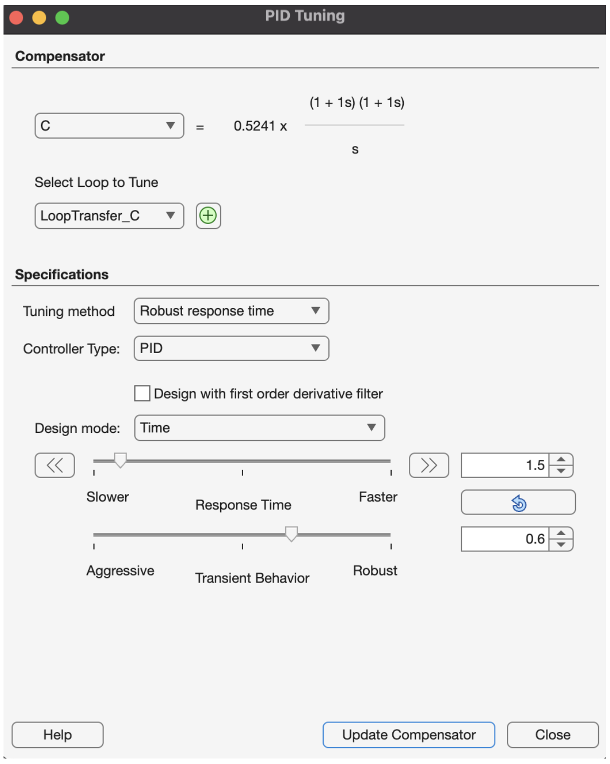
as we see

$$C_{(S)} = 0.026294 \times \frac{1 + 43s}{s} \approx 1.13 + \frac{0.0263}{s} \tag{3.3}$$

this mean $k_p = 1.13$, $K_i = 0.0263$

we notice that the addition of integral control helped reduce the average error in the signal more quickly, but it didn't help reduce the oscillation. Let us try also adding a derivative term to our controller.

5. the last step is to tuning PID


PID Tuning

Compensator

C = 0.5241 x (1 + 1s) (1 + 1s) / s

Select Loop to Tune: LoopTransfer_C

Specifications

Tuning method: Robust response time

Controller Type: PID

Design with first order derivative filter: ☐

Design mode: Time

Response Time: 1.5

Transient Behavior: 0.6

Figure 3.15: PID Tuning

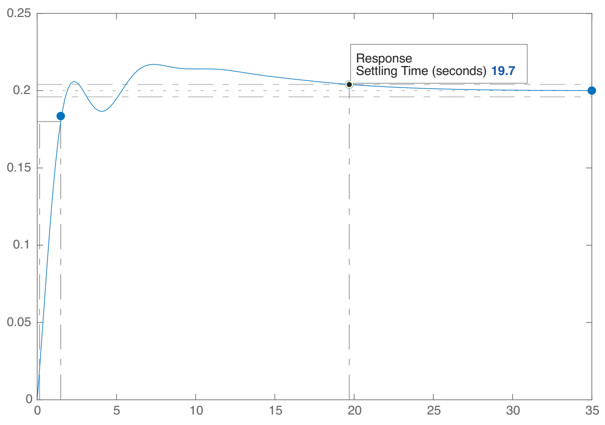


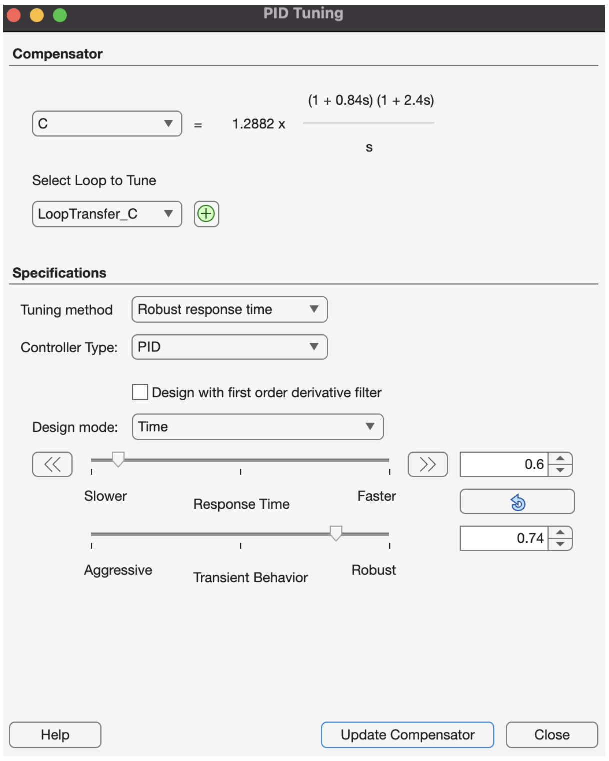
Figure 3.16: Caption

As we see

$$C_{(S)} = 0.5241 \times \frac{(1+s)(1+s)}{s} \approx 1.0482 + \frac{0.5241}{s} + 0.5241s \tag{3.4}$$

this mean $K_p = 1.0482$,$K_i = 0.5241$,$K_d = 0.5421$ This response meets all of the requirements except for the settle time which at 19.7 seconds is larger than the given requirement of 10 seconds so we will try to retune PID again till settle time less than 10

6. first modification set response time to be 0.6 and transient behavior to be 0.74


PID Tuning

Compensator

C = 1.2882 x $\frac{(1 + 0.84s)(1 + 2.4s)}{s}$

Select Loop to Tune: LoopTransfer_C

Specifications

Tuning method: Robust response time

Controller Type: PID

Design with first order derivative filter

Design mode: Time

Response Time: 0.6 (Slower … Faster)

Transient Behavior: 0.74 (Aggressive … Robust)

Figure 3.17: PID first modification

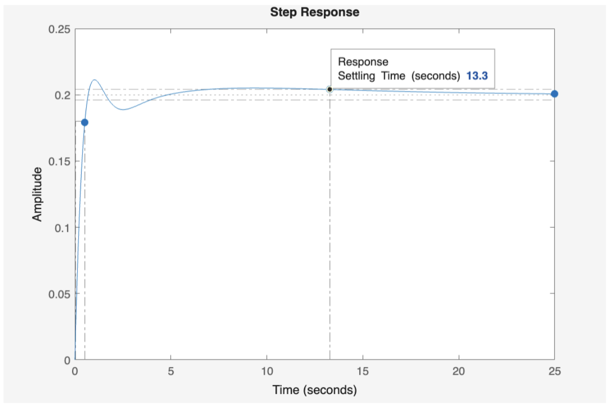


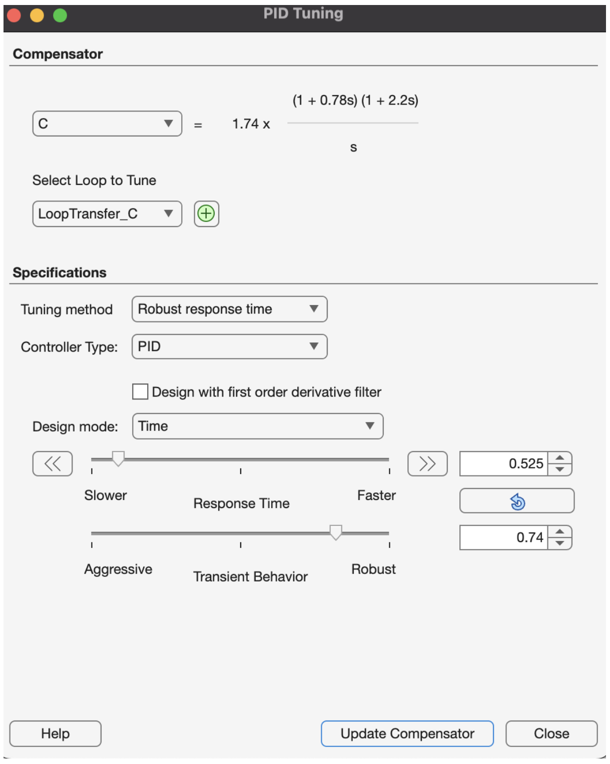
Figure 3.18: first modification response

As we see

$$C_{(S)} = 1.2882 \times \frac{(1+0.84s)(1+2.4s)}{s} \approx 4.17 + \frac{1.2882}{s} + 0.26s \qquad (3.5)$$

this mean $K_p = 4.17$,$K_i = 1.2882$,$K_d = 0.26$ Here we can see that moving both sliders to the right made the response faster and reduced the oscillation. However, the settling time =13.3 is still greater than the required 10 seconds. We again try increasing the required speed of response

7. Second modification set response time to be 0.525 and transient behavior to be 0.74


PID Tuning

Compensator

C = 1.74 x (1 + 0.78s) (1 + 2.2s) / s

Select Loop to Tune: LoopTransfer_C

Specifications

Tuning method: Robust response time

Controller Type: PID

☐ Design with first order derivative filter

Design mode: Time

Response Time (Slower – Faster): 0.525

Transient Behavior (Aggressive – Robust): 0.74

Help | Update Compensator | Close

Figure 3.19: PID second modification

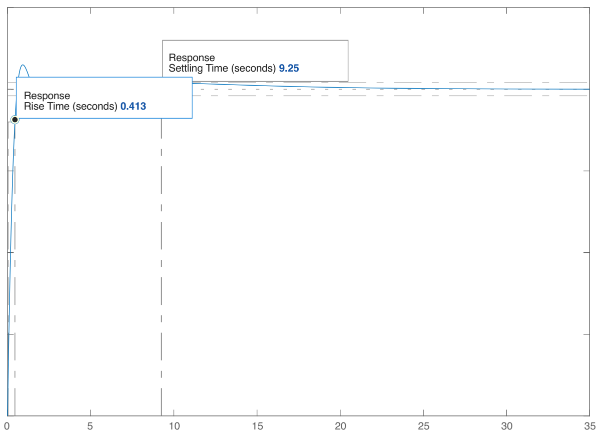


Figure 3.20: Caption

As we see

$$C_{(S)} = 1.74 \times \frac{(1 + 0.78s)(1 + 2.2s)}{s} \approx 5.1852 + \frac{1.74}{s} + 2.98s \qquad (3.6)$$

this mean $K_p = 5.185$,$K_i = 1.74$,$K_d = 2.98$

Now Our system meets all specified requirements as demonstrated below:

Transient Response

- **Overshoot:**

$$\text{Actual} = 7.5\% \quad (< 10\% \text{ requirement}) \quad \checkmark$$

- **Rise Time (t_r):**

$$\text{Actual} = 0.413\,\text{s} \quad (< 2\,\text{s requirement}) \quad \checkmark$$

- **Settling Time (t_s):**

$$\text{Actual} = 9.25\,\text{s} \quad (< 10\,\text{s requirement}) \quad \checkmark$$

Steady-State Performance

$$\text{Steady-State Error} = 0\% \quad (< 2\% \text{ requirement}) \quad \checkmark$$

8. **Root locus after Using PID**

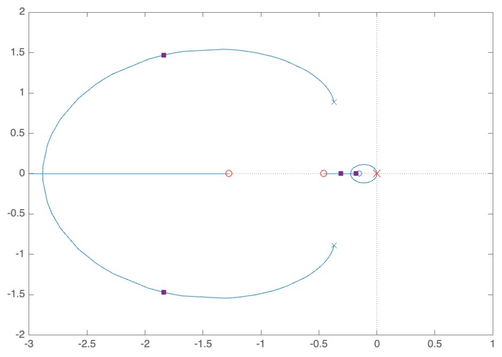


Figure 3.21: Root Loucs with PID

9. **PID controller Gains**
The PID controllor we will use has the following paramters

- $K_p = 5.185$
- $K_i = 1.74$
- $K_d = 2.98$

3.5 Simulink:

3.5.1 model:

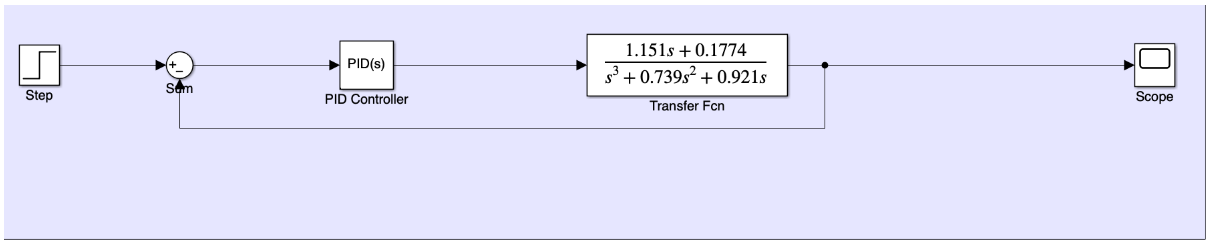


Figure 3.22: Simulink model

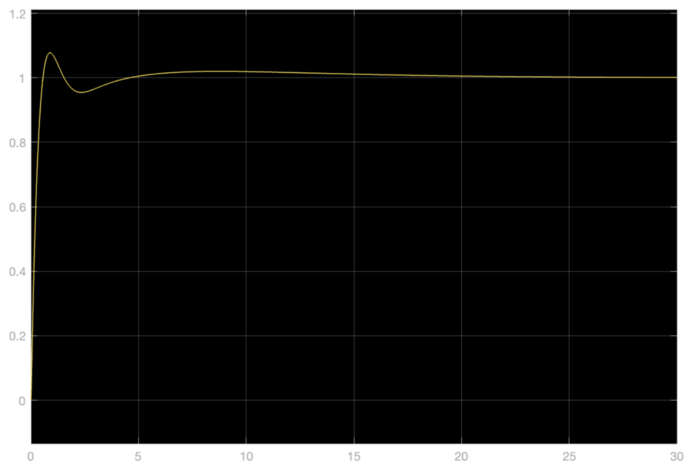


Figure 3.23: output

As observed, our system exhibits zero steady-state error, and overshoot is also minimal. Furthermore, the rise time is negligible. Consequently, our designed PID controller is well-suited for the system.

3.6 Conclusion

This research has presented a comprehensive analysis and design approach for an aircraft pitch control system using classical control methodologies. The investigation commenced with the development of a mathematical model represented by the transfer function $P(s) = \frac{1.151s+0.1774}{s^3+0.739s^2+0.921s}$, which accurately captures the essential dynamics of aircraft pitch motion.

Through rigorous analytical techniques including pole-zero mapping, Routh-Hurwitz stability criterion, and root locus analysis, we established the baseline characteristics of the uncompensated system. The presence of a pole at the origin in the open-loop transfer function indicated marginal stability, while the complex conjugate poles at $s = -0.3695 \pm j0.8861$ revealed the system's natural oscillatory behavior.

The implementation of a proportional-integral-derivative (PID) controller was methodically pursued through an iterative tuning process guided by established performance criteria. The finalized controller configuration, characterized by gains $K_p = 5.185$, $K_i = 1.74$, and $K_d = 2.98$, successfully achieved all specified performance metrics:

- Peak overshoot limited to 7.5%, well within the 10% constraint
- Rise time of 0.413 seconds, significantly below the 2-second threshold
- Settling time of 9.25 seconds, satisfying the 10-second requirement
- Steady-state error elimination, surpassing the 2% maximum allowance

Verification through Simulink simulations corroborated the analytical predictions, demonstrating that the designed control system effectively stabilizes the aircraft pitch dynamics while maintaining precise reference tracking. The simulation results confirm the robustness of the control strategy in maintaining the desired pitch angle under nominal operating conditions.

This research demonstrates the efficacy of classical control theory in addressing complex aerospace control problems. The systematic methodology employed—from mathematical modeling to controller design and validation—provides a structured framework applicable to similar flight control challenges. The successful implementation of the PID controller transforms the inherently unstable open-loop aircraft pitch system into a stable, responsive, and accurate closed-loop system that meets all prescribed performance specifications.

Future research directions may include the incorporation of robust control techniques to address parametric uncertainties, the exploration of adaptive control strategies to accommodate varying flight conditions, and the integration of this pitch control subsystem into a comprehensive flight control architecture.

References

1. Control Tutorials for MATLAB and Simulink (CTMS). ”Aircraft Pitch: System Modeling.” University of Michigan. https://ctms.engin.umich.edu/CTMS/index.php?example=AircraftPitch&section=SystemModeling (accessed May 7, 2025).

2. NASA Glenn Research Center. ”Aircraft Pitch Motion.” NASA. https://www.grc.nasa.gov/WWW/K-12/VirtualAero/BottleRocket/airplane/pitch.html (accessed May 7, 2025).