Flight Systems E3 Assignment Report (Digital Servo and MATLAB Assignment)

Philip Beswick @00662943

The University of Salford

Flight Systems E3

Professor T X Mei, Dr B A Sangolola, Dr Y Govdeli, Dr O K Ariff

18/03/25

Abstract

Some abstract yada yada yada

Contents

1 Digital Control Design Laboratory

Introduction

1.1 Objectives, Theory, Apparatus

1.2 Models and Results

1.3 Discussion

1.4 Conclusions

2 Altitude Hold Autopilot Design

Both safety and efficiency have been significantly improved thanks to the introduction of autopilot systems, which have become an essential aspect of modern aviation. Initially, autopilots were introduced to reduce the flight crew's workload during long-haul flights but over time they have evolved into highly sophisticated systems, which are capable of completing various complex tasks autonomously. One of the different functionalities of autopilots is altitude hold. This feature is of paramount importance for safety and efficiency, as it ensures a steady and controlled flight path is followed, this means that the desired cruising altitude is maintained. Any deviation in the desired cruising altitude could result in serious safety concerns, the aircraft could come into conflict, and cause fuel inefficiencies, thus reducing profit margins for airlines.

The primary function of altitude hold in autopilots is to maintain a specific altitude, autonomously; this allows pilots to focus on other critical aspects of flight, including communicating with air traffic control. This is vitally important for long flights, as small deviations in altitude accumulate over time, resulting in a very significant altitude change (Prasad et al., 2017). This is especially important when flying through turbulence, which can drastically change the aircraft's altitude in a very short period of time, this turbulence is also very frequently encountered between 30kft and 40kft (the altitude range of most modern airliners) (Prasad et al., 2017). Altitude hold systems will automatically account for these altitude changes, returning the aircraft to equilibrium; the best autopilot systems do this quickly and efficiently (with as few oscillations as possible).

2.1   Objectives and Theory

2.1.1   Objectives

The key objectives of any altitude hold system are:

1. To maintain a fixed altitude, which ensures compliance with Air Traffic Control regulations.

2. Minimise pilot workload, by providing automatic altitude maintenance during cruise flight.

3. Enhance flight stability, by reducing oscillations due to atmospheric disturbances and therefore, improving passenger comfort.

4. Creating smooth transitions when changing altitude, which improves fuel efficiency.

2.1.2   Theory

The altitude hold system is modelled using the two degrees of freedom linear equations of motion, which describe aircraft longitudinal dynamics. They are given by

$$\dot{w} = \overline{Z}_w w + V_R q + \overline{Z}_\eta \eta \,, \tag{2.1}$$

$$\dot{q} - \overline{M}_{\dot{w}} \dot{w} = \overline{M}_w w + \overline{M}_q q + \overline{M}_\eta \eta \,, \tag{2.2}$$

$$q = \dot{\theta} \,, \tag{2.3}$$

$$\dot{h} = V_R \theta - w \,, \tag{2.4}$$

These equations describe how the aircraft's vertical velocity (w), pitch rate (q) and altitude (h) change due to control surface deflections (η). From these equations, transfer functions in the laplace domain can be derived

$$G_\eta^q(s) = \frac{(\overline{M}_\eta + \overline{M}_{\dot{w}}\overline{Z}_\eta)s + (\overline{M}_w\overline{Z}_\eta - \overline{M}_\eta\overline{Z}_w)}{s^2 - (V_R\overline{M}_{\dot{w}} + \overline{Z}_w + \overline{M}_q)s + (\overline{M}_q\overline{Z}_w - V_R\overline{M}_w)} \,, \tag{2.5}$$

Equation 2.5 describes how elevator deflection affections pitch rate.

$$G_\eta^\theta(s) = \frac{1}{s} \cdot G_\eta^q \,, \tag{2.6}$$

Equation 2.6 describes how elevator deflection affects pitch angle.

$$G_\theta^h(s) = \frac{1}{s} \cdot \frac{-\overline{Z}_\eta s^2 + (\overline{M}_q + V_R\overline{M}_{\dot{w}})\overline{Z}_\eta s + V_R(\overline{M}_w\overline{Z}_\eta - \overline{M}_\eta\overline{Z}_w)}{(\overline{M}_\eta + \overline{M}_{\dot{w}}\overline{Z}_\eta)s + (\overline{M}_w\overline{Z}_\eta - \overline{M}_\eta\overline{Z}_w)} \,, \tag{2.7}$$

Equation 2.7 describes how pitch angle affects altitude.

The autopilot system has been designed using a sequential loop closure approach, with two distinct stages. The first is the design of a pitch attitude hold system, which is made up of the two inner loops seen in Figure 2.1. The second is the closure of the feedback loop around the pitch attitude hold system; again as seen in Figure 2.1.

Figure 2.1

Block diagram of an Altitude Hold System.
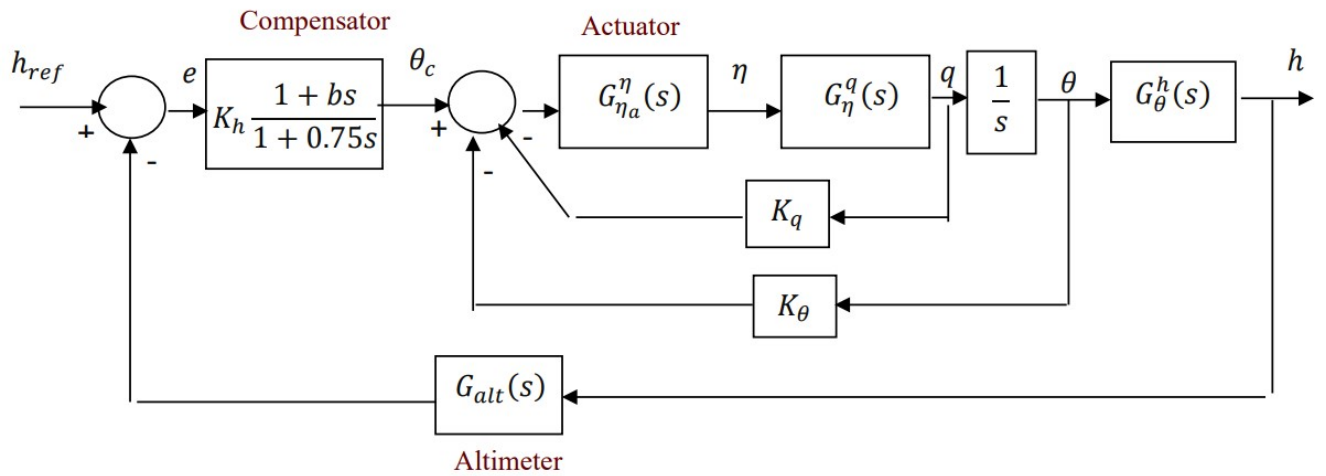
The first stage (inner loops) is responsible for regulating pitch attitude. It consists of both a pitch rate feedback loop, which damps excessive pitch rate variations, and a pitch angle feedback loop, which ensures the aircraft maintains required altitude by adjusting the elevator control. The closed loop transfer function for the pitch attitude design is derived by root locus design, to achieve a damping ratio of 0.5 and natural frequency of 3 rad/s. This is achieved as follows:

$$s_1 = -\sigma + j\omega_d, \tag{2.8}$$

$$\sigma = \zeta\omega_n, \tag{2.9}$$

$$\omega_d = \omega_n\sqrt{1 - \zeta^2}, \tag{2.10}$$

When ζ takes the value of 0.5 and ω_n takes the value of 3, $s_1 = -1.5 + 2.5981j$.

Then, a compensator zero is placed at $s = -a$ so that the desired pole lies on the root locus. This is achieved by first reducing the block diagram shown in Figure 2, by replacing $G_{\eta_a}^{\eta}$ and G_{η}^{q} with transfer function G_p. Finally transfer function G_1 can be found by dividing by s:

$$G_1(s) = \frac{G_p}{s_1}, \tag{2.11}$$

Finally, the value of a can be determined by ensuring that the sum of a and the angle of s_1 multipled by the angle of G_1 is equal to π. This can be expressed mathematically as:

$$\angle((s_1 + a)G_1(s)) = \pi, \tag{2.12}$$

With the compensator pole in place, values of the gains K_q and K_θ can be obtained, so that the closed-loop transfer function of the pitch attitude control system can be found.

$$K_q = \frac{1}{|G_1(s_1)|}, \tag{2.13}$$

$$K_\theta = aK_q, \tag{2.14}$$

So the closed-loop transfer function of the pitch attitude system is:

$$G_{\theta_c}^{\theta} = \frac{G_1(s)}{1 + K_q(s + a)G_1(s)}, \tag{2.15}$$

The denominator of the transfer function in Equation 2.15 is the characteristic equation of the pitch attitude system. This means that the zero of the system is located at $s = -a$, with point s_1 lying on the root locus branch for the open loop transfer function:

$$G_2(s_1) = (s+a)G_1(s), \qquad (2.16)$$

This transfer function should also satisfy the root locus criterion, as the angular value should be exactly equal to π.

The second stage (outer loop) regulates the aircraft's altitude by adjusting the pitch attitude. The altimeter provides altitude feedback, and a compensator adjusts the pitch command to correct altitude deviations. The altitude loop is only closed after designing the pitch control loop, so that appropriate pitch commands are generated, and then executed by the inner loop. Therefore, since the transfer function for the inner loop has been set (Equation 2.15) the design for the outer loop can be determined by block diagram reduction, as seen in Figure 2.2.

Figure 2.2

Simplified block diagram for the altitude hold loop (outer loop).
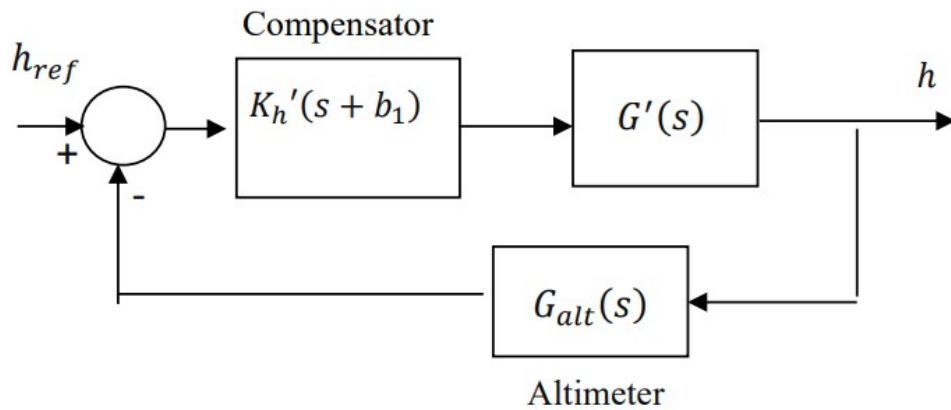
Where:

$$G' = \left(\frac{1}{1+0.75s}\right)G_{\theta_c}^{\theta}(s)G_{\theta}^{h}(s), \qquad (2.17)$$

And:

$$G^*(s) = G'(s)G_{alt}(s), \qquad (2.18)$$

Terms K_h' and b_1 are to be determined, however transfer function $G_{alt}(s)$ has known properties as the performance of the aircraft's elevators have been measured.

$$G_{alt}(s) = \frac{10}{s+10}, \tag{2.19}$$

For the outer loop, an identical method, to what was used in the inner loop, is used to design a root locus. The only difference is that we solve with complex number s_2 by using a ζ of 0.5 and a ω_n of 0.5 rad/s. Therefore, by recalling Equations 2.8, 2.9 and 2.10, the complex number s_2 is found. Therefore, a compensator zero can be placed when $s = b_1$, which can be found by slightly modifying Equation 2.12:

$$\angle((s_2 + b_1)G^*(s_2)) = \pi, \tag{2.20}$$

Once a suitable b_1 has been determined, the transfer function $G_3(s_2)$ can be determined, by taking G_{alt} and multiplying by $(s+b_1)$, which now includes our compensator zero in the transfer function. This means that, the value of the gains K_h' and K_h can be found by direct calculation:

$$K_h' = \frac{1}{|G_3(s2)|}, \tag{2.21}$$

$$K_h = \frac{K_h'}{b_1}, \tag{2.22}$$

Additionally, just like for $G_2(s_1)$ the angular value of the transfer function $G_3(s2)$ should be exactly equal to π.

Now, all unknown terms have been evaluated, including the gains used in the Compensator block in Figure 2.2. This means that the outer loop can be closed around the

inner loop, completing the theoretical altitude hold design. The simplified closed loop transfer function is:

$$G_{h_c}^h(s) = \frac{K_h'(s+b_1)G'}{1+G_3(s_2)K_h'},$$ (2.23)

It's important to note the key relationship between the two loops: the inner loop must be fast and stable to respond effectively to pitch commands from the outer loop, whereas, the outer loop uses the inner loop to influence the altitude, but operates slower to avoid excessive oscillaitons.

2.2 Modelling and Results

2.2.1 Modelling

The mathematical modelling of the altitude hold system was completed inside of a MATLAB environment, with two distinct scripts (.m files). One of these files is responsible for handling all variables that will be used throughout the course of the design process. The second file contains the mathematical model of the altitude hold system as outlined previously.

Storing the variables in a different script provides a great deal of flexibility, as changing one line of code will enable the user to simulate the autopilot hold system in different conditions, without having to manually change each aerodynamic and control derivative. Variable files are stored in the following format:

```
1  %%% Stability and Control Derivatives %%%
2  function [Xu, Zu, Mu, Xw, Zw, Mw, Xw_dot, Zw_dot, Mw_dot, Xq, Zq, Mq,
       Xeta, Zeta, Meta, VR, g] = final_assignment_variables ()
3     Xu = -1.58E-02;     Zu = -1.442E-01;    Mu = -1.3442E-04;
4     Xw_dot = 0;         Zw_dot = -2.2E-03;  Mw_dot = -2.0483E-04;
5     Xw = 3.79E-03;      Zw = -8.73E-01;     Mw = -6.0107E-03;
6     Xq = 0;             Zq = 0;             Mq = -9.88E-01;
7     Xeta = 0;           Zeta = -1.2408E+01; Meta = -1.153E+01;
8
9     VR = 236; g = 9.8100;
10  end
11  %%% END OF VARIABLES %%%
```

The main substance of the Autopilot script has been broken down into 17 steps, each aiming to accomplish a specific design goal. The MATLAB code is as follows:

```matlab
1  %%% Workspace Cleanup %%%
2  clc;
3  clear;
4  close all;
5
6  s = tf('s');
7  %%% Stability and Control Derivatives and Flight Condition %%%
8  [Xu, Zu, Mu, Xw, Zw, Mw, Xw_dot, Zw_dot, Mw_dot, Xq, Zq, Mq, Xeta, Zeta
      , Meta, VR, g] = final_assignment_variables(); % Change the name of
      this function to to run this system with different variable values.
9
10 %%% Actuator and Altimeter Dynamics %%%
11 G_etaA_eta = -4/(s+4);
12 G_etaA_eta_Numerator = cell2mat(G_etaA_eta.Numerator); % Gets numerator
       in appropriate form for Simulink
13 G_etaA_eta_Denominator = cell2mat(G_etaA_eta.Denominator); % Gets
      denominator in appropriate form for Simulink
14 G_alt = 10/(s+10);
15 G_alt_Numerator = cell2mat(G_alt.Numerator);
16 G_alt_Denominator = cell2mat(G_alt.Denominator);
17
18 %%% Transfer Functions - TASK 1 %%%
19 G_eta_q = ((Meta + (Mw_dot*Zeta))*s + ((Mw*Zeta) - (Meta*Zw)) )/((s^2)
       - ((VR*Mw_dot) + Zw + Mq)*s +((Mq*Zw) - (VR*Mw)));
20 G_eta_q_Numerator = cell2mat(G_eta_q.Numerator); % Gets numerator in
      appropriate form for Simulink
21 G_eta_q_Denominator = cell2mat(G_eta_q.Denominator); % Gets denominator
       in appropriate form for Simulink
22 G_eta_theta = 1/s * G_eta_q;
23 G_theta_h = 1/s * ((-Zeta*(s^2))+ Zeta*s*(Mq+VR*Mw_dot) + VR*(Mw*Zeta-
      Meta*Zw))/(s*(Meta+Mw_dot*Zeta) + (Mw*Zeta-Meta*Zw));
24 Gp = G_etaA_eta*G_eta_q;
25 G1 = Gp/s;
```

```
26
27 %%% Root Locus Design for Pitch Control - TASK 2 %%%
28 Wn = 3;
29 zeta = 0.5;
30 sigma = zeta*Wn;
31 Wd = Wn*sqrt(1-zeta^2);
32 s1 = -sigma + 1i * Wd;
33
34 %%% Plot The Root Locus for Pitch Control - TASK 3 %%%
35 figure;
36 rlocus(G1)
37 hold on;
38 plot(real(s1), imag(s1), 'rx', 'MarkerSize', 10, 'LineWidth', 2);
39 title('Root Locus of G1(s) - Pitch Control');
40 grid off;
41
42 %%% Evaluating the Complex Number of G1 - TASK 4 %%%
43 G1_s1 = evalfr(G1, s1);
44 mag_G1_s1 = abs(G1_s1);
45 angle_G1_s1 = angle(G1_s1);
46
47 %%% Find the location of the compensating Zero - TASK 5 %%%
48 alpha_c = pi - angle_G1_s1;
49 a = sigma + Wd / tan(alpha_c); % Finds the zero location, a, by
       accounting for the phase shortfall at point s1.
50
51 %%% Define G2 and show the point s1 on it's Root Locus Plot - TASK 6
       %%%
52 G2 = (s+a)*G1;
53 G2_s1 = evalfr(G2, s1);
54 angle_G2_s1 = angle(G2_s1); % May output a value close to, but not
       exactly pi - the criterion should still be met.
55 fprintf('The phase of G2 at s1 is: %.4f radians\n', abs(angle_G2_s1));
56 figure;
57 rlocus(G2);
```

```matlab
58 hold on;
59 plot(real(s1), imag(s1), 'rx', 'MarkerSize', 10, 'LineWidth', 2);
60 title('Root Locus of G2(s) - Pitch Control');
61 grid off;
62
63 %%% Check to see if the angle criterion is satisfied %%%
64 if abs(abs(angle_G2_s1) - pi) < 1e-3
65     disp('The angle criterion has been satisfied for G2 at s1 - the
          phase is approximately pi.');
66 else
67     disp('The angle criterion has not been satisfied.');
68 end
69
70 %%% Calculating the Gains - TASK 7 %%%
71 Kq = 1/(abs(G2_s1)); % Kq*mag_G1_s1 = 1. This satisfies the magnitude
      condition at s1. This means the CL pole will be at s1.
72 Ktheta = a*Kq; % Therefore, Krg = Kq.
73
74 %%% Determine the CLTF and find the Zeros and Poles - TASK 8 %%%
75 G_theta_c = feedback(G1,Kq*(s+a));
76
77 info = stepinfo(G_theta_c*5); % Gets the properties from the CLTF with
      a step change of 5 degrees.
78 overshoot = info.Overshoot;       % Percentage overshoot of CLTF
79 settlingTime = info.SettlingTime;   % Settling time in seconds of CLTF
80 finalValue = dcgain(G_theta_c*5);      % Final value of CLTF
81
82 figure;
83 [response, time] = step(G_theta_c*5);
84 plot(time, response, 'LineWidth',2);
85
86 [maxVal, maxIdx] = max(response);
87 maxTime = time(maxIdx); % Find the maximum response value and its time
88
89 grid on;
```

```matlab
90  title('Step Response of Pitch Attitude Control - Pitch Control');
91
92  annotationStr = sprintf('Maximum Overshoot: %.2f%%\nSettling Time: %.2f
        s\nFinal Value: %.2f', overshoot, settlingTime, finalValue); %
      Creates the annotation text with variable placeholders.
93
94  annotation('textbox', [0.3, 0.3, 0.3, 0.2], 'String', annotationStr, '
        FitBoxToText', 'on', 'BackgroundColor', '#ADD8E6', 'EdgeColor', '
        black', 'FontSize', 50); % Creates the annotation with a text box.
95
96  pole_G_altitude = pole(G_theta_c);
97  zero_G_altitude = zero(G_theta_c);
98  disp('The poles of G_theta_c are:');
99  disp(pole_G_altitude);
100 disp('The zeros of G_theta_c are:');
101 disp(zero_G_altitude);
102
103 %%% Define Altitude Loop Transfer Functions - TASK 9 %%%
104 a_c = 0.75;
105 G_prime = (1 / (1 + a_c*s)) * G_theta_c * G_theta_h;
106 G_star = G_prime * G_alt;
107
108 %%% Root Locus Design for Altitude Control - TASK 10 %%%
109 Wn_2 = 0.5;
110 zeta_2 = 0.5;
111 sigma_2 = zeta_2*Wn_2;
112 Wd_2 = Wn_2*sqrt(1-zeta_2^2);
113 s2 = -sigma_2 + 1i * Wd_2;
114
115 %%% Plot the Root Locus for Altitude Control - TASK 11 %%%
116 figure;
117 rlocus(G_star);
118 hold on;
119 plot(real(s2), imag(s2), 'rx', 'MarkerSize', 10, 'LineWidth', 2);
120 title('Root Locus of G*(s) - Altitude Control');
```

```matlab
121 grid off;
122
123 %%% Evaluating the Complex Number of G_star - TASK 12 %%%
124 G_star_s2 = evalfr(G_star, s2);
125 mag_G_star_s2 = abs(G_star_s2);
126 angle_G_star_s2 = angle(G_star_s2);
127
128 %%% Compute Zero Location for Altitude Control - TASK 13 %%%
129 alpha_c_prime = pi - angle_G_star_s2;
130 b1 = sigma_2 + Wd_2 / tan(alpha_c_prime);
131
132 %%% Define G3(s) and Root Locus Plot for Altitude Control - TASK 14 %%%
133 G3 = (s + b1) * G_star;
134 G3_s2 = evalfr(G3, s2);
135 angle_G3_s2 = angle(G3_s2);
136 fprintf('The phase of G3 at s2 is: %.4f radians\n', abs(angle_G3_s2));
137 figure;
138 rlocus(G3);
139 hold on;
140 plot(real(s2), imag(s2), 'rx', 'MarkerSize', 10, 'LineWidth', 2);
141 title('Root Locus of G3(s) - Altitude Control');
142 grid off;
143
144 %%% Check to see if the angle criterion is satisfied %%%
145 if abs(abs(angle_G3_s2) - pi) < 1e-3
146     disp('The angle criterion has been satisfied for G3 at s2 - the
          phase is approximately pi.');
147 else
148     disp('The angle criterion has not been satisfied.');
149 end
150
151 %%% Calculate the gain Kh_prime - TASK 15 %%%
152 kh_prime = 1/abs(G3_s2);
153
154 %%% Calculate the gain Kh - TASK 16 %%%
```

```
155 kh = kh_prime * b1;

156

157 %%% Compute Closed Loop Transfer Function - TASK 17 %%%

158 G_altitude = feedback(kh*(s+b1)*G_prime, G_alt);

159

160 info_2 = stepinfo(50 * G_altitude); % Gets the properties from the CLTF
        with a step change of 5 degrees.

161 overshoot_2 = info_2.Overshoot;        % Percentage overshoot of CLTF

162 settlingTime_2 = info_2.SettlingTime;   % Settling time in seconds of
        CLTF

163 finalValue_2 = dcgain(G_altitude*50);      % Final value of CLTF

164

165 figure;

166 [response_2, time_2] = step(50 * G_altitude);

167 plot(time_2, response_2, 'LineWidth',2);

168

169 [maxVal_2, maxIdx_2] = max(response);

170 maxTime_2 = time(maxIdx_2); % Find the maximum response value and its
        time

171

172 title('Step Response of Altitude Hold System - Altitude Control');

173 grid on;

174

175 annotationStr_2 = sprintf('Maximum Overshoot: %.2f%%\nSettling Time:
        %.2f s\nFinal Value: %.2f', overshoot_2, settlingTime_2,
        finalValue_2); % Creates the annotation text with variable
        placeholders.

176

177 annotation('textbox', [0.3, 0.3, 0.3, 0.2], 'String', annotationStr_2,
        'FitBoxToText', 'on', 'BackgroundColor', '#ADD8E6', 'EdgeColor', '
        black', 'FontSize', 50); % Creates the annotation with a text box.

178

179 pole_G_altitude = pole(G_altitude);

180 zero_G_altitude = zero(G_altitude);

181 disp('The poles of G_altitude are:');
```

```
182  disp(pole_G_altitude);

183  disp('The zeros of G_altitude are:');

184  disp(zero_G_altitude);

185

186  %%% Get the compensator and G_prime block for Simulink %%%

187

188  overallBlock = G_prime * kh_prime*(s+b1);

189  overallBlock_Numerator = cell2mat(overallBlock.Numerator);

190  overallBlock_Denominator = cell2mat(overallBlock.Denominator);

191

192  %%% END OF SCRIPT %%%
```

Originally, a requirement for this assignment was to also complete separate simulations inside of SIMULINK, but this element of the assessment has been subsequently removed. All the code relating to this part of the assessment remain in the script above.

Tasks 3, 6, 8, 11, 14 and 17 each provide a major output, or result, which can be used to monitor the progress of the simulation.

2.2.2 *Results*

First of all, the results from Task 3 will be discussed. Task 3 outputs a root locus plot, as seen in Figure 2.3.

Figure 2.3

The Root Locus of $G_1(s)$ - Pitch Control
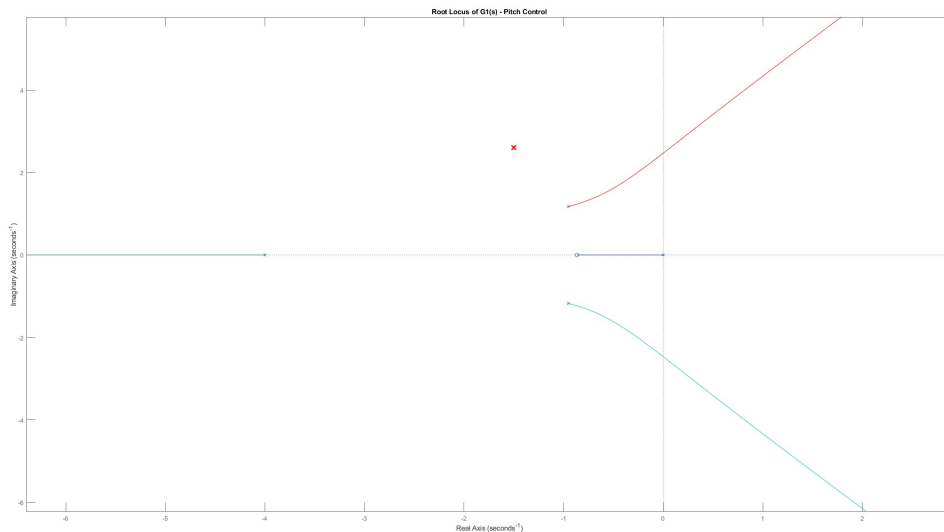
We can recall the equation of transfer function $G_1(s)$ in Equation 2.11. The root locus of G_1 has been plotted along side the complex number s_1. However, the complex number does not lie upon a branch of the root locus. This complex number must lie on a branch for the system to function correctly as they represent the location of the system's closed-loop poles. This is why in following tasks, the location of a compensating zero is found. When all the criteria are met, a compensating zero will move the branches of the root locus so that the complex number will lie upon them.

Task 6 acts a check for this, as it outputs the root locus plot of transfer function $G_2(s_1)$, Equation 2.16 shows it's mathematical representation. The important difference between $G_1(s)$ and $G_2(s_1)$ is that $G_2(s_1)$ has been compensated with a new zero. This results in the the plot shown in Figure 2.4.

Figure 2.4

The Root Locus of $G_2(s_1)$ - Pitch Control
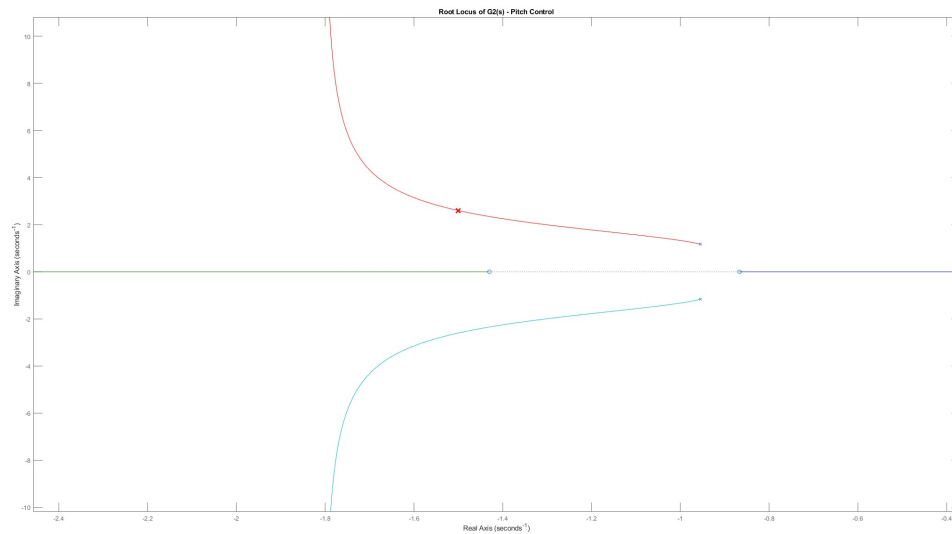
As seen in Figure 2.4, the root locus plot now intersects with the complex number s_1. This means that an acceptable closed-loop pole of the system has been found. Therefore, the step response of the pitch attitude system can be plotted. For this particular simulation, the aircraft experiences a $5°$ change to it's pitch attitude. The output of this step response is seen in Figure 2.5.
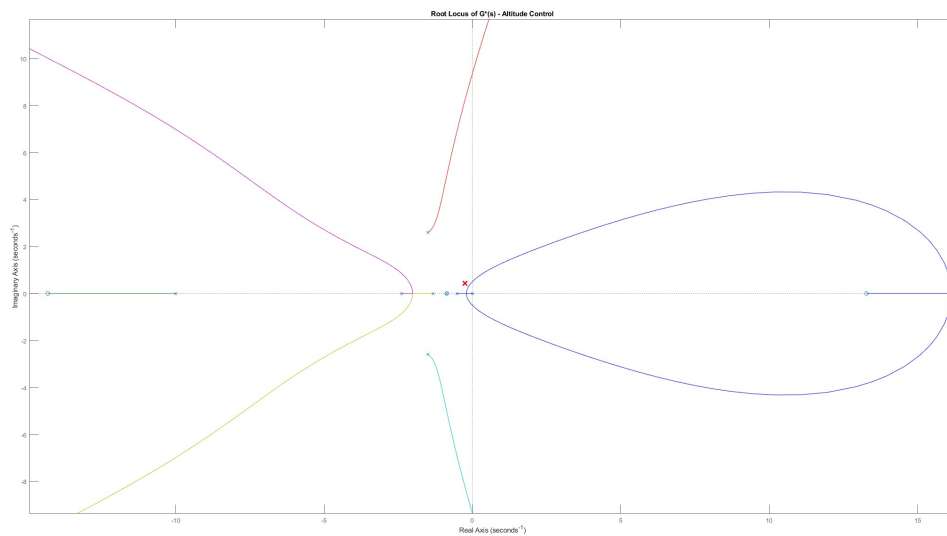
Figure 2.5

Step Response of Pitch Attidude System.
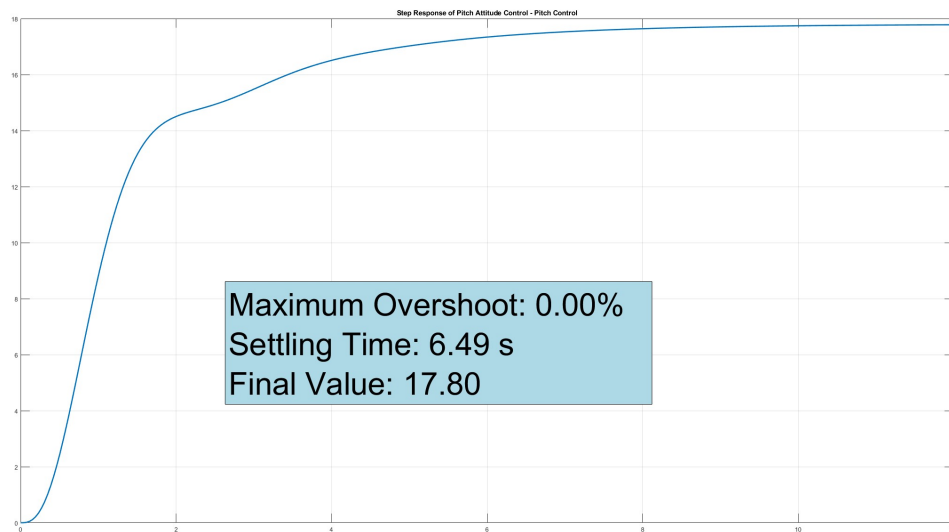
Figure 2.5 shows that the pitch control system responds to the 5° change by sending a pitch correction command to the altitude hold system after 6.49 seconds. The pitch needed to correct is 17.8°. All tasks from this point onwards are for the altitude hold section.

Tasks 11 and 14 are identical to tasks 3 and 6 in terms of their basic function. However, the root locus is designed for transfer function G^* which is defined in Equation 2.18. Therefore, the inital root locus plot is produced, for complex number s_2, as seen in Figure 2.6.

Figure 2.6

Root Locus Design for G^*.

Just as in task 3, the complex number does not lie on a branch for the root locus design. Therefore, by the method outlined in the theory, it is moved onto a branch, in an identical method to task 6. The result of task 14 can be seen in Figure 2.7

Figure 2.7

Root Locus Design for $G_3(s_2)$.
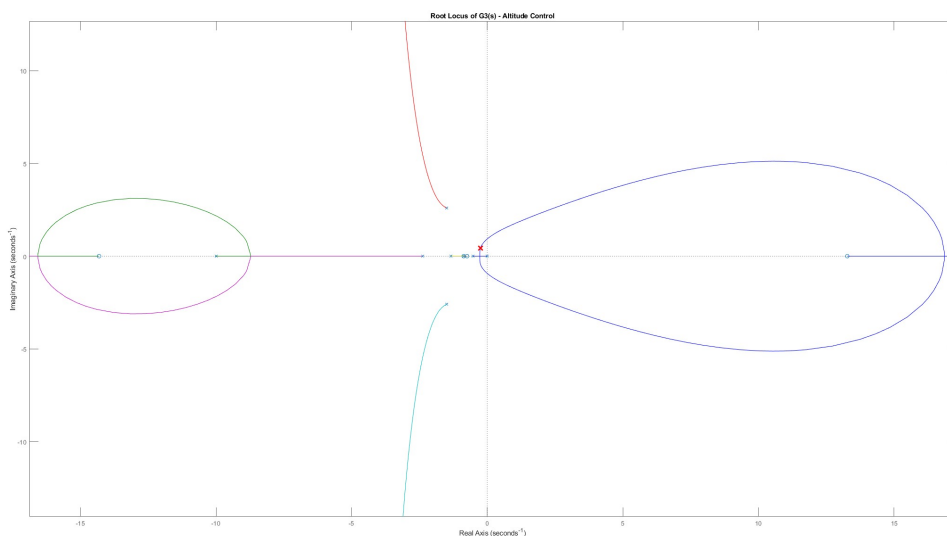
This means that the final design has been completed for the altitude hold system.

Therefore, the step response can be plotted in task 17. The following simulation shows the response, assuming the aircraft excperienced an altitude change of 50*m* due to the pitch change discussed in task 8. The step response can be seen in Figure 2.8.

Figure 2.8

Step Response of Altitude Hold System.
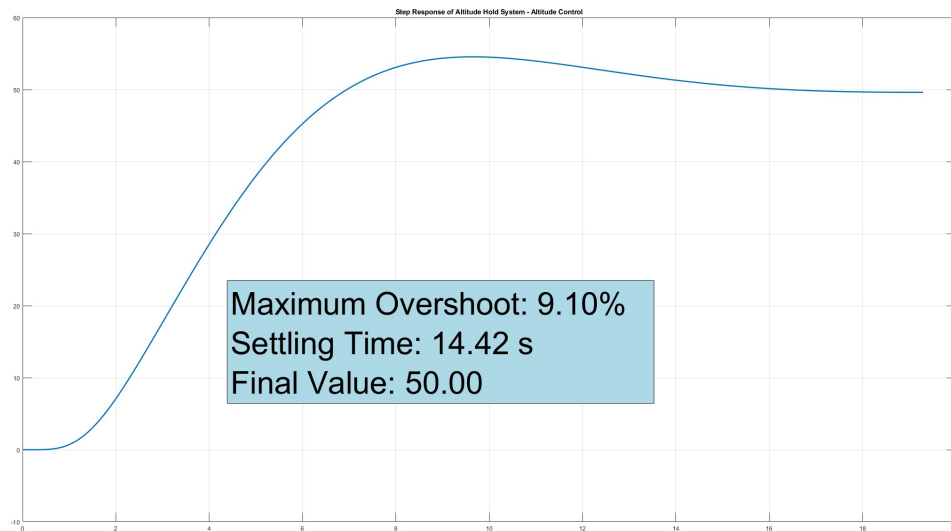
As seen in this step response, after 14.42 seconds, the aircraft has returned exactly to it's original height (the difference between the altitude change and final value is zero). There is a slight oscillation, as the aircraft initially over-corrects before returning to the original value.

2.3   Discussion

2.4   Conclusions

3   Appendix

List of Figures

List of Tables

List of Equations

References

Prasad, S., Ivanco, M. L., Ivanco, T. G., & Ancel, E. (2017). Nasa technical reports server

[Accessed: 2025-03-11].

https://ntrs.nasa.gov/api/citations/20170005878/downloads/20170005878.pdf