UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

SCHOOL OF MECHANICAL, INDUSTRIAL & AERONAUTICAL ENGINEERING

Control development of an Under Water Drone

MECN4029: Mechatronics II Group Assignment

Authors:	Student Numbers:
Adam Kassim	2281429
Kivash Jagarath	2369656
Reece Govender	2308030
Senthil Krishna	2429481

Lecturer: Dr Aarti Panday

Submitted: May 20th, 2024

Disclosure – Use of Artificial-Intelligence (AI) Generated Content

Select all applicable statements and complete the sections fully. Delete all statements that are not applicable.

1. Disclosure: No AI use

☒ I acknowledge that no AI tools/technologies (Grammarly, ChatGPT, Bard, Quillbot, OpenAI etc) were used in the completion of this assessment.

☒ ***I declare that the disclosure is complete and truthful.***

Student number: 2281429, 2369656, 2308030, 2429481.
Course code: MECN4029
Date: May 13th, 2024

Executive Summary

The mission profile was to hunt for geode formations on the ocean bed where the drone must image and survey the ocean floor and reefs at a maximum depth of 100 meters below sea level. The modelling of the system was done by first considering the sum of forces in the perturbation axes direction. These forces where then derived with respect to the relevant state variables for longitudinal motion resulting in the non-linear stability derivatives which were then further linearised to produce the state space model this allows for the analytical analysis of the dynamics of the underwater done through direct analytical analysis in MATLAB and analysis of the response through transfer functions in Simulink. Since the system was unstable, a controller analysis was done to compare the root locus and Proportional Integral Derivate method. The two methods exhibited their own weaknesses and strengths: gain was more reasonable for the root locus at 2483.4 as compared to a proportional gain of 743420 and derivative gain of 299446; Proportional Derivative system was more accurate to the set point whereas root locus has a maximum overshoot percentage of 10.3%. Proportional Derivative system was chosen as the mission profile required an accurate and responsive controller to better image capture, but further studies into the disturbance response of the system are required to determine if the gain causes uncontrollable response.

Table of Contents

List of tables

List of figures

Nomenclature

Symbol	Description
$\bar{\bar{c}}$	Mean Aerodynamic Chord [m]
∞	Infinity
c	Viscous damping coefficient [N.s/rad]
C_{do}	Zero lift drag coefficient
Cl	Lift coefficient
C_m	Pitching moment
K_e	Back emf proportionality constant [V.s/rad]
K_T	Motor torque constant [N.m/A]
L_a	Motor inductance [H]
q	Pitch rate [rad/s]
R_a	Motor Resistance [Ω]
S	Wing area [m^2]
V_0	Velocity [m/s]
θ	Pitch attitude [rad]
ω_p	Rotational speed [rad/s]

Moments of Inertia

Symbol	Description
I_{yy}	Pitch moment of inertia [$kg.m^2$]
J_p	Propellor and shaft moment of inertia [$kg.m^2$]

Derivatives

Symbol	Description
M_q	Pitching moment to pitch rate
M_u	Pitching moment due to velocity
M_w	Pitching moment due to incidence
$M_{\dot{w}}$	Pitching moment to downwash lag
M_τ	Pitching moment due to thrust
X_q	Axial force due to pitch rate
X_u	Axial force due to velocity
X_w	Axial force due to incidence
$X_{\dot{w}}$	Axial force due to downwash lag
X_τ	Axial force due to thrust
Z_q	Normal force due to pitch rate
Z_u	Normal force due to velocity
Z_w	Normal force due to incidence
$Z_{\dot{w}}$	Normal force due to downwash lag
Z_τ	Normal force due to thrust

1. Introduction

1.1 Mission Profile

To hunt for geode formations in on the ocean bed. A drone is requested to survey the area for local deposits for undersea conservation. The drone is required to operate 100 metres below the ocean's surface and operate 4 to 6 metres above the surface of the ocean bed to provide a clear view of the seabed for image capturing while staying out of the strong currents that occur at higher distances above the sea floor. The ocean environment tends to have uneven surfaces resulting in the drone wanting to pitch to effectively capture images of geode and mineral formations that occur in the sea floor. This is also useful to scientist who wish to image sea life that may live in these cracks. The drone is required to capable of reaching a pitch angle of 10 ° degrees within a space of time of 3 seconds to allow for easy image capturing. The system is also required to be capable of following a specimen fish that may tend to move in a sinusoidal pattern.

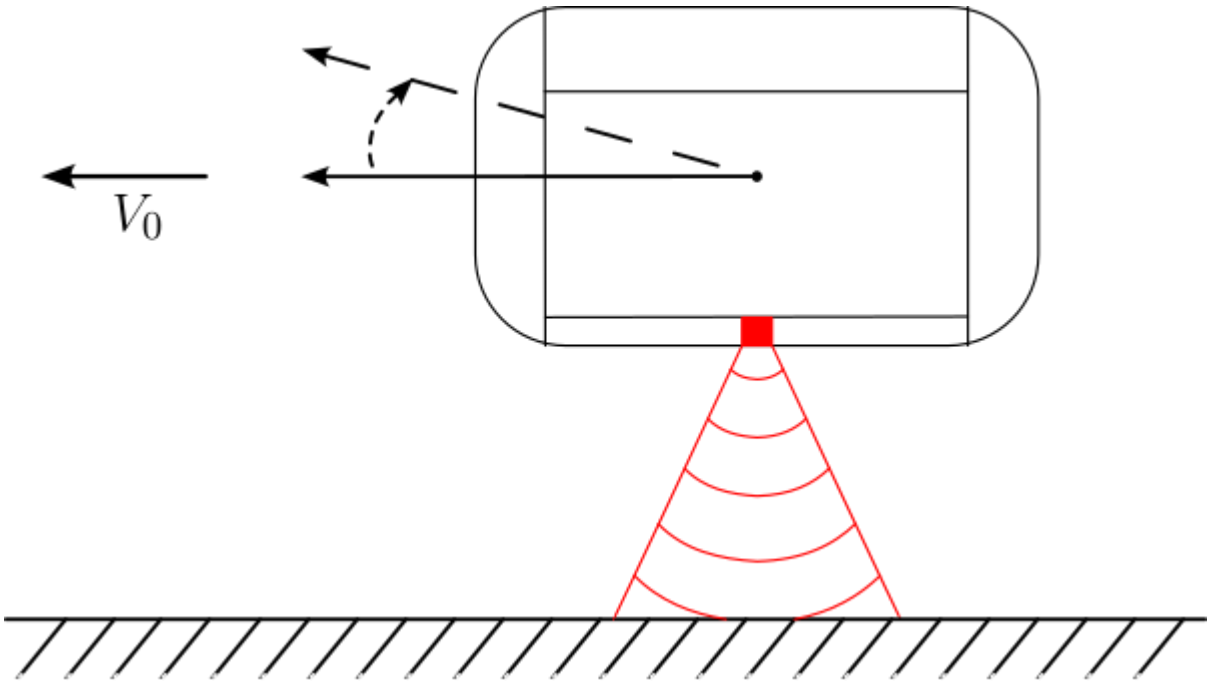


Figure 1: Diagram of the mission parameters.

A base drone body has been selected that is capable of operating at this depth has been chosen. But due to changes for the imaging equipment the resulting in the drone having the defining dimensions shown in Figure 3 and the inertial properties.

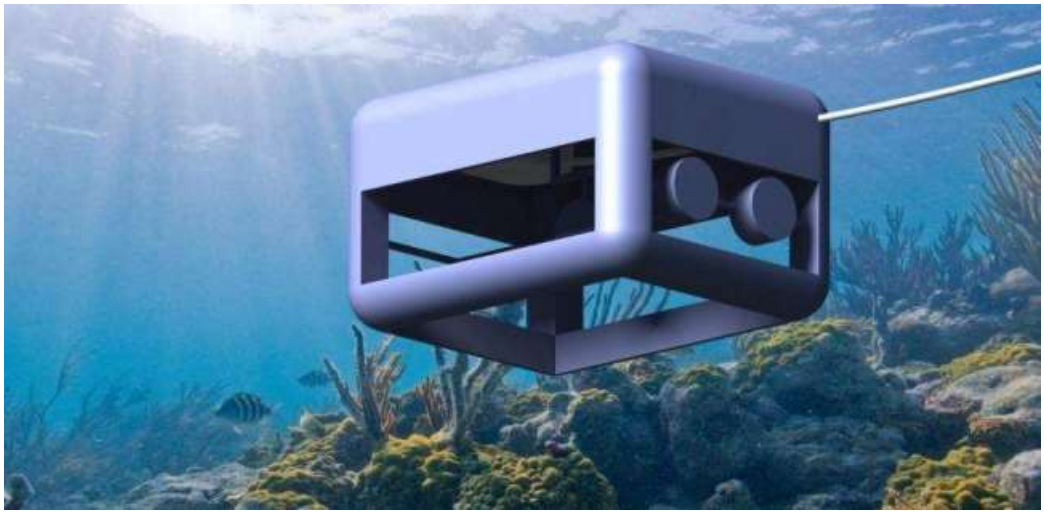


Figure 2: Rendering of the drone base to be used. [1]

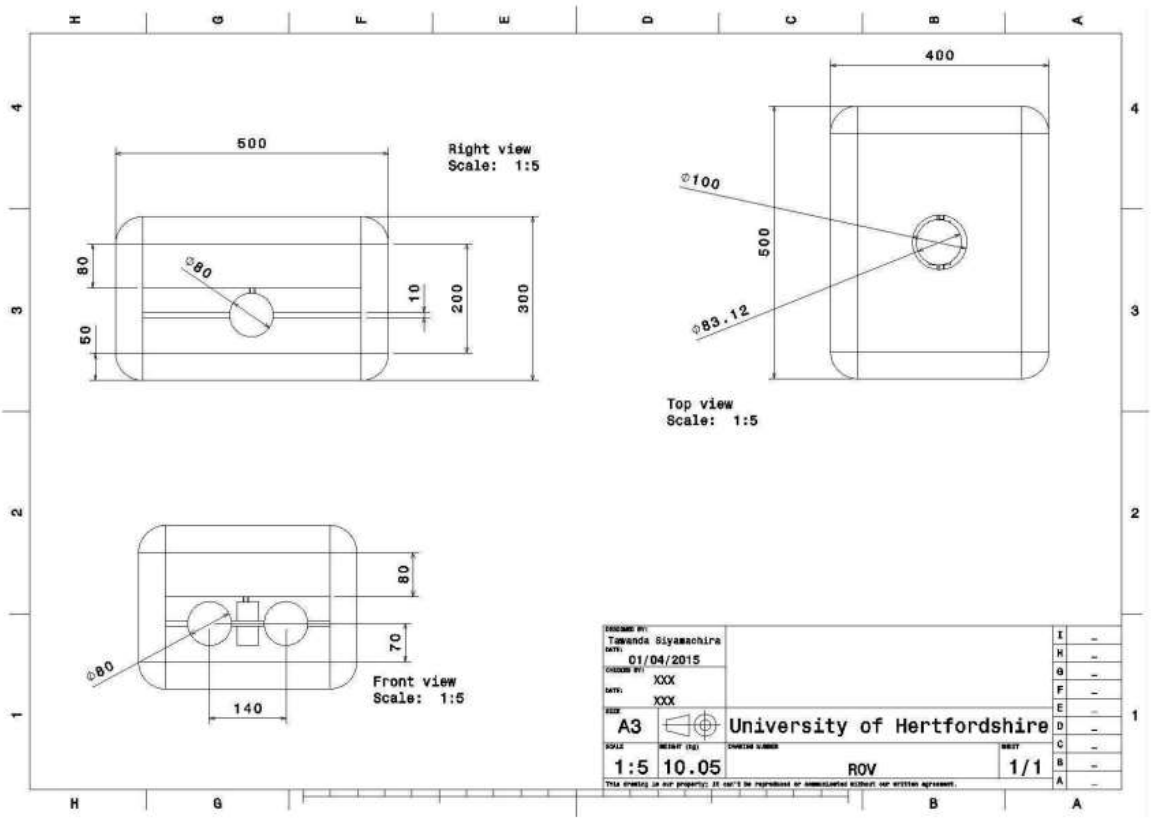


Figure 3: 3-view of the body geometry of the underwater drone model. [1]

Table 1: Centre of gravity and centre of buoyancy positions. [1]

	Centre of gravity	Centre of buoyancy
X position [m]	0.412	0.412
Y position [m]	0.128	0.128
Z position [m]	0.2	0.25

Table 2: Mass and inertia properties of the underwater drone. [1]

Property	Value
Mass [kg]	10
Length [m]	0.5
Breath [m]	0.3
Height [m]	0.4
Ixx [kg.m^2]	0.466
Ixy [kg.m^2]	0.007
Iyy [kg.m^2]	0.642
Izz [kg.m^2]	0.766

The dynamic properties relating to drag, lift and pitching moment of the underwater drone are presented as follows and are derived through CFD modelling in [1].

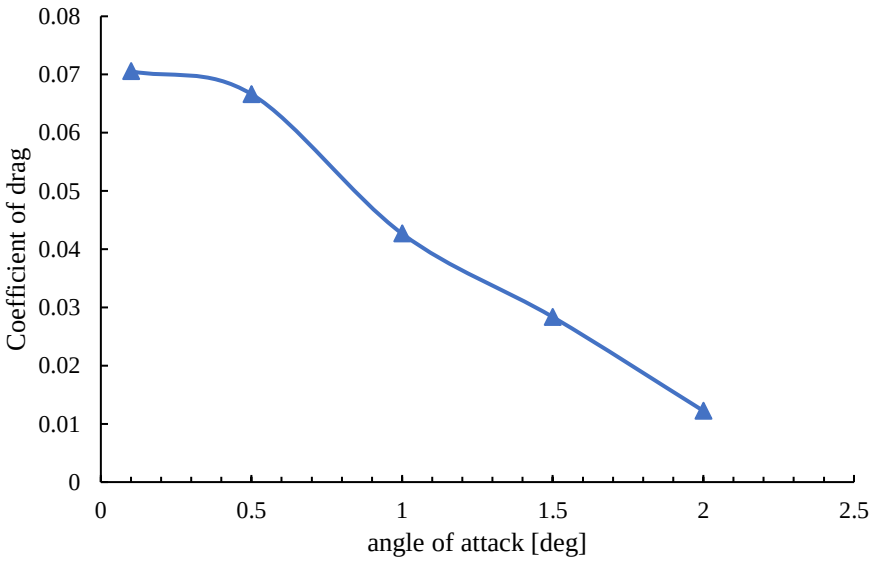


Figure 4: Plot of coefficient of drag versus angle of attack. [1]

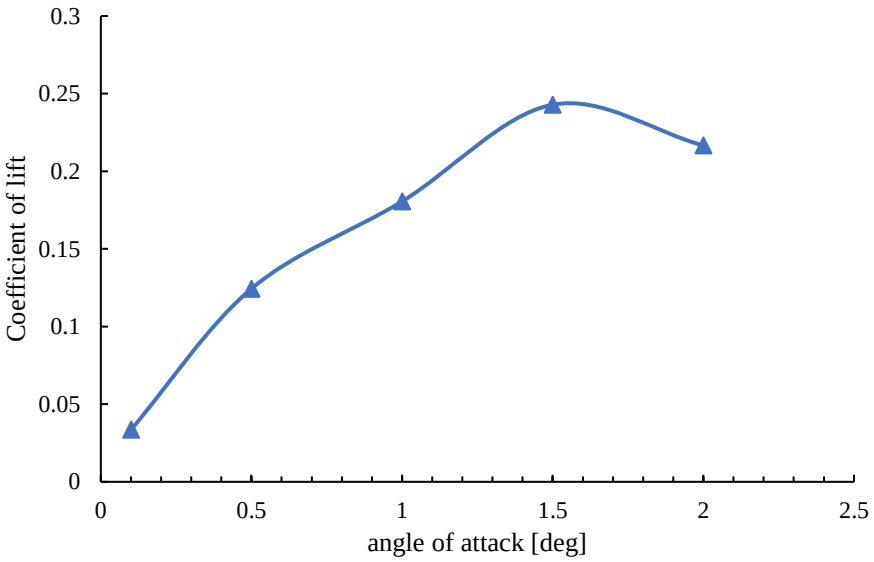


Figure 5;Plot of coefficient of lift versus angle of attack. [1]

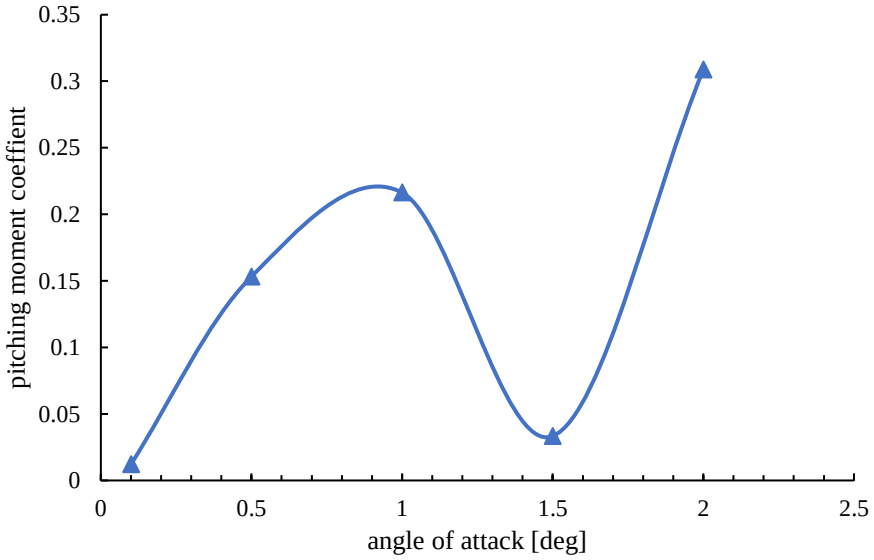


Figure 6: Plot of pitching moment versus angle of attack. [1]

2 System Modelling

2.1 Vehicle dynamics

The dynamic motion of the model can be described by Figure 7. It demonstrates the basic forces acting on the body why the drone is in a perturbed state from the path by angle θ.

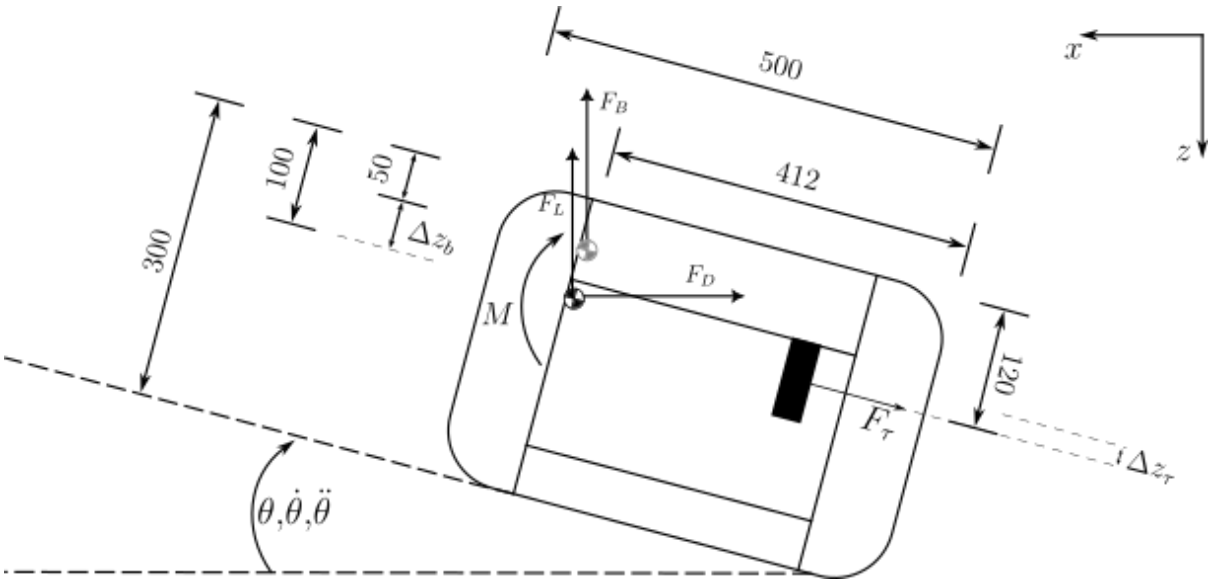


Figure 7: Force diagram of the underwater drone.

The following equations represent the sum of forces or moments in that direction and will be used to generate. The dimensional stability derivatives and the dimensional control derivatives used to model the vehicle dynamics.

Sum of forces in the x-direction,

$$X = F_L \sin\theta - F_D \cos\theta + F_\tau$$
$$X = \frac{1}{2}\rho V^2 S(C_L \sin\theta - C_D \cos\theta) + F_\tau \quad (2.1)$$

Sum of forces in the z-direction,

$$Z = -(F_L cos\theta + F_D sin\theta) + F_B$$
$$Z = -\frac{1}{2}\rho V^2 S(C_L \sin\theta + C_D \cos\theta) + F_B \quad (2.2)$$

Sum of moments in the clockwise direction,

$$M = \frac{1}{2}\rho V^2 S\bar{c} C_m + F_\tau z_\tau - F_B sin\theta z_b \quad (2.3)$$

To consider the dynamics of the model due to response the variables of interest, specifically the perturbation velocities u and w and we are interested in the pitch, θ, and the pitch rate, q. The response equations can be represented using stability and control derivative coefficients as follows [2]:

$$\begin{aligned} m\dot{u} &= X_u u + X_w w + (X_q - m\,W_e)q - mg\cos\theta_e\theta + X_\tau\tau \\ m\dot{w} &= Z_u u + Z_w w + (Z_q - mU_e)q - mg\sin\theta_e\theta + Z_\tau\tau \\ I_{yy}\dot{q} &= M_u u + M_q w + M_q q + M_\theta\theta + M_\tau\tau \\ \dot{\theta} &= q \end{aligned} \qquad (2.4)$$

The additional equation is an auxiliary equation relating pitch rate to attitude rate for small perturbations.

The set of equations (2.4) can be reduced into a state space representation as shown in (2.5) [2].

$$\boldsymbol{m\dot{x}(t) = A'x(t) + B'u(t)} \qquad (2.5)$$

Where $\mathbf{x}^T$ = [u w q θ] , and $\mathbf{u}^T$ = [τ] and, [2]

$$\boldsymbol{m} = \begin{bmatrix} m & 0 & 0 & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & I_{yy} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},\ \boldsymbol{A'} = \begin{bmatrix} X_u & X_w & X_q - m\,W_e & -mg\cos\theta_e \\ Z_u & Z_w & Z_q - mU_e & -mg\sin\theta_e \\ M_u & M_q & M_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\text{and } \boldsymbol{B'} = \begin{bmatrix} X_\tau \\ Z_\tau \\ M_\tau \\ 0 \end{bmatrix}$$

The actual modelling of the stability and the control derivatives requires some development of some basic relations.

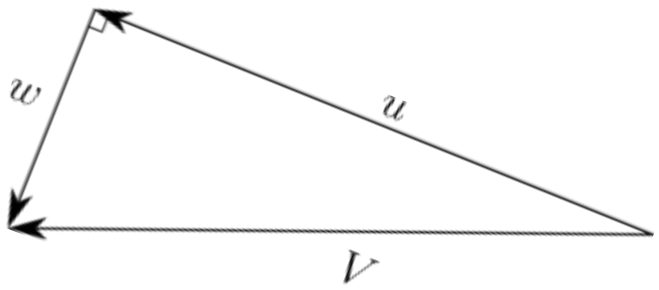


Figure 8: Velocity triangle of the perturbed system axis.

Using Figure 8, The following set of relationships can be made:

$$V^2 = U^2 + W^2 \tag{2.6}$$

$$U = U_e + u = V\cos(\theta)$$
$$W = W_e + w = V\sin(\theta) \tag{2.7}$$

Where U_e and W_e represent steady velocity components and u and w represents velocity perturbations [2].

From differentiating (2.6) implicitly it can be shown that.

$$\frac{\partial V}{\partial U} = \frac{U}{V} = \cos(\theta) \cong 1 \tag{2.8}$$

$$\frac{\partial V}{\partial W} = \frac{W}{V} = \sin(\theta) \cong 0 \tag{2.9}$$

From differentiating (2.7) and (2.8);

$$\frac{\partial \theta}{\partial U} \equiv \frac{\partial \alpha}{\partial U} = \frac{-\sin(\theta)}{V} \cong 0 \tag{2.10}$$

$$\frac{\partial \theta}{\partial W} \equiv \frac{\partial \alpha}{\partial W} = \frac{\cos(\theta)}{V} \cong \frac{1}{V} \tag{2.11}$$

2.1.1 X_u, Axial force derivatives due to axial velocity

The method presented is derived and adapted from Flight dynamics principles by M.V. Cook[2].
Differentiate (2.1) to obtain the non-linear form:

$$X_u = \frac{\partial X}{\partial U} = \frac{1}{2}\rho V^2 S\left(\frac{\partial C_L}{\partial U}\sin\theta + C_L\cos\theta\frac{\partial \theta}{\partial U} - \frac{\partial C_D}{\partial U}cos\theta + C_D\sin\theta\frac{\partial \theta}{\partial U}\right) + \rho VS\frac{\partial V}{\partial U}(C_L\sin\theta - C_D\cos\theta) + \frac{\partial F_\tau}{\partial U} \tag{2.12}$$

Simplifying sightly and substituting for exact relations where possible gives,

$$X_u = \frac{\partial X}{\partial U} = \frac{1}{2}\rho V^2 S\left(\frac{\partial C_L}{\partial U}\sin\theta - \frac{\partial C_D}{\partial U}cos\theta - \frac{\sin(\theta)}{V}(C_D\sin\theta + C_L\cos\theta)\right) + \rho VS\frac{\partial V}{\partial U}(C_L\sin\theta - C_D\cos\theta) + \frac{\partial F_\tau}{\partial U} \tag{2.13}$$

This can be linearised using the approximations produced in (2.8) to (2.11).

$$X_u = \frac{\partial X}{\partial U} = -\frac{1}{2}\rho V^2 S \frac{\partial C_D}{\partial U} - \rho V S C_D + \frac{\partial F_\tau}{\partial U} \quad (2.14)$$

Now,

$$\frac{\partial C_D}{\partial U} = \frac{\partial C_D}{\partial V}\frac{\partial V}{\partial U} = \frac{\partial C_D}{\partial V} \quad (2.15)$$

And similarly,

$$\frac{\partial F_\tau}{\partial U} = \frac{\partial F_\tau}{\partial V} \quad (2.16)$$

Giving the linearised form,

$$X_u = \frac{\partial X}{\partial U} = -\frac{1}{2}\rho V^2 S \frac{\partial C_D}{\partial V} - \rho V S C_D + \frac{\partial F_\tau}{\partial V} \quad (2.17)$$

2.1.2 Z_u, Normal force due to axial velocity

Differentiating (2.2) To obtain non-linear form:

$$Z_u = \frac{\partial Z}{\partial U} = -\frac{1}{2}\rho V^2 S\left(\frac{\partial C_L}{\partial U}\cos\theta + \frac{\partial C_D}{\partial U} sin\theta + \frac{\partial \theta}{\partial U}(C_D\cos\theta - C_L\sin\theta)\right) - \rho V S \frac{\partial V}{\partial U}(C_L\cos\theta + C_D\sin\theta) \quad (2.18)$$

This can be linearised using the approximations produced in (2.8) to (2.11).

$$Z_u = \frac{\partial Z}{\partial U} = -\frac{1}{2}\rho V^2 S \frac{\partial C_L}{\partial U} - \rho V S C_L \quad (2.19)$$

Now,

$$\frac{\partial C_L}{\partial U} = \frac{\partial C_L}{\partial V}\frac{\partial V}{\partial U} = \frac{\partial C_L}{\partial V} \quad (2.20)$$

Resulting in the linearised equation,

$$Z_u = \frac{\partial Z}{\partial U} = -\frac{1}{2}\rho V^2 S \frac{\partial C_L}{\partial V} - \rho V S C_L \quad (2.21)$$

2.1.3 X_w, Axial force due to normal velocity

Differentiating (2.1) to obtain non-linear form:

$$X_w = \frac{\partial X}{\partial W} = \frac{1}{2}\rho V^2 S\left(\frac{\partial C_L}{\partial W}\sin\theta - \frac{\partial C_D}{\partial W} cos\theta + \frac{\partial \theta}{\partial W}(C_L\cos\theta - C_D\sin\theta)\right) + \rho V S \frac{\partial V}{\partial W}(C_L\cos\theta - C_D\sin\theta) + \frac{\partial F_\tau}{\partial W} \quad (2.22)$$

This can be linearised using the approximations produced in (2.8) to (2.11).

$$X_w = \frac{\partial X}{\partial W} = \frac{1}{2}\rho V^2 S\left(C_L \frac{\partial \theta}{\partial W} - \frac{\partial C_D}{\partial W}\right) + \frac{\partial F_\tau}{\partial W}$$

(2.23)

Using, noting $\alpha = \theta$, where α is the vehicle path angle.

$$\frac{\partial C_D}{\partial W} = \frac{\partial C_D}{\partial \theta}\frac{\partial \theta}{\partial W} = \frac{1}{V}\frac{\partial C_D}{\partial \theta} \equiv \frac{1}{V}\frac{\partial C_D}{\partial \alpha}$$

(2.24)

And similarly, but the thrust variation with angle of attack is negligible.

$$\frac{\partial F_\tau}{\partial W} = \frac{1}{V}\frac{\partial F_\tau}{\partial \alpha} = 0$$

(2.25)

Results in the linearised term,

$$X_w = \frac{\partial X}{\partial W} = \frac{1}{2}\rho V S\left(C_L - \frac{\partial C_D}{\partial \alpha}\right)$$

(2.26)

2.1.4 Z_w, Normal force due to normal velocity

Differentiating (2.2) To obtain non-linear form:

$$Z_w = \frac{\partial Z}{\partial W} = -\frac{1}{2}\rho V^2 S\left(\frac{\partial C_L}{\partial W}\cos\theta + \frac{\partial C_D}{\partial W} sin\theta + \frac{\partial \theta}{\partial W}(C_D\cos\theta - C_L\sin\theta)\right) - \rho V S\frac{\partial V}{\partial W}(C_L\cos\theta + C_D\sin\theta)$$

(2.27)

This can be linearised using the approximations produced in (2.8) to (2.11).

$$Z_w = \frac{\partial Z}{\partial W} = -\frac{1}{2}\rho V^2 S\left(\frac{\partial C_L}{\partial W} + \frac{C_D}{V}\right)$$

(2.28)

Using, noting $\alpha = \theta$.

$$\frac{\partial C_L}{\partial W} = \frac{\partial C_L}{\partial \theta}\frac{\partial \theta}{\partial W} = \frac{1}{V}\frac{\partial C_L}{\partial \theta} \equiv \frac{1}{V}\frac{\partial C_L}{\partial \alpha}$$

(2.29)

Results in the linearised term,

$$Z_w = \frac{\partial Z}{\partial W} = -\frac{1}{2}\rho V S\left(\frac{\partial C_L}{\partial \alpha} + C_D\right)$$

(2.30)

2.1.5 M_u, Pitching moment due to axial velocity

Differentiating (2.3)To obtain non-linear form:

$$M_u = \frac{\partial M}{\partial U} = \frac{1}{2}\rho V^2 S\bar{\bar{c}}\frac{\partial C_m}{\partial U} + \rho V\frac{\partial V}{\partial U}S\bar{\bar{c}}C_m - F_B cos\theta\frac{\partial \theta}{\partial U}z_b$$

(2.31)

This can be linearised using the approximations produced in (2.8) to (2.11).

$$M_u = \frac{\partial Z}{\partial U} = \frac{1}{2}\rho V^2 S\bar{\bar{c}}\frac{\partial C_m}{\partial U} + \rho V S\bar{\bar{c}}C_m$$

(2.32)

And,

$$\frac{\partial C_m}{\partial U} = \frac{\partial C_m}{\partial V}\frac{\partial V}{\partial U} = \frac{\partial C_m}{\partial V}$$

(2.33)

In the limit as the perturbation tends to zero the pitching moment coefficient in the second term tends to the steady state equilibrium value which is zero. Resulting in:

$$M_u = \frac{\partial Z}{\partial U} = \frac{1}{2}\rho V^2 S\bar{\bar{c}}\frac{\partial C_m}{\partial V}$$

(2.34)

2.1.6 M_w, Pitching moment due to normal velocity

Differentiating (2.3)To obtain non-linear form:

$$M_w = \frac{\partial M}{\partial W} = \frac{1}{2}\rho V^2 S\bar{\bar{c}}\frac{\partial C_m}{\partial W} + \rho V\frac{\partial V}{\partial W}S\bar{\bar{c}}C_m - \ F_B cos\theta\frac{\partial \theta}{\partial W}z_b$$

(2.35)

This can be linearised using the approximations produced in (2.8) to (2.11).

And,

$$\frac{\partial C_m}{\partial W} = \frac{\partial C_m}{\partial \theta}\frac{\partial \theta}{\partial W} = \frac{\partial C_m}{\partial \theta} \equiv \frac{1}{V}\frac{\partial C_m}{\partial \alpha}$$

(2.36)

Resulting in the following Linearised equation,

$$M_w = \frac{\partial Z}{\partial W} = \frac{1}{2}\rho V S\bar{\bar{c}}\frac{\partial C_m}{\partial W} \ - \ \frac{1}{V}F_B cos\theta z_b$$

(2.37)

2.1.7 X_τ, Z_τ, M_τ, The control derivatives

The control derivatives can be found by differentiating the correspond directional force function to it corresponding thrust coefficient.

Control stability coefficient in the x-direction:

$$X_\tau = \frac{\partial X}{\partial F_\tau} = \ 1$$

(2.38)

Control stability coefficient in the z-direction:

$$Z_\tau = \frac{\partial Z}{\partial F_\tau} = \ 0$$

(2.39)

Control stability coefficient in the clockwise direction:

$$M_\tau = \frac{\partial Z}{\partial F_\tau} = z_\tau$$

(2.40)

2.1.8 The pitch rate derivatives

All derivatives related to pitch rate and downwash lag are equal to zero due to the design of the drone not including a tail plane.

2.2 Linear equations of motion

To solve the equation of motion the state space equation (2.5) is re-arranged to give (2.41).

$$\boldsymbol{y(t)} = \boldsymbol{M^{-1}Ax(t)} + \boldsymbol{M^{-1}Bu(t)}$$

(2.41)

The Laplace transform of the pitch response of the vehicle to thrust inputs has been in (2.42).

$$\frac{\Theta(s)}{T(s)} = \frac{0.002611s^2 + 0.0121s + 0.01389}{s^4 + 5.178s^3 + 10.18s^2 + 7.663s - 0.2603}$$

(2.42)

2.3 Motor Dynamics

2.3.1 Motor modelling

A transfer function was obtained for the motors shown in Figure 9 for the input voltage (e_a) to the rotational speed of the propellor blades (ω_p). For this analysis it is assumed that the startup torque of the motor is negligible.

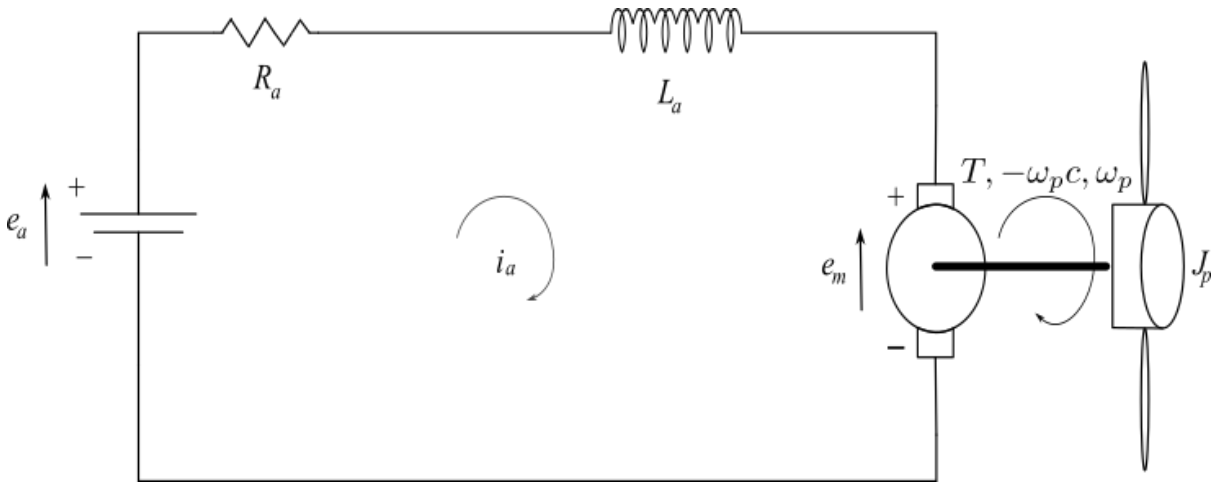


Figure 9: Armature-controlled propellor diagram.

Mechanical System

The torque is related to the rotational speed of the propellors by (2.43):

$$T = J_p \dot{\omega}_p + c\omega_p \tag{2.43}$$

Taking the Laplace transform of (2.43) and rearranging to find:

$$T(s) = \left[J_p s + c\right]\Omega_p(s) \tag{2.44}$$

Electrical System

Applying Kirchhoff's Voltage Law:

$$e_a = R_a i_a + L_a \frac{di_a}{dt} + e_m \tag{2.45}$$

Taking the Laplace transform of (2.45) and rearranging to find the current:

$$I_a(s) = \left[\frac{1}{L_a s + R_a}\right][E_a(s) - E_m(s)] \tag{2.46}$$

The torque of the motor and the back e.m.f (e_m) given by:

$$T = K_T i_a \tag{2.47}$$

$$e_m = K_e \omega_p \tag{2.48}$$

Taking the Laplace transform of (2.47) and (2.48):

$$T(s) = K_T I_a(s) \quad (2.49)$$

$$E_m(s) = K_e \Omega_p(s) \quad (2.50)$$

Substituting (2.50) into (2.46) and the result into (2.49):

$$T(s) = \left[\frac{K_T}{L_a s + R_a}\right]\left[E_a(s) - K_e \Omega_p(s)\right] \quad (2.51)$$

Equating (2.44) and (2.51):

$$\left[\frac{K_T}{L_a s + R_a}\right]\left[E_a(s) - K_e \Omega_p(s)\right] = \left[J_p s + c\right]\Omega_p(s) \quad (2.52)$$

After rearranging (2.52) the final transfer function of the motor voltage as an input to the rotational speed of the propellor can be found to be:

$$\frac{\Omega_p(s)}{E_a(s)} = \frac{K_T}{\left(J_p L_a\right)s^2 + \left(J_p R_a + L_a c\right)s + \left(R_a c + K_e\right)} \quad (2.53)$$

The inputs for (2.53) can be found in Table 3.

Table 3: Inputs for propellor transfer function.

Input	Value
K_T	0.05
J_p	1.3333e-4
L_a	0.008
R_a	1.9
c	0.01
K_e	0.06

2.3.2 Thrust equation.

The thrust was assumed to vary linearly with the rotation speed of the propellors such that:

$$T = A_T \omega_p$$

(2.54)

If (2.54) is converted to the frequency domain, a transfer function that relates the thrust to the rotational speed of the propellor by (2.55):

$$\frac{T(s)}{\Omega_p(s)} = A_T \ = 0.01309 \text{ Ns/rad}$$

(2.55)

2.4 Pitch and voltage transfer function

The block diagram for the plant that combines the three transfer functions takes an input voltage and describes the vehicles response in pitch is shown in Figure 10.

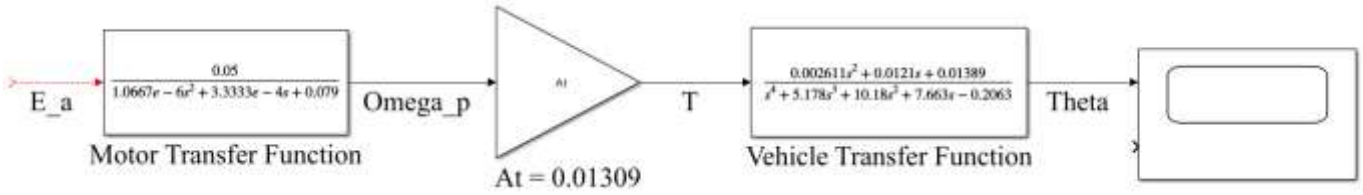


Figure 10: Plant diagram.

The transfer functions may be combined to form a single transfer function that describes the plant. The combined transfer function is shown in Figure 11.

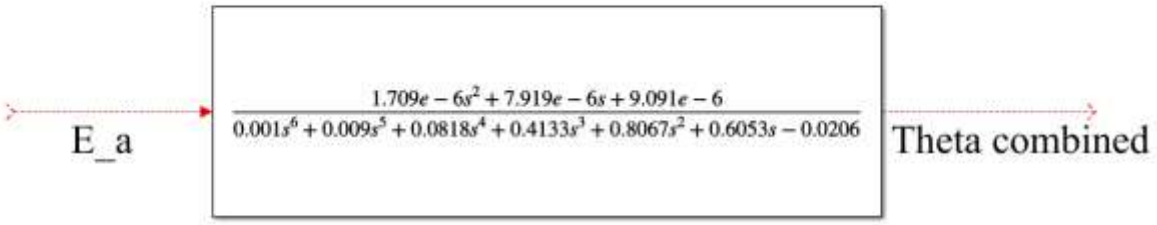


Figure 11: Combined transfer function block diagram.

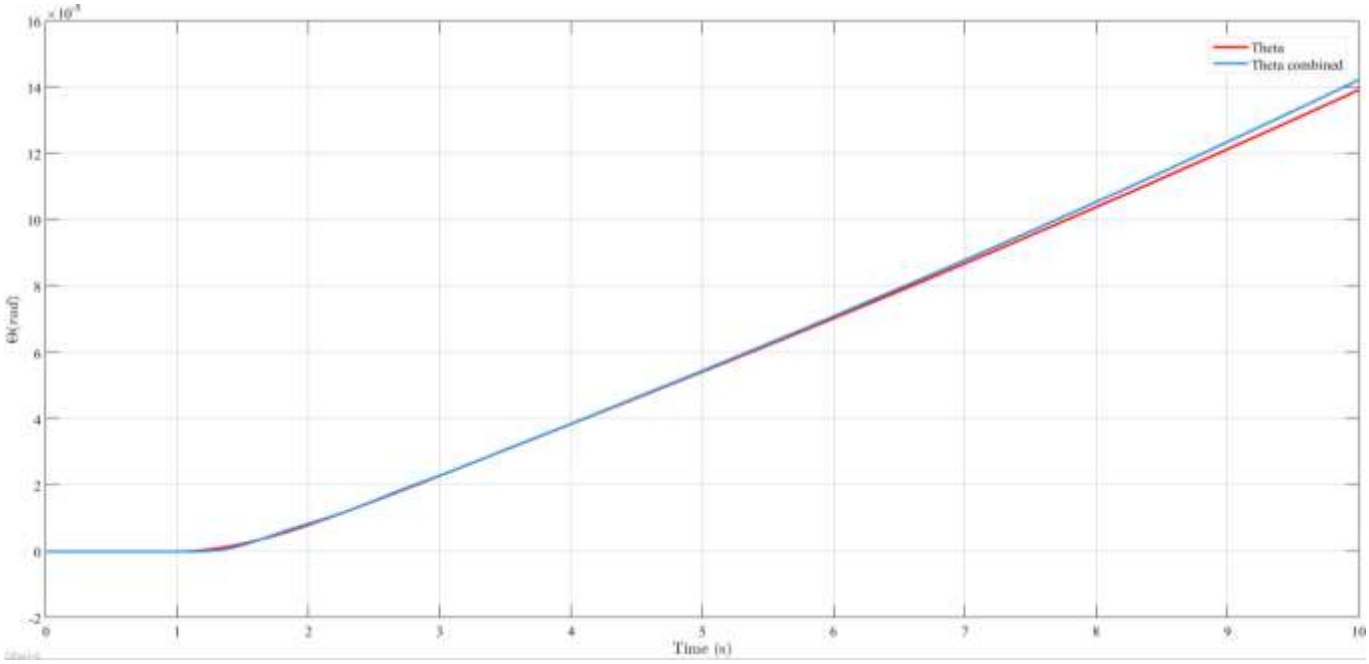


Figure 12: Step response comparison.

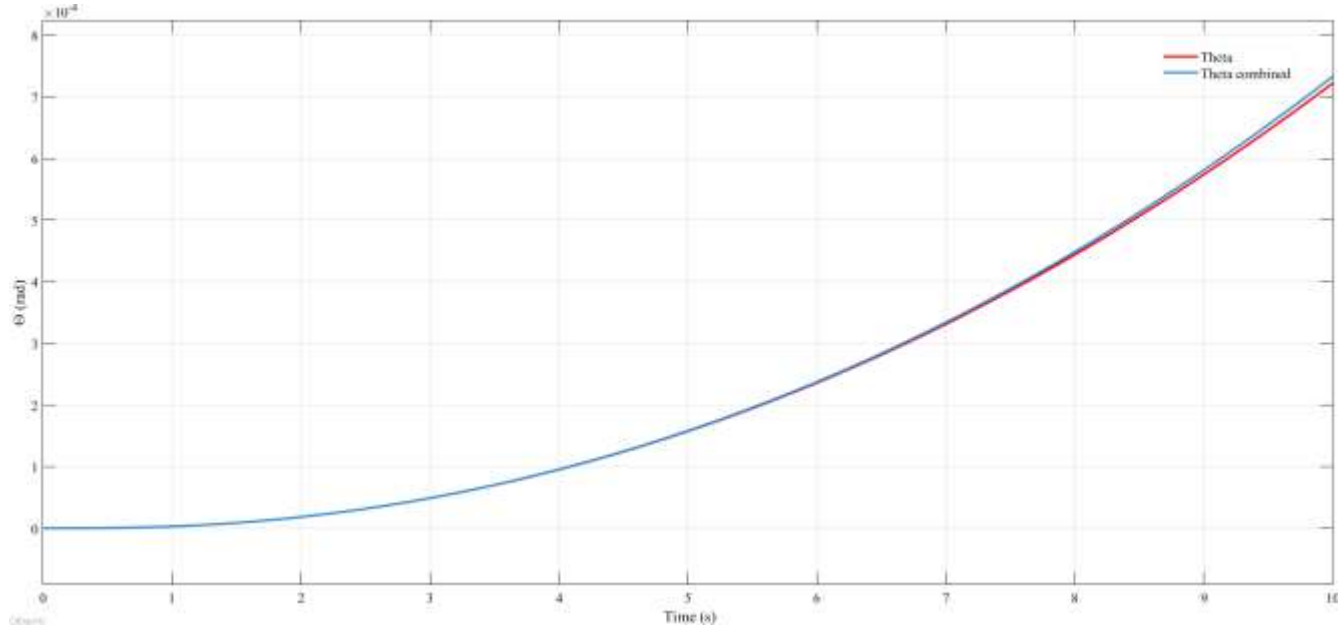


Figure 13: Ramp response comparison.

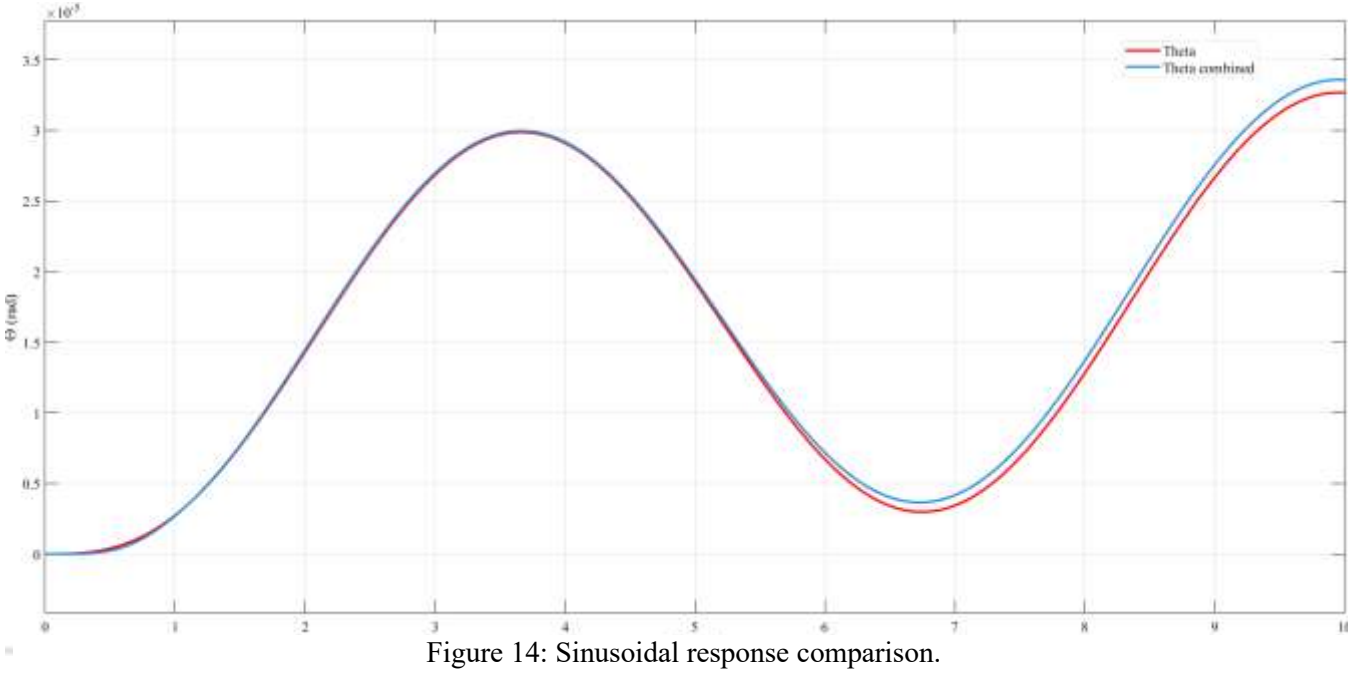


Figure 14: Sinusoidal response comparison.

Figure 12, Figure 13 and Figure 14 show the comparison of the three separate block diagrams and the combined plot in response to a step, ramp, and sinusoidal input respectively. The combined plant only varies from the separated plant by a maximum of only 1×10^{-5} rad in the 10 second time frame analysed and may be used for further analysis.

3 Open loop stability analysis

A plot of the poles of the open loop pitch response of the vehicle in Figure 16 shows that the system has a single right half plane pole and is unstable.

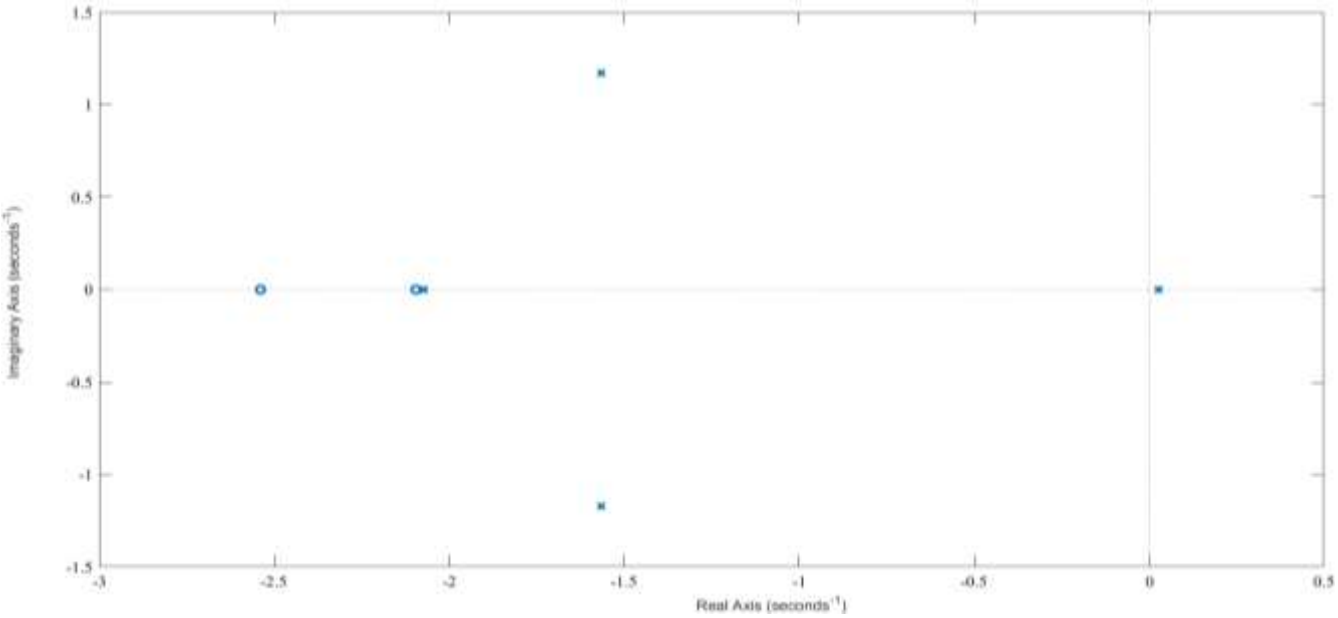


Figure 15: Vehicle transfer function poles.

The response of the system to a 1N thrust step, ramp and sinusoidal input is shown in Figure 16, Figure 17 and Figure 18 respectively and it can be seen that the vehicle is inherently unstable and requires a control system to keep the vehicle stable. To observe the response to the sinusoidal thrust input the time scale for the response has been increased to 50 seconds.

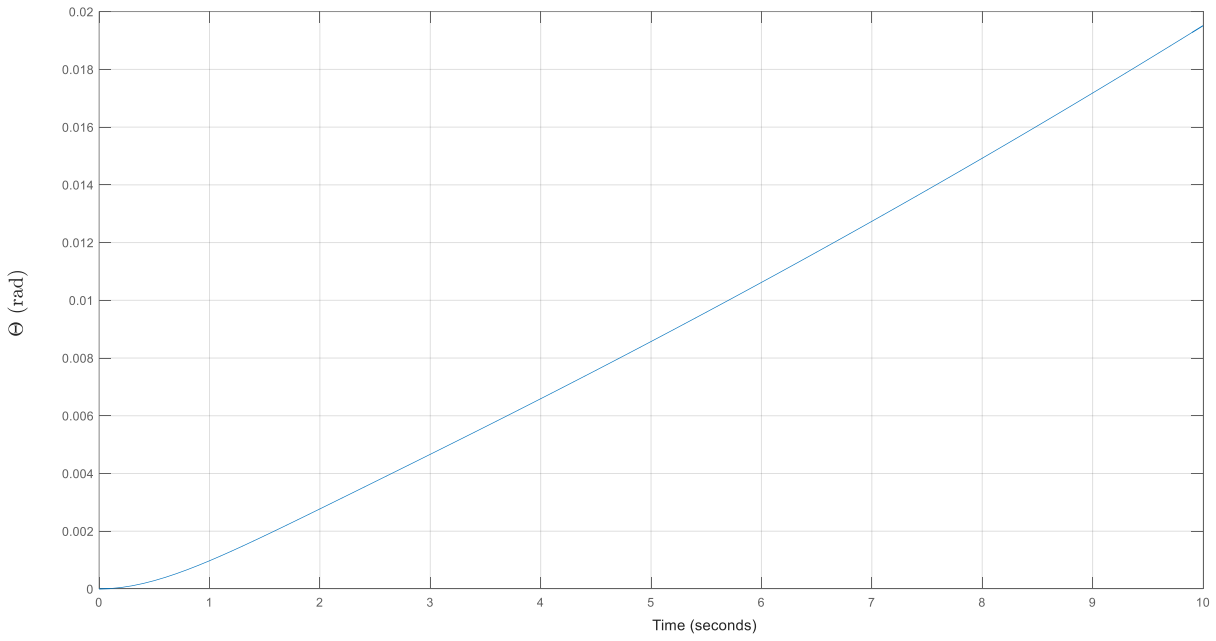


Figure 16: Vehicle response to 1N step input.

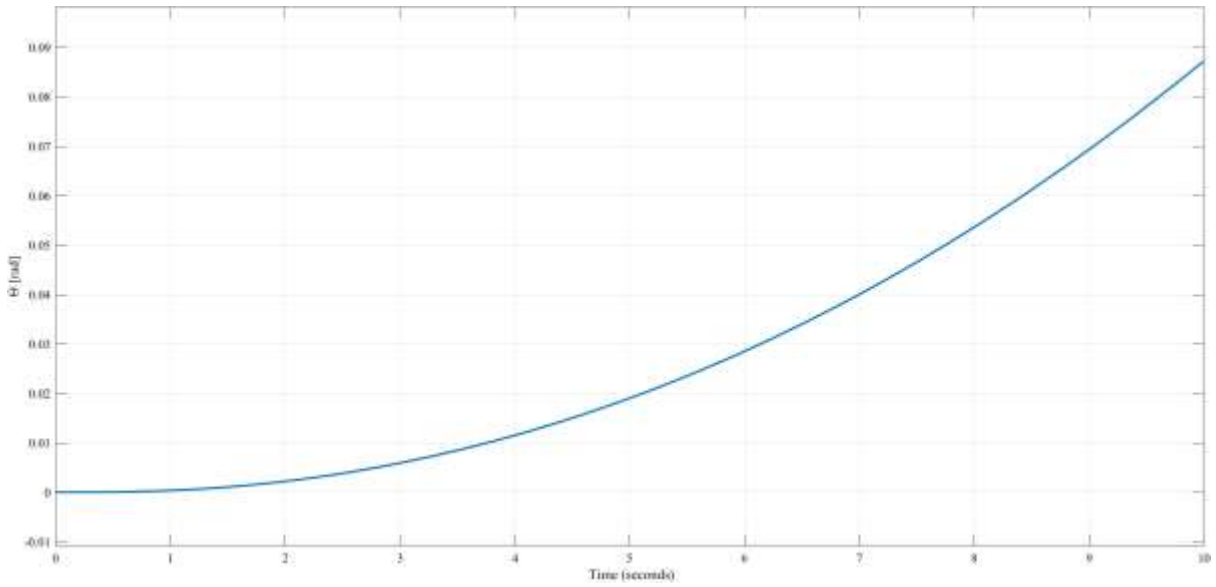


Figure 17: Vehicle response to 1N ramp input.

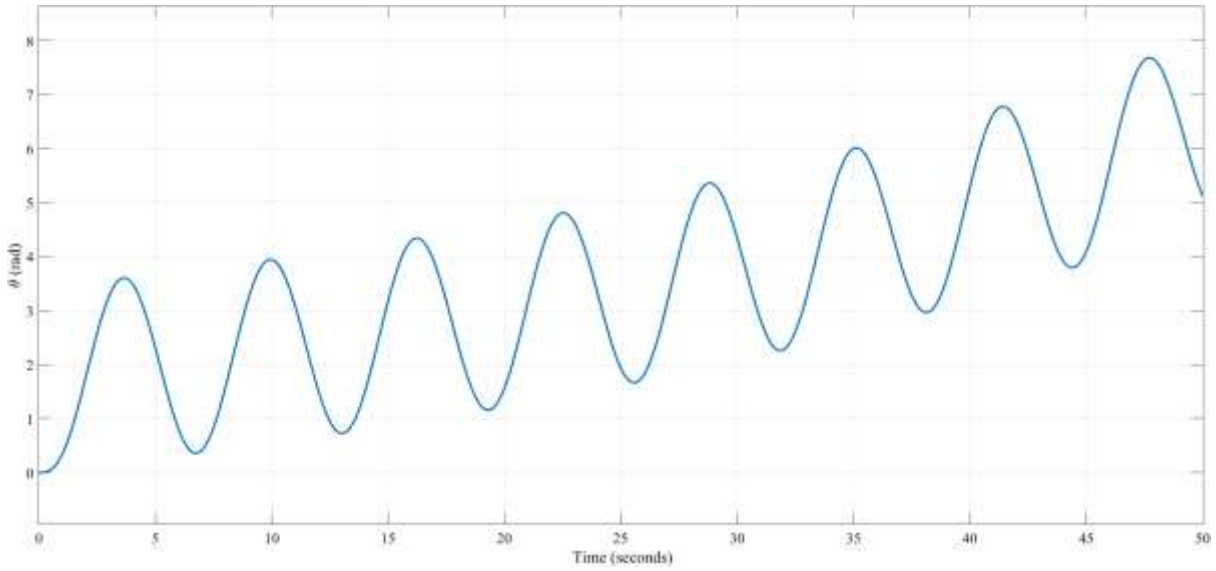


Figure 18: Vehicle response to sinusoidal input.

From Figure 19 it can be seen the bandwidth of the pitch response is only 0.029 rad/s, the phase shift between the input thrust and the output shift has a large phase shift at very low and very high frequencies with the phase shift between the output and the input reducing below 120˚ at a frequency between 0.1 and 10 rad/s the vehicle attenuates high frequency thrust inputs and there is very little gain in pitch at high frequency thrust inputs.

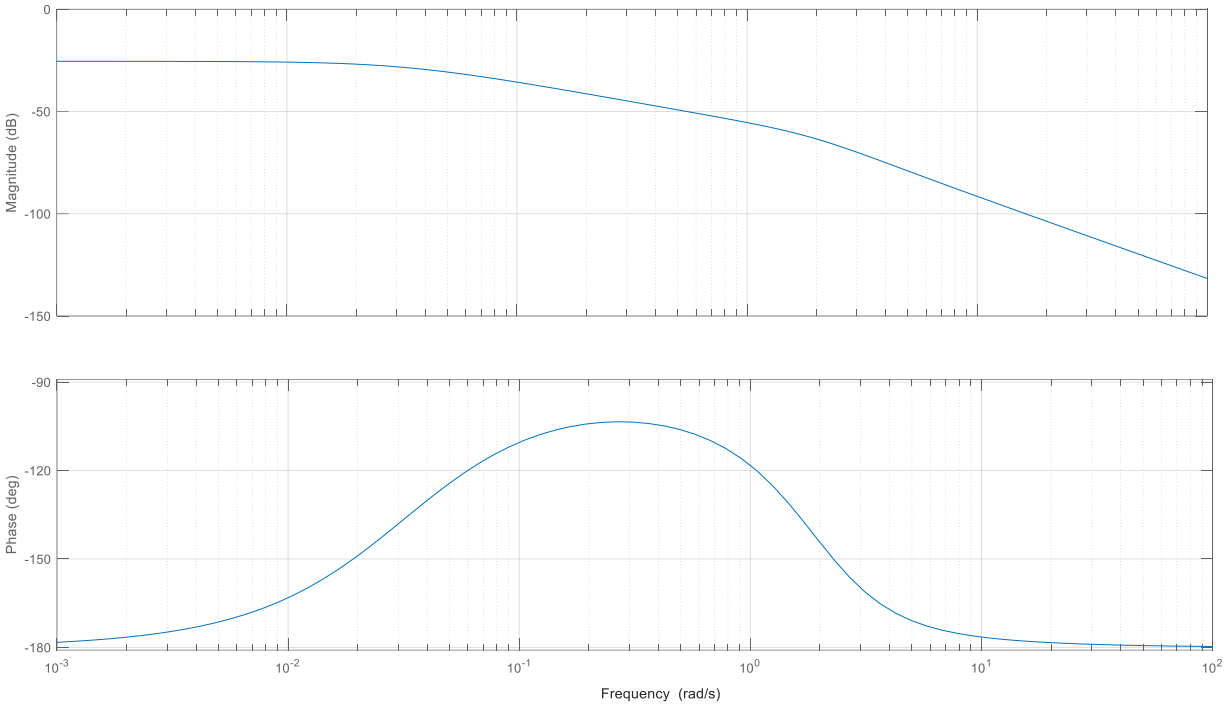


Figure 19: Bode diagram of pitch response.

The motor response to a 1V step input can be seen in Figure 20 and the motor system is asymptotically stable and settles at a rotational speed of 0.6337 rad/s.

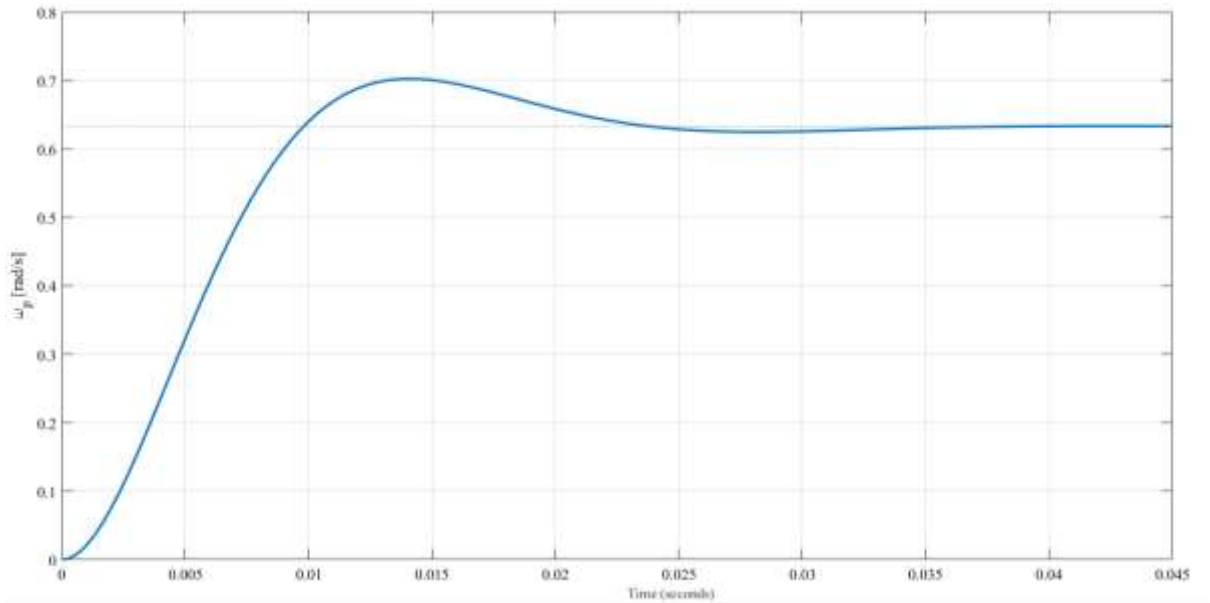


Figure 20: Motor response.

3.1 Nyquist Stability Criterion

As the motor open loop system is asymptotically stable and the polar plot does not encircle the critical point (-1,$i0$) in Figure 22, the closed loop response of voltage to motor torque will be asymptotically stable.

The vehicle pitch response to a thrust input has one right half plane root and from Figure 22 it can be seen that the pole plot does not encircle the critical point the same number of times as the number of right hand half poles of the characteristic equation and thus the closed loop system is unstable.

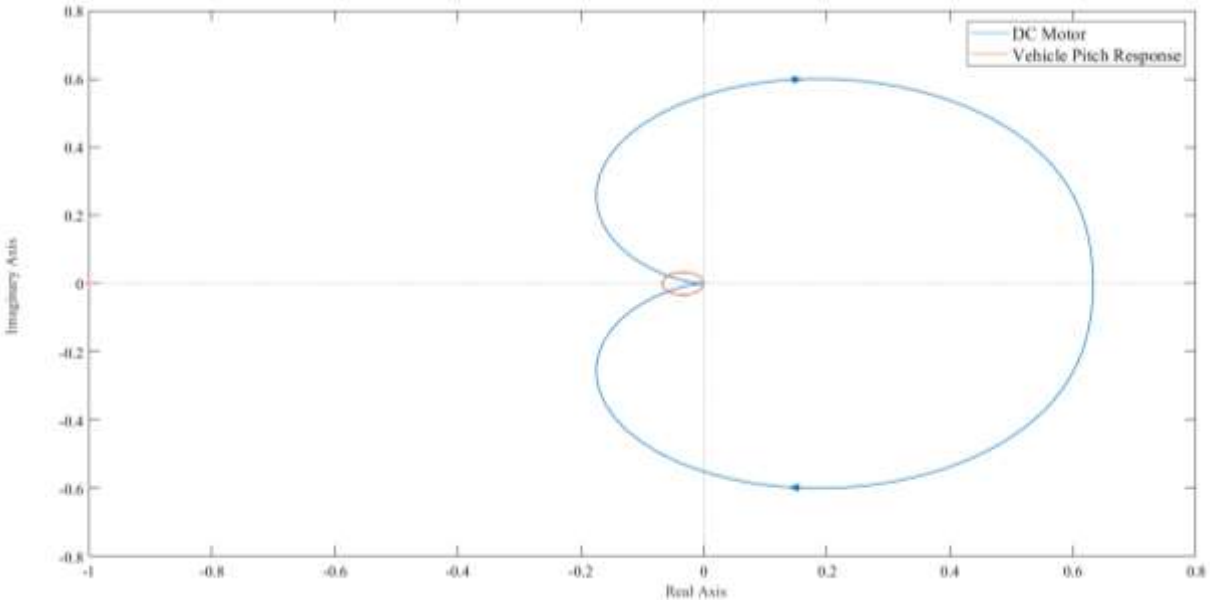


Figure 21: Nyquist Stability analysis.

4 Root Locus Control

4.1.1 Problem Set-Up

Given the transfer function as seen in (4.1) the poles are 0.0326, -2.0967, -1.9072 ± 28.0068i and -1.5608 ± 1.1787i; the zeros are -2.5367 and -2.097.

$$\frac{1.709 \times 10^{-6}s^2 + 7.919 \times 10^{-6}s + 9.091 \times 10^{-6}}{0.0001s^6 + 0.0009s^5 + 0.0818s^4 + 0.4133s^3 + 0.8067s^2 + 0.6053s - 0.0206} \quad (4.1)$$

Plotting the poles and zeros, the root locus exists on the real axis between 0.0326 and -2.0967. Using trigonometry, the angle of departure for the complex poles are found to be -96.67 and -3.18 for the complex poles using (4.2). This information is required as only two loci end at -2.097 and -2.5367, while four end at ∞ along an asymptote.

$$Angle\ of\ departure\ =\ 180\ +\sum sum\ of\ poles - \sum sum\ of\ zeros \quad (4.2)$$

To build the shape of the Root Locus, the asymptotes and their real axis intercept is required. From this, it can be inferred whether loci intersect the imaginary axis or 'break' into the real axis. To solve for the real axis intercept and asymptotes, (4.3) (4.3)and (4.4) is used, where P and Z is the number of poles and zeros respectively. Since k = 0, 1, 2…, P-Z-1, there are four asymptotes: 45°, 135°, 225° and 315° since k = 0, 1, 2 and 3.

$$\sigma = \frac{\sum s_p - \sum s_z}{P - Z} \quad (4.3)$$

$\sum s_p$ is the sum of real values of poles; $\sum s_p$ is the sum of real values of zeros.

$$\theta = \frac{2k + 1}{P - Z}180 \quad (4.4)$$

Lastly, since the asymptotes run along the real axis preventing break-ins, the loci must either intercept the imaginary axis or follow the asymptotes to -∞ along the real axis. To solve for the possible intercepts, (4.5) is used where $s = iw$ such that any real solution will be the intercepts. Solving the equation, the intercepts were 10.4686 and -10.4686 with a 4.2783×10^6 gain using the MATLAB code in Appendix B: Part 1.

$$0.0001s^6 + 0.0009s^5 + 0.0818s^4 + 0.4133s^3 + 0.8067s^2 + 0.6053s - 0.0206 + K(1.709 \times 10^{-6}s^2 + 7.919 \times 10^{-6}s + 9.091 \times 10^{-6}) = 0 \quad (4.5)$$

From this information, it can be determined that there are no break-ins or breakaways. The final analytical root locus can be seen in Figure 22.

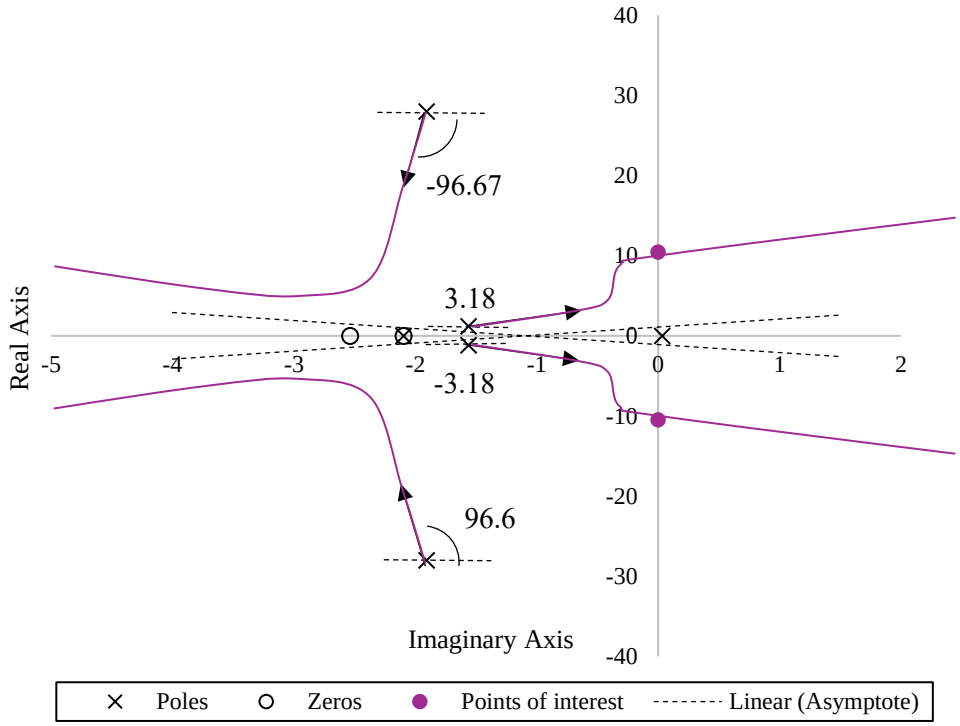


Figure 22: Analytical Root Locus.

Observing Figure 22, the system is unstable for $K > 4.2783 \times 10^6$ and not all roots are real and negative but rather those where the gain is: $9.3767 \times 10^6 > K > 4.2783 \times 10^6$. It should be noted that these K values are significantly large. To determine if the analytical understanding and solution are accurate, the *rlocus* MATLAB function was used to generate Figure 23. Comparing the two figures, the loci and roots are similar and if scaled could be accurate to each other.

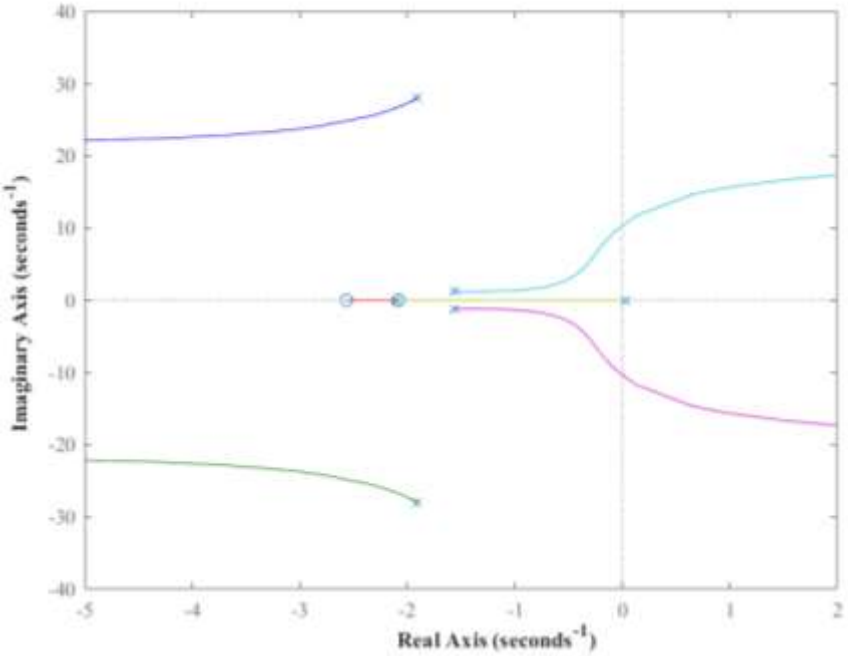


Figure 23: MATLAB Root Locus.

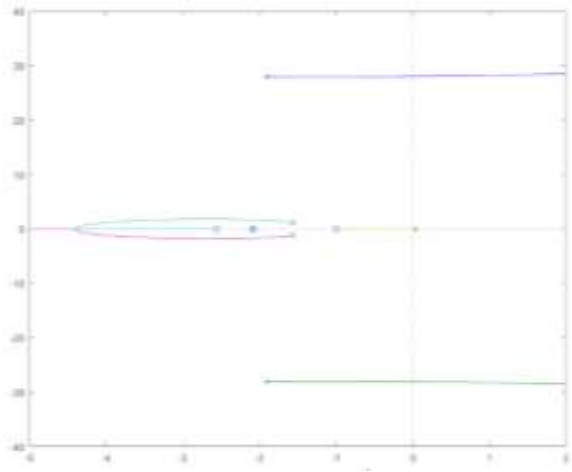
Now that the root locus for the uncontrolled system has been obtained, a control must be added using the root locus method such that the system is stable. To do this, the loci increasing towards ∞ along the real axis needs to be flipped ~180° such that all loci lie on the right half plane and ideally do not travel to -∞. Root locus is the process of adding zeros to the system to increase break-ins such that loci terminating at ∞, terminate on the real axis instead. However, there are limitations imposed by MATLAB and analytical approaches:

1. $P - Z - 1 \geq 0$ as it is impossible to have a -1 asymptote.
2. The power of the numerator < power of the denominator as Simulink does not accept transfer functions of equal or greater than powers for $\frac{Z(s)}{P(s)}$.
3. The gain of the system must lie on a dampening ratio line but cannot be critically damped as that would make the system marginally stable. However, to reduce oscillations, the dampening ratio should be high. Therefore, $0.7 < \delta < 0.9$ such that the lowest gain and lightest damper is required [3].

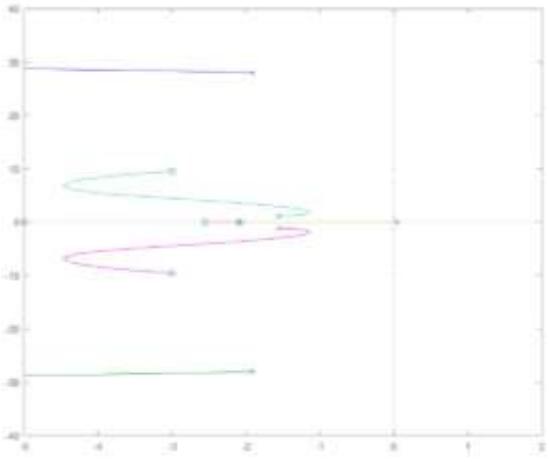
From these rules, it can be ascertained that a maximum of three zeros can be added.

4.1.2 MATLAB Investigation

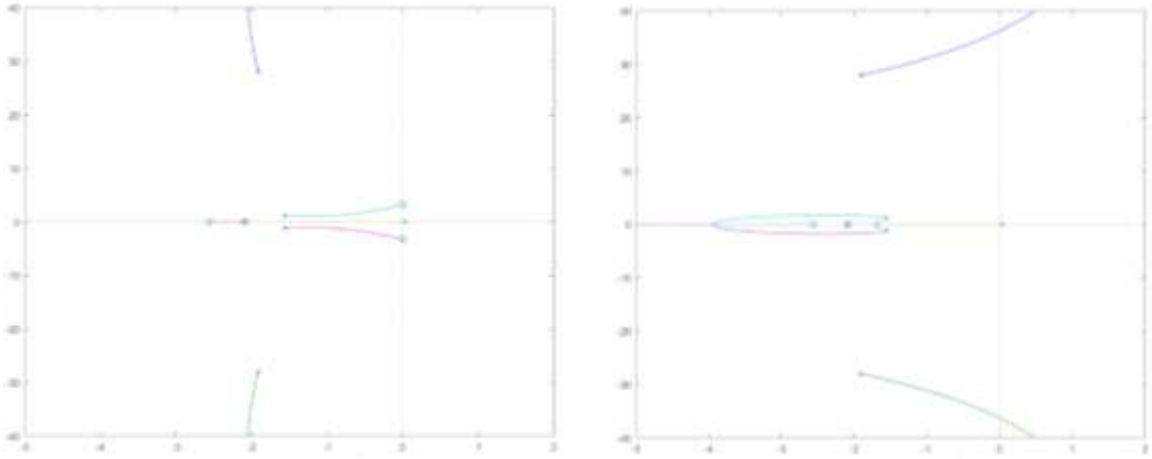
To test how locations of zeros affect the root locus, a set of test transfer functions were multiplied into the system as see in Figure 24. If we look at a single zero in 16a, the complex poles has flipped but the other complex pole pair now move into the unstable regaion. Then observing 16c and 16d, having two zeros allows for all loci to remain in the stable region, but they do not achieve the ideal stability and one pair follow the imaginary axis to ∞. By adding a third pole, as seen in 16b, all loci are on the left-plane and in the appropriate directions. Therfeore, three poles are required where the right combination should result in a stable system. To simplify the problem, it was assumed that there was one pole on the real axis and a pair of complex poles on the left-half plane.



a.) $s + 1$



b.) $s^3 + 11s^2 + 130s + 150$



c.) $s^2 + 10.6$ d.) $s^2 + 10s + 14$

Figure 24: Effects of Poles on Root Locus.

Testing varying values of d, e and f in Appendix B: Part 2 using (4.6) the poles that stabilised the system were -4, -1.5 and -0.5 where f = 4, e = 8 and d =6. The stabilised system can be seen in Figure 25 and Figure 26.

$$(s^2 + bs + c)(s + f) \quad (4.6)$$

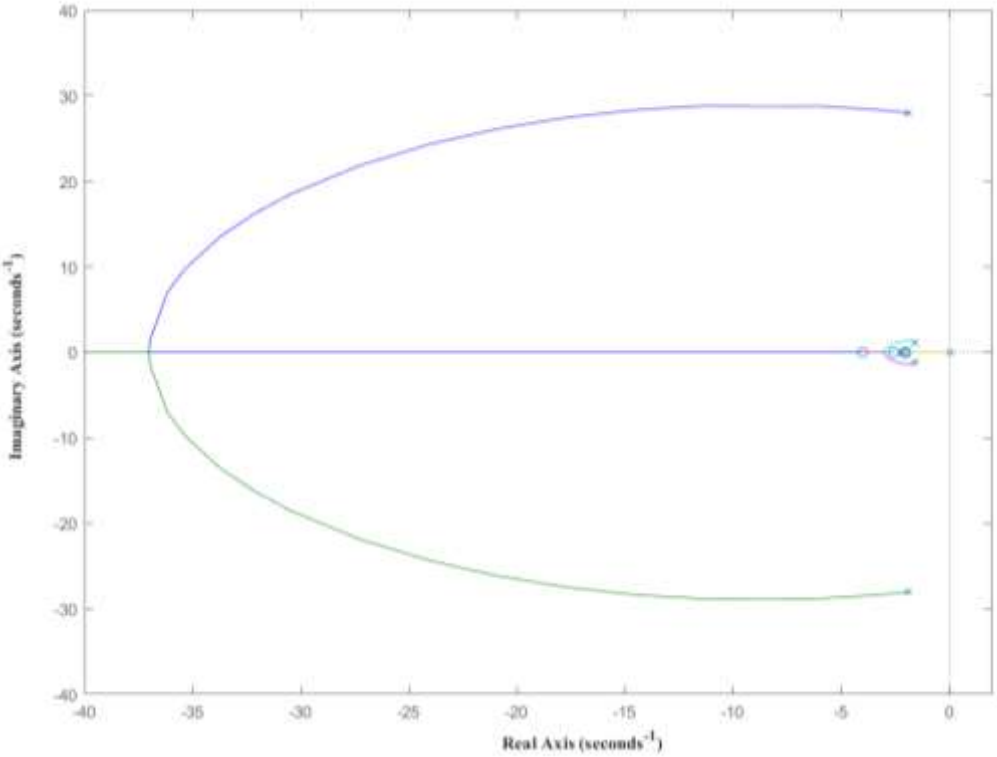


Figure 25: Stabilised Root Locus.

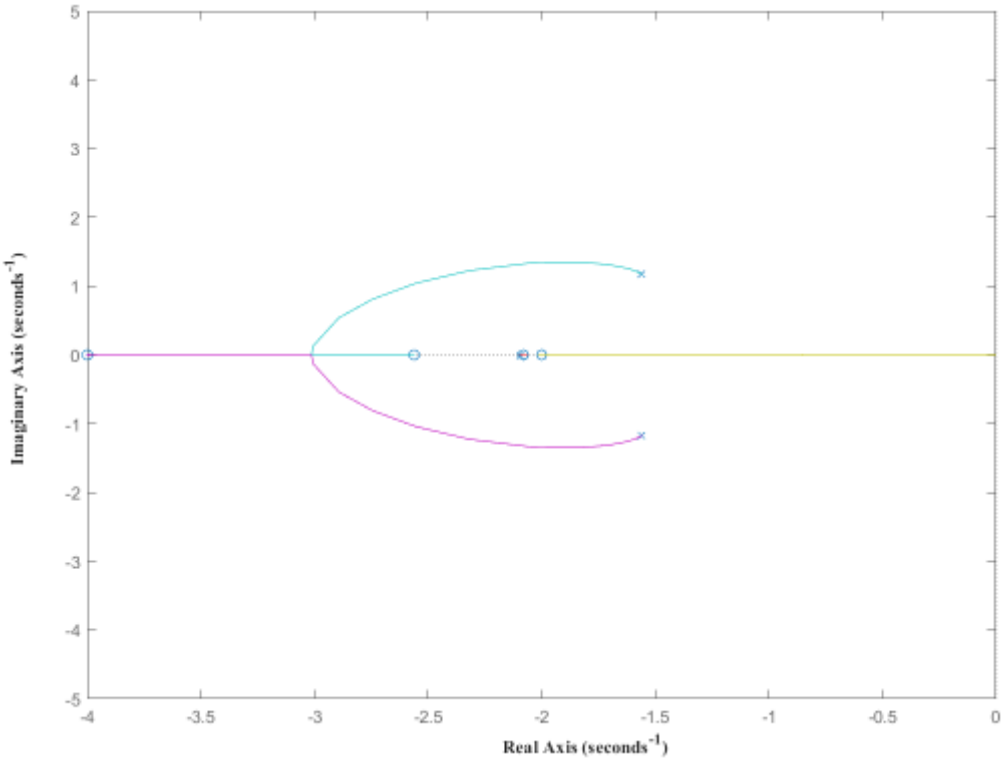


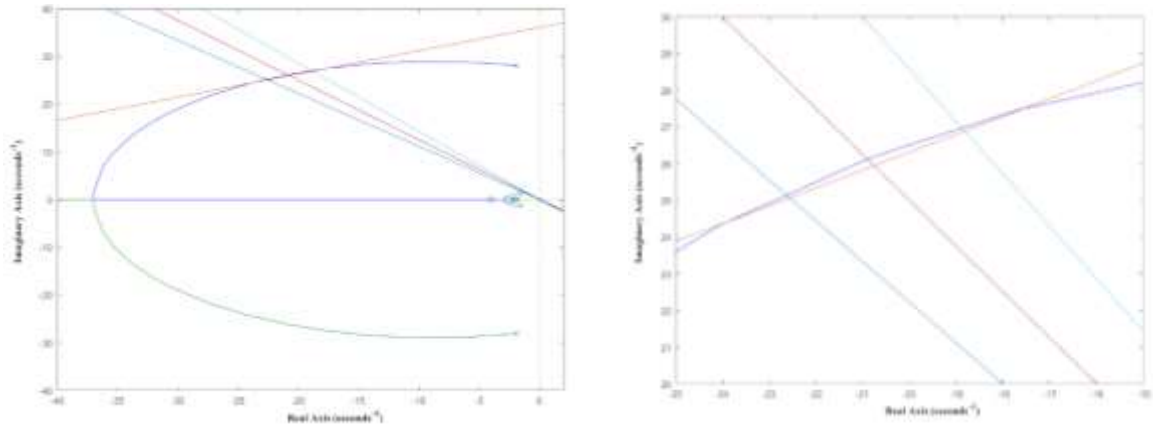
Figure 26: Close-up of Stable Root Locus.

Using this information, a new transfer function system must be calculated as seen in (4.7) because Simulink does not allow zeros to be added as blocks in accordance with limitation 2.

$$\frac{1.709\times10^{-6}s^5+2.498\times10^{-5}s^4+1.429\times10^{-4}s^3+3.989\times10^{-4}s^2+5.443\times10^{-4}s+2.909\times10^{-4}}{0.0001s^6+0.0009s^5+0.0818s^4+0.4133s^3+0.8067s^2+0.6053s-0.0206} \quad (4.7)$$

Before the model can be transferred to Simulink, a suitable gain must be input into the system. A gain is chosen based off the intersection between the root locus and the dampening line described in limitation 3. To find the s value, three lines of 0.7, 0.8 and 0.9 were plotted in MATLAB where the root locus was approximated as a straight line represented by (4.8) to find the intersection as seen in Figure 27.

$$y = 0.4871x + 36.057 \quad (4.8)$$



Full Plot Close up

Figure 27: Finding intersections of dampening and root locus.

Solving for the gain, 2483.4513 > K > 2035.2533 by substituting the s values into (4.7).

4.1.3 Controller

To determine which gain is more appropriate for the system, a Simulink controller was built as seen in Figure 28 where Theta 1 and 2 has 2483.4513 and 2035.2533 respectively [4].

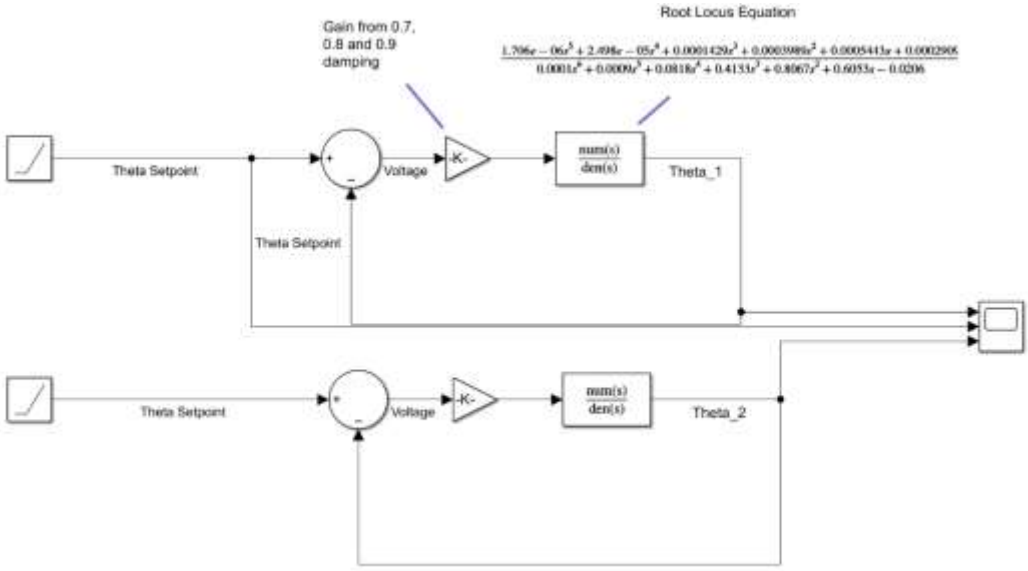


Figure 28: Root Locus Simulink Model.

To determine which damping ratio, and therefore gain, is the most appropriate for the model, a step, ramp, and sinusoidal signal was input into the model using the new zeroes and two gain options. Observing Figure 29, Figure 30,and Figure 31, the model is stable and does not exponentially increase as in the uncontrolled plant. However, it does have a steady-state error because the root locus technique does not include a correcting component from which negative feedback can be applied as seen in the Simulink model.

However, when comparing Theta 1 and Theta 2 using time domain performance specifications as seen in Table 4, Theta 1 achieves a lower steady state error and reaches that point quicker than Theta 2. Therefore, 0.9 dampening would be the most ideal case and if required, 0.7 dampening is a good enough substitute.

It should be noted that the steady-state error could be a result of the combined transfer function error found in the uncontrolled system. Because of limitation 2, this consideration is speculative as it cannot be reproduced in Simulink.

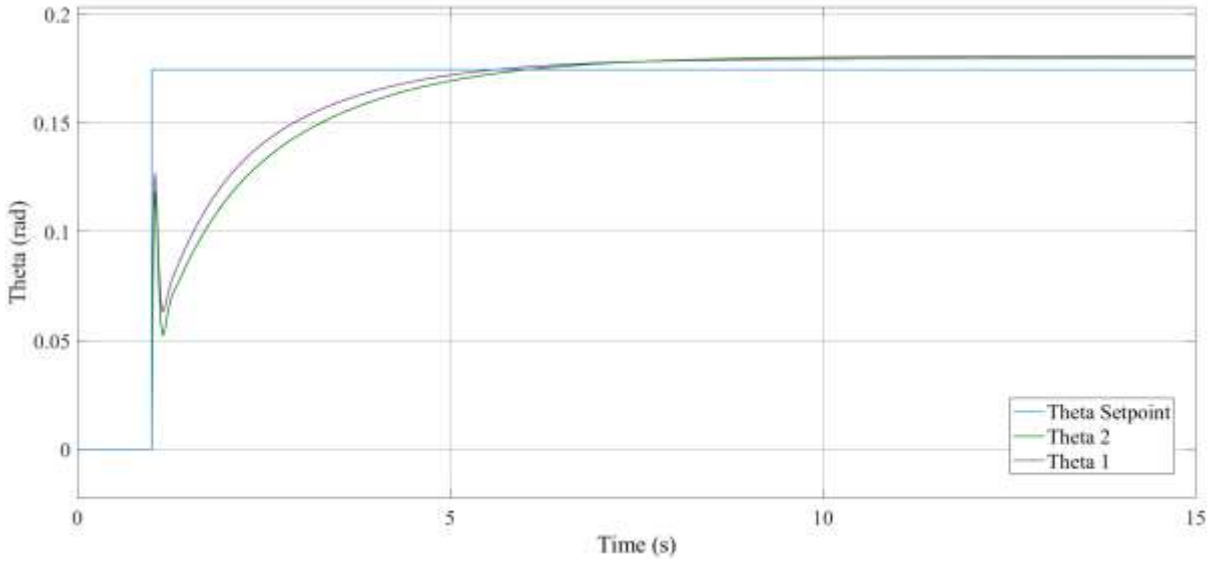


Figure 29: Root Locus Step Output.

Table 4: Gain Step Function Comparison.

	Theta 1	Theta 2
Maximum Overshoot y(t_p) [rad]	0.1794	0.1807
Maximum Overshoot [%]	10.2789	10.3533
Peak Time [s]	9.965	12.608
Delay Time [s]	1.003	1.004
Rise Time [s]	3.32	3.37

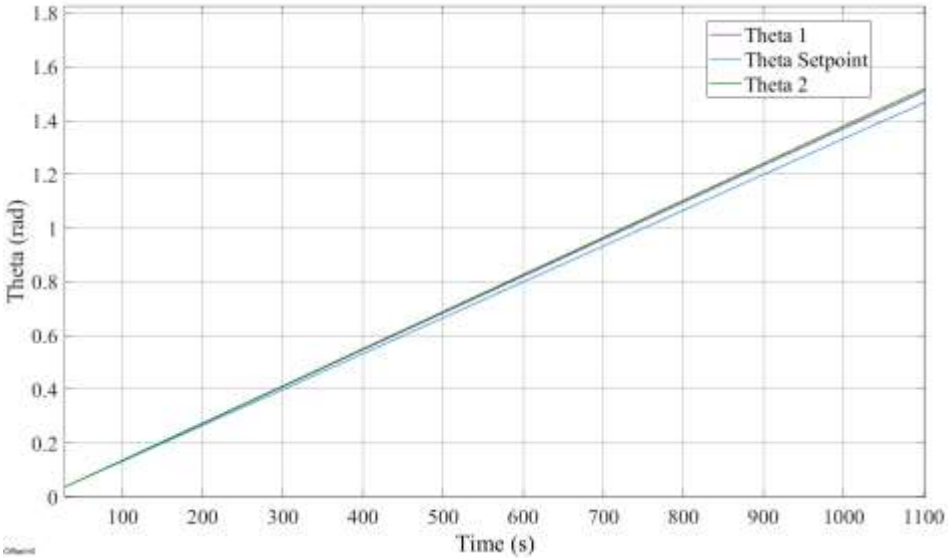


Figure 30: Root Locus Ramp Output.

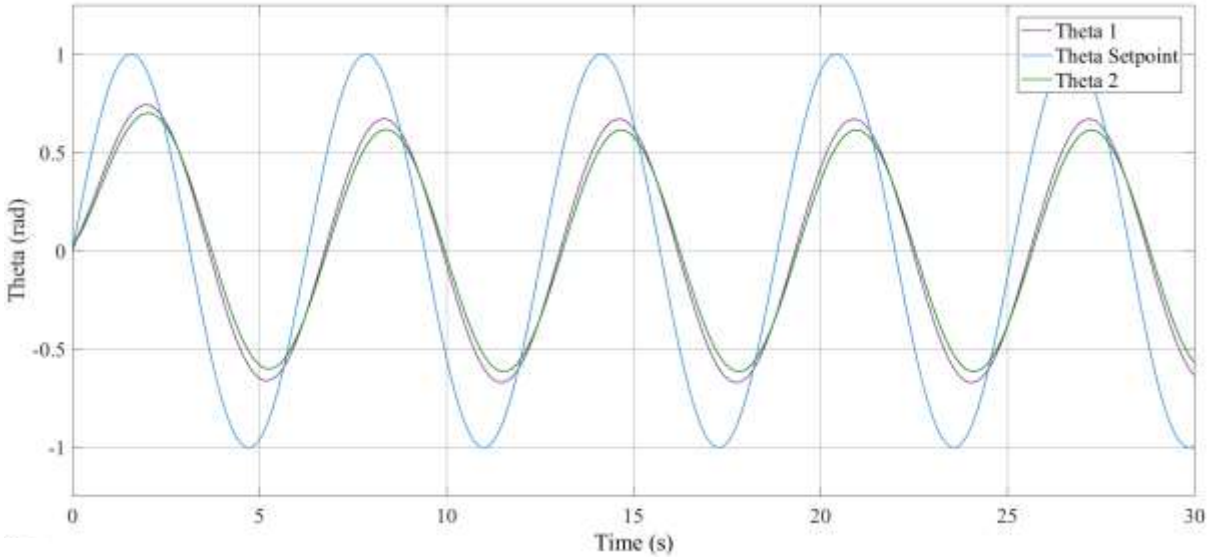


Figure 31: Root Locus Sinusoidal Output.

5 PID control

5.1 Controller Design

A feedback control system was deemed necessary to obtain the stability required for the drone to be operational in the set mission profile. In its uncontrolled state any input would result in instability with the measured results diverging. The controller was designed to manipulate the pitch of the drone at any given moment using a step input and the error. The step input would define a specific theta value with reference to the earth fixed reference frame that the drone will pitch towards. The error is defined as the difference between the input theta value and the measured. Once the desired theta value is reached the drone will return to its original steady state with 0 pitch. Figure 32 displays the Simulink model used to determine the best controller between Proportional (P), Proportional-Integral (PI), Proportional-Derivative (PD) and Proportional-Integral-Derivative (PID). A step input of 0.175 radians (10 degrees) was inputted into the various controllers and their responses measured and critically assessed to determine the best controller.

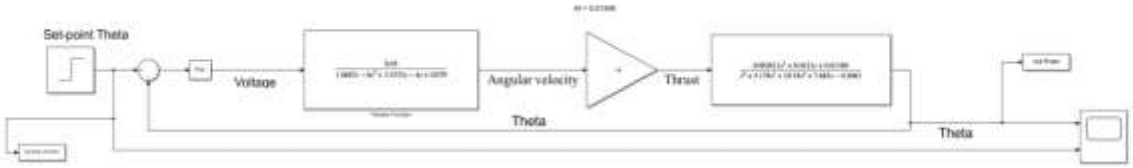


Figure 32: Simulink Controller Design.

5.1.1 Proportional (P) Controller

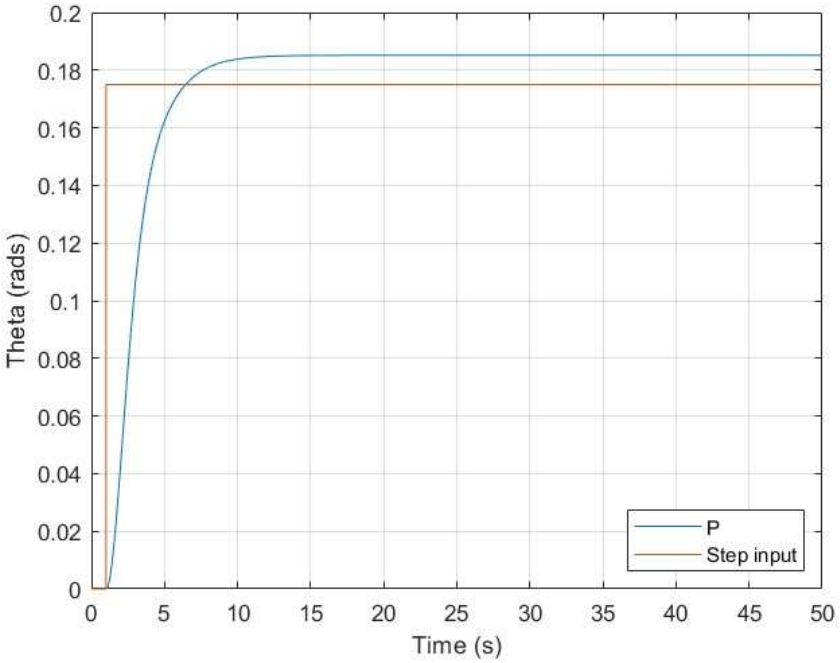


Figure 33: Proportional Control.

The proportional controller maintains a steady state error of 5.8% and rises to its maximum value of 0.187 within 10 seconds of the step input. Due to its nature the error cannot be eliminated. However, it does not oscillate and displays critically damped behaviour. The system gain value was tuned using Simulink Control design and it has a gain value of:

- $K_P = 32685.39$

5.1.2 Proportional Integral (PI) Controller

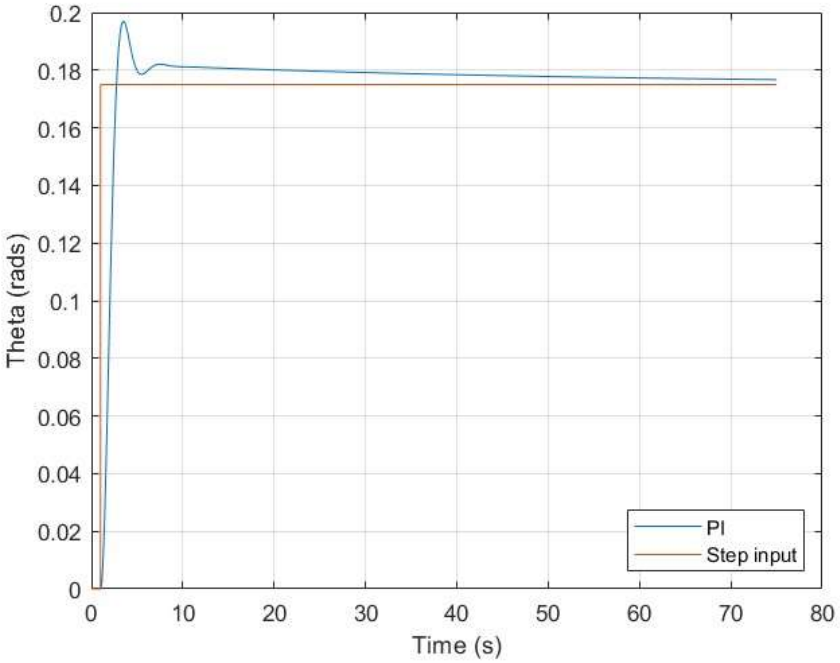


Figure 34: Proportional-Integral Control.

The PI controller has a rise time less than 5 seconds however, it takes more than 75 seconds for the steady state error to reduce to 0. Effectively the error is negligible after 20 seconds with a percentage difference of less than 5%. Compared to the P controller its error can be reduced to zero, but the time taken for that is exceptionally high. The response does oscillate with a maximum overshoot of 14.3%. Its main benefit however is how quickly it rises achieving its peak value in under 5 seconds. Its gain values were tune using the Simulink Control design and has gain values of:

- $K_P = 78327.41$
- $K_I = 1499.66$

5.1.3 Proportional Derivative (PD) Controller

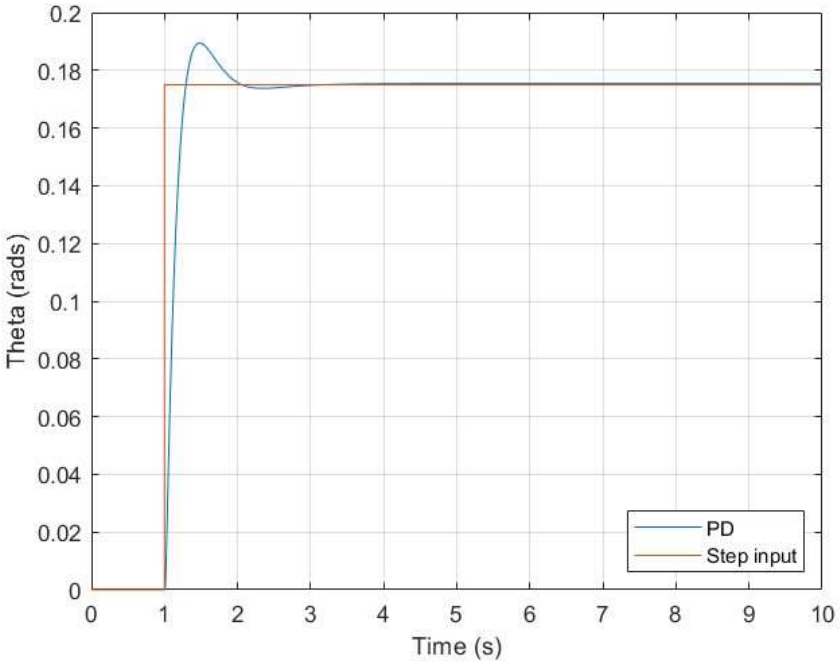


Figure 35: Proportional-Derivative Control.

The PD controller has a rise time of less than 2 seconds and obtains a steady state error of 0 within approximately 3 seconds. Its maximum overshoot is 5.71% with very little oscillation before reaching steady state. Its gain values were tune using the Simulink Control design and has gain values of:

- $K_P = 743420$
- $K_D = 299446$
- Filter coefficient = 83.50

5.1.4 Proportional Derivative Integral Controller

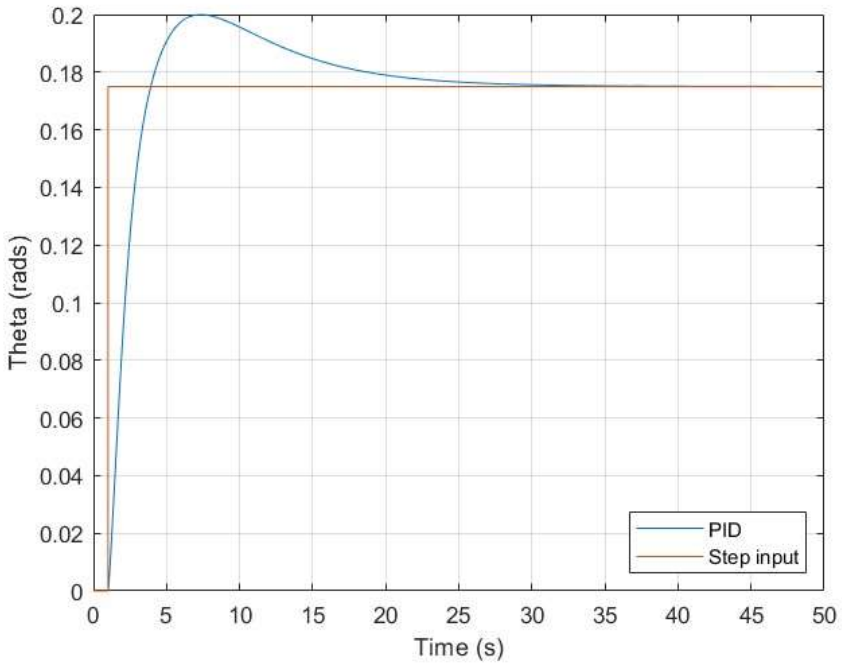


Figure 36: Proportional-Integral-Derivative Control.

The PID controller implements all error reducing aspects. It has a rise of time of roughly 7 seconds with a settling time around 30 seconds where the steady state error reduces to 0. It has a maximum overshoot of 15%. Its gain values were tune using the Simulink Control design and has gain values of:

- K_P = 47844.75
- K_I = 5791.18
- K_D = 17975.59
- Filter coefficient = 79.23

5.2 Controller Selection

The PD controller exhibits the best behaviour having the smallest maximum overshoot percentage error, rise time and settling time. The controller is robust and responds quickly to changes in comparison the other controllers. The P controller maintains a steady state error that it cannot reduce and the PI and PID controllers take a very long time in comparison to the PD controller to reduce its error to 0. Therefore, the PD controller is the best controller option to be further implemented.

5.3 Controller performance simulation evaluation

In response to a linearly increasing theta value from 0 to 90 degrees over 200 minutes the PD controller proves competent to maintain the corresponding angle and not deviate or diverge drastically to constant input changes. The graph linearly increases showing no signs of deviating.

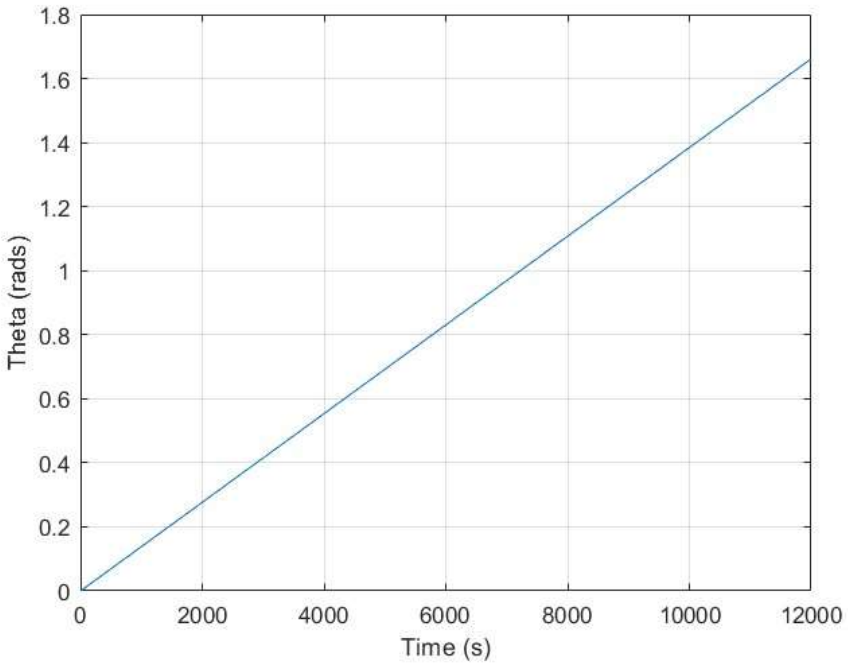


Figure 37: PD Controller for continuous input change.

5.4 Hardware

Since PID and Root Locus techniques are fundamentally the same system, the require the same hardware [4]. To measure pitch and depth, an IMU and radar are required as the sensors of the system which will act as negative feedback. The processing unit required is a microcontroller that will process the signals and feedback to return the system to its setpoint.

The motors will be underwater thrusters of which there can be up to four [5]. The thrusters will require an encoder to isolate the high voltage from the rest of the components and a motor driver [6]. The general system can be seen in Figure 38.

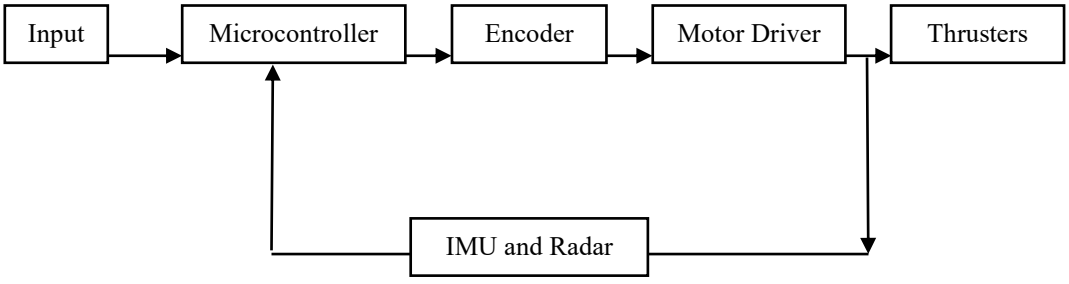


Figure 38: Hardware of Controller.

6 Controller Discussion

For all the investigated PID controllers the gain values were extremely high which can result in the plant having an uncontrollable reaction if a strong enough disturbance occurs. The gain values were a result of the step input, attempting to achieve a 10-degree pitch change in 1 second. If the step input were delayed or the set value decreased the gain values would substantially decrease and thus the potential of an uncontrollable reaction would decrease too. Of the PID controllers, the PD controller was picked as it had the best system response.

Using the same inputs, the Root Locus yields results less accurate results and with a lower response time as seen in Table 5, but the system requires a significantly lower gain. With the addition of a sensor to provide feedback as well as the summing block on Simulink, the root locus technique is a better alternative for the control system. However, this depends on the quality of the sensors. Since the mission profile requires an accurate movement of the drone to capture images, PD is the better system despite the large gain. Since disturbance response was not part of the testing of the controller, it is recommended for future testing to include it to make a better decision on the controller.

Table 5: PD and Root Locus Comparison

	PD	Root Locus
Gain	$K_P = 743420$ $K_D = 299446$	2483.45
Maximum overshoot [%]	5.71	10.2789
Rise time [s]	2	3.32
Steady-state error [rad]	0	4.867×10^{-3}

Conclusion

The plant was found to be inherently unstable; the motor and propeller assembly is dynamically stable and settles to a steady-state rotational speed, the vehicle dynamics are inherently unstable due to the lack of a tail plane to provide a restoring moment to the vehicle. The vehicle and motor plant were found to need a controller for stability control. A PD controller was chosen for its better time domain performance criteria and lower error for better underwater imaging. It is recommended that a disturbance analysis is performed to determine if the gain makes the system unstable.

References

[1] T. Siyamachira and L. Kunzwa, “DESIGN, MANUFACTURE AND TESTING OF AN UNDERWATER REMOTELY OPERATED VEHICLE. Manufacture And Testing Of A Submersible Remotely Operated Vehicle,” *www.academia.edu*, Accessed: May 20, 2024. [Online].Available: https://www.academia.edu/27069525/DESIGN_MANUFACTURE_AND_TESTING_OF_AN_UNDERWATER_REMOTELY_OPERATED_VEHICLE_Manufacture_And_Testing_Of_A_Submersible_Remotely_Operated_Vehicle

[2] M. V. Cook, Flight dynamics principles : a linear systems approach to aircraft stability and control. Amsterdam ; Boston: Butterworth-Heinemann, 2013.

[3] B. Douglas, “The Root Locus Method - Introduction,” 2 February 2013. [Online]. Available: https://www.youtube.com/watch?v=CRvVDoQJjYI.

[4] S. Kuruppu, “Project Help Video 03 - Controller Design with Root Locus Approac,” YouTube, 3 April 2023. [Online]. Available: https://www.youtube.com/watch?v=mVxyBQ_-yLI. [Accessed 19 May 2024].

[5] T-Motor, “30 4-6S Underwater Thruster for ROVs/AUVs/ Boats Specially,” [Online]. Available: https://store.tmotor.com/product/w30-rov-dynamics.html. [Accessed 19 May 2024].

[6] T. U. Haq, “Speed Control of DC Motor Using PID Algorithm (STM32F4),” 2019. [Online]. Available: https://www.instructables.com/Speed-Control-of-DC-Motor-Using-PID-Algorithm-STM3/. [Accessed 19 May 2024].

Appendix A: MATLAB Code

Vehicle Pitch transfer function

```
% ROV Data
m = 10;              % Mass [kg]
Iyy = 0.766;
S = 0.15;          % reference area [m]
MAC = 0.5;         % Mean aerodynamic chord [m]
zb = 0.25;
zg = 0.2;
Cd0 = 0.07;
Cl = 0.015;
Cl_alpha = 0.0995;  % dCl/dalpha [rad^-1]
Cd_alpha = -0.0824; % dCd/dalpha [rad^-1]
Cm_alpha = 0.1592;  % dCm/dalpha [rad^-1]
Cm_q = -0.1455;     % dCm/dq [s.m^-1]
Fb = 10*9.81;


% Operating conditions
g = 9.81;            % Gravitational acceleration
rho = 997.99;        % Density [kg.m^-3]
V0 = 2;              % Steady state speed [m.s^-1]
alpha = 0;
theta = 0;


% Dimensional stability derivatives
Xu = -2*Cd0*0.5*rho*V0*S
Xw = (Cl-Cd_alpha)*0.5*rho*V0*S
Xq = 0 % no tail
Xq_dot = 0 % no tail
Xw_dot = -3.563893


Zu = -2*Cl*0.5*V0*S
Zw = (-Cd0-Cl_alpha)*0.5*rho*V0*S
Zq = 0 % no tail
Zq_dot = 0 % no tail
Zw_dot = 0 % no tail


Mu = 0 % no tail
Mw = Cm_alpha*(0.5*rho*V0*S*MAC)
Mq = Cm_q*0.5*rho*V0*S*MAC^2 % would be zero, but we do have pitch damping
from the body
Mw_dot = 0 % No tail an no downwash lag


% Equations of motion
M = [m -Xw_dot 0 0;
```

```
    0 m-Zw_dot 0 0;
    0   -Mw_dot  Iyy 0;
    0 0 0 1];
A_prime = [Xu  Xw  Xq  -m*g;
     Zu Zw Zq+m*V0 0;
     -Mu -Mw Mq Fb*(zb-zg);
     0 0 1 0];
B_prime = [1; 0; 0.02; 0]
A = m^-1*A_prime
B = M^-1*B_prime
C = [eye(4)];
D = 0;
% Linear time invariant system
ltisys = ss(A,B,C,D)


% Transfer function for u,w,q disturbance
sys = tf(ltisys)
poles = pole(sys)
stepplot(sys(4,1),10)
xlabel("Time","FontSize",12,"FontName","Times New Romans"), ylabel("$\Theta$
(rad)", "Interpreter","latex","FontName","Times New Roman", "FontSize",12)
grid on


bode(sys(4,1))
grid on
```

Motor and vehicle stability analysis

```
% Motor Transfer Function
TF1 = tf(0.05,[1.0667e-6 3.3333e-4 0.079])


% Rotational speed to thrust
A_t = 0.01309;


% Vehicle Equations of motion assuming that sin(theta) = 0 and cos(theta) =
% 1
TF2 = tf([0.002611 0.0121 0.01389], [1 5.178 10.18 7.66 -0.2063])


% Combined transfer function
TF_comb = TF1*A_t*TF2;


stepplot(TF_comb,10)


nyquistplot(TF1)
hold on
```

```
nyquistplot(TF2)
legend("DC Motor", "Vehicle Pitch Response", fontname="Times New Roman",
fontsize=14)
hold off


stepplot(TF1)
bode(TF1)


pzmap(TF2)
```

Appendix B: Root Locus Code

Part 1: Finding gain and intercepts for uncontrolled system.

```
p = [0.0001 0.0009 0.0818 0.4133 0.8067 0.6053 -0.0206];
z = [1.709e-6 7.919e-6 9.091e-6];


a=-0.0001;
b=0.0009;
c=0.0818;
d=-0.4133;
e=-0.8067;
f=0.6053;
g=-0.0206;
h=-1.709e-6;
j=7.919e-6;
k=9.091e-6;


one=a-((h*b)/j);
two=c-(((k*b)+(h*d))/j);
three=e-(((k*d)+(f*h))/j);
four=g-(f*k/j);
equation= [one 0 two 0 three 0 four];
eqution=roots(equation)
w=-10.4686
K=(1/-j)*((b*(w^4))+(d*(w^2))+(f))
```

Part 2: Root Locus Plots

```
%Original Root Locus
s = tf('s');
GH = ((1.706e-6*s^2 + 7.919e-6*s + 9.091e-06))/((0.0001*s^6 + 0.0009*s^5 +
0.0818*s^4 + 0.4133*s^3 + 0.8067*s^2 + 0.6053*s - 0.0206))
rlocus(GH)
xlim([-5 2])
ylim([-40 40])
ylabel('Imaginary Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
xlabel('Real Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')



%Using variables to calculate new root locus with 3 new zeroes
a=1.706e-6;
b=7.919e-6;
c=9.091e-06;
d=6;
e=8;
```

```
f=4;


four=a;
three=b+(d*a);
two=c+(d*b)+(a*e);
one=(d*c)+(e*b);
zero=c*e;


b1=a;
b2=three+(f*a);
b3=two+(f*three);
b4=one+(f*two);
b5=zero+(f*one);
b6=f*zero;


% Plotting damping ratios 0.7, 0.8 and 0.9
x1=[];
y1=[];
x2=[];
y2=[];
x3=[];
y3=[];
x4=[];
y4=[];


p=0;


for n=2:-1:-60
    p=p+1;
    x1(p)=n;
    x2(p)=n;
    x3(p)=n;
    x4(p)=n;
    y1(p)=n/-0.7;
    y2(p)=n/-0.8;
    y3(p)=n/-0.9;
    y4(p)=(0.4871*n)+36.057;
end


%Combining root locus with intersecting lines
G2=((b1*s^5+b2*s^4+b3*s^3+b4*s^2+b5*s+b6))/(0.0001*s^6 + 0.0009*s^5 +
0.0818*s^4 + 0.4133*s^3 + 0.8067*s^2 + 0.6053*s - 0.0206);
rlocus(G2)
xlim([-4 0])
ylim([-5 5])
```

```
ylabel('Imaginary Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
xlabel('Real Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
title('')


G2=((b1*s^5+b2*s^4+b3*s^3+b4*s^2+b5*s+b6))/(0.0001*s^6 + 0.0009*s^5 +
0.0818*s^4 + 0.4133*s^3 + 0.8067*s^2 + 0.6053*s - 0.0206);
rlocus(G2)
xlim([-40 2])
ylim([-40 40])
ylabel('Imaginary Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
xlabel('Real Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
title('')


G2=((b1*s^5+b2*s^4+b3*s^3+b4*s^2+b5*s+b6))/(0.0001*s^6 + 0.0009*s^5 +
0.0818*s^4 + 0.4133*s^3 + 0.8067*s^2 + 0.6053*s - 0.0206);
rlocus(G2)
hold on
plot(x1,y1,x2,y2,x3,y3,x4,y4)
xlim([-25 -15])
ylim([20 30])
ylabel('Imaginary Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
xlabel('Real Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
title('')
hold off


G2=((b1*s^5+b2*s^4+b3*s^3+b4*s^2+b5*s+b6))/(0.0001*s^6 + 0.0009*s^5 +
0.0818*s^4 + 0.4133*s^3 + 0.8067*s^2 + 0.6053*s - 0.0206)
rlocus(G2)
hold on
plot(x1,y1,x2,y2,x3,y3,x4,y4)
xlim([-40 2])
ylim([-40 40])
ylabel('Imaginary Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
xlabel('Real Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
title('')
hold off


%Determining value of gain at intersection
G2_f= @(s) (0.0001*s.^6 + 0.0009*s.^5 + 0.0818*s.^4 + 0.4133*s.^3 +
0.8067*s.^2 + 0.6053*s - 0.0206)/(b1*s.^5+b2*s.^4+b3*s.^3+b4*s.^2+b5*s+b6);
```

```
s=-18.82+26.89i % The value of s changes depending on what point you use
disp(G2_f(s))


%Test examples
s = tf('s');


a=(s+1)
sys = ((1.706e-6*s^2 + 7.919e-6*s + 9.091e-06)*a)/((0.0001*s^6 + 0.0009*s^5 +
0.0818*s^4 + 0.4133*s^3 + 0.8067*s^2 + 0.6053*s - 0.0206));
rlocus(sys)
xlim([-5 2])
ylim([-40 40])
ylabel('Imaginary Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
xlabel('Real Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
title('')


a=(s^2+10.6)
sys = ((1.706e-6*s^2 + 7.919e-6*s + 9.091e-06)*a)/((0.0001*s^6 + 0.0009*s^5 +
0.0818*s^4 + 0.4133*s^3 + 0.8067*s^2 + 0.6053*s - 0.0206));
rlocus(sys)
xlim([-5 2])
ylim([-40 40])
ylabel('Imaginary Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
xlabel('Real Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
title('')


a=(s^2+10*s+14)
sys = ((1.706e-6*s^2 + 7.919e-6*s + 9.091e-06)*a)/((0.0001*s^6 + 0.0009*s^5 +
0.0818*s^4 + 0.4133*s^3 + 0.8067*s^2 + 0.6053*s - 0.0206));
rlocus(sys)
xlim([-5 2])
ylim([-40 40])
ylabel('Imaginary Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
xlabel('Real Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
title('')


a=(s^2+6*s + 100)*(s+5)
sys = ((1.706e-6*s^2 + 7.919e-6*s + 9.091e-06)*a)/((0.0001*s^6 + 0.0009*s^5 +
0.0818*s^4 + 0.4133*s^3 + 0.8067*s^2 + 0.6053*s - 0.0206));
rlocus(sys)
xlim([-5 2])
```

```
ylim([-40 40])
ylabel('Imaginary Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
xlabel('Real Axis', 'FontSize', 11, 'FontName','Times New
Roman','FontWeight','bold')
title('')
```