Imperial College London

AERO96005 - GROUP DESIGN PROJECT

Wholly Reusable Launch System (Recovery)

WRR05: Flight mechanics and stability analysis of a synchropter concept

Student: Andrew Ng
CID: 01180665

Project Supervisor: Dr. E. Levis
Team Supervisor: Dr. A. Wynn

Submission Date: 15/06/2020

Department of Aeronautics
South Kensington Campus
Imperial College London
London SW7 2AZ

Abstract

A state space model of the synchropter was designed with inputs from the aerodynamics and structures teams. It takes into account rigid body dynamics and slungload dynamics. An iterative solver calculation of trim point was developed and displayed reliable convergence properties.

Open loop responses of the synchropter was studied at hover and cruise with the second stage payload. Static stability analysis revealed speed stability in both flight modes.

Dynamic stability studies reveal an unstable phugoid mode at hover, with a oscillation period of 7.7s. At cruise, this phugoid mode becomes stable as the associated eigenvalue crosses the left half plane. Roll subsidence and spiral mode were identified to be unstable at hover. These modes were highlighted to the controls team for further action.

This third year Group Design Project (GDP) aims to develop a preliminary design for the mid-air recovery aspect of a wholly reusable launch system for which as much as possible can be recovered and reused for subsequent launches of LEO satellites. This report forms a subset of the marking criteria for the *AERO 96005 - Group Design Project* module, seeking to test engineering judgement and design tradeoffs within group settings.

WRR Team Members

Table 1: *Team Members and Roles.*

Name	Team Name	Team Code
Alessandro Bullitta Yassine Ghenima Cyril Gliner	Project co-ordination	WRR01
Chi Wing Cheng Alessandro Rossi Polvara Tin Hang Un Le Yin	Aerodynamics	WRR02
Anton Le Tejasva Malhotra **Atri Sharma** Alastair Wood	Structures & CAD	WRR03
Omar Arangath Thomas Cross Zuzanna Janusz **Sam Jenner** Jo Tan Tom Walker	Systems & Propulsion	WRR04
Filip Bastien Ananya Dubey Kai Hon Andrew Ng India Tavares	Flight Mechanics, Performance & Control	WRR05
Tom Alston **Hakan Serpen** Tze Leung Wong	Flight Simulation & Testing	WRR06

†Sub-team leads are shown in **bold**

Contents

List of Tables

List of Figures

Nomenclature

The following list details all symbols and conventions used in the report. Typical flight dynamics conventions as used in rotorcraft literature are used as far as possible. In the case of deviation from convention, this will be made clear to the reader in the report. Body axes coordinate system are used unless otherwise specified, following a right hand rule. Standard stability derivative notation applies, normalised with respect to the term's associated mass or inertia component.

Greek Symbols

β_{1c} Flap angle of rotor 1 along the lateral direction

β_{1s} Flap angle of rotor 1 along the longitudinal direction

β_{2c} Flap angle of rotor 2 along the lateral direction

β_{2s} Flap angle of rotor 2 along the longitudinal direction

δ_e Horizontal tailplane setting angle

δ_R Rudder deflection

λ Eigenvalue

ω_d Damped natural frequency

ω_n Undamped natural frequency

ϕ Roll angle

ϕ_L Payload cable angle about body x axis, refer to A.1

ψ Yaw angle

θ Pitch angle

θ_0 collective pitch

θ_L Payload cable angle about body y axis,refer to A.1

θ_{1c} Lateral cyclic of rotor 1

θ_{1s} Longitudinal cyclic of rotor 1

θ_{2c} Lateral cyclic of rotor 2

θ_{2s} Longitudinal cyclic of rotor 2

ξ Damping ratio

Roman Symbols

$\boldsymbol{Q}$ Normalised Eigenvector matrix

$\boldsymbol{X}$ Eigenvector matrix

L Resultant moment along the body x axis

M	Resultant moment along the body y axis
M_a	Mass of entire aircraft
N	Resultant moment along the body z axis
p	Angular velocity about the body x axes
q	Angular velocity about the body x axes
r	Angular velocity about the body z axes
T	Period of oscillation
t_2	Time to double amplitude
$t_{1/2}$	Time to half amplitude
u	Velocity along the body x axes
v	Velocity along the body y axes
w	Velocity along the body z axes
X	Resultant force along the body x axis
Y	Resultant force along the body y axis
Z	Resultant force along the body z axis

Part I
Task overview

The design of an unmanned synchropter mandates a keen understanding of rotorcraft and working from first principles. There is extensive literature on conventional rotorcraft dynamics; most texts are written with a conventional helicopter with a main rotor and tail rotor in mind [3], Simulink models can also be found online [4].
The same cannot be said for synchropter designs. Consequently, generating and verifying a state space model of the synchropter is crucial, before any significant controls and stability analyses can be undertaken.
Additionally, our synchropter is unmanned, with remote piloting capability in key operating stages. More detail about this can be found in [5] and [6].

Part II
Mission breakdown

In the conceptual design phase of this project, 3 main configurations of helicopters were shortlisted: Conventional design, tandem rotors and synchropter design. After internal and external trade studies [7], the synchropter configuration was chosen due chiefly to compactness, higher cruise efficiency and good lifting performance.

1 Mission Legs

The detailed mission profile expected of our synchropter is found in [8]. In this report, we have chosen 2 mission legs to study. We will be working with the stage 2 payload, with a mass of m_L= 554kg. The simulations team conducted a preliminary flight test at hover with the stage 1 payload, of mass $m_L = 1400kg$ and found that hovering at 200ft was virtually impossible[9]. We hence focused our resources on studying the stage 2 payload instead. Hover and cruise responses in later sections of this report will be based off the values listed in table 2.

Table 2: Mission parameters

Parameter	Description	units	Hover	Cruise
L	Slung load cable length	m	36.0	36.0
u_{cruise}	forward cruise speed	ms^{-1}	1.0	30.0
m_L	Payload mass	m	554.0	554.0
h	Altitude from sea level	ft	200	3000

Part III
Model formulation

The formulation of a state space model is the first step to understanding the behaviour of our helicopter. Trim point calculations, equation of motion definition and linearisation will be covered in this part.

2 External forces and moments

The individual rotor and fuselage forces and moments are solved in [10]. Empennage contributions can be found in [11]. Payload contributions can be found in [5]. Analytical expressions will be covered in depth in the above. A detailed explanation of the derivation method can also be understood from [12]. In this report, we are interested in solving for the resultant forces and moments, written below. These equations are adapted from the method proposed in [13]. A free body diagram was independently developed [14] and cross referenced with this method, leading to the same conclusion.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_{r1} + X_{r2} + X_{fus} + X_{tp} + X_{fn} \\ Y_{r1} + Y_{r2} + Y_{fus} + Y_{tp} + Y_{fn} \\ Z_{r1} + Z_{r2} + Z_{fus} + Z_{tp} + Z_{fn} \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} L_{R1} + L_{R2} + Y_{R1}h_{R1} + Y_{R2}h_{R2} + Z_{R1}y_{R1} + Z_{R2}y_{R2} + Y_{fus}h_{fus} + L_{fus} + L_{PL} + Y_{PL}h_{PL} \\ M_{R1} + M_{R2} - X_{R1}h_{R1} - X_{R2}h_{R2} + Z_{R1}l_{R1} + Z_{R2}l_{R2} - X_{tp}h_{tp} + Z_{tp}l_{tp} - X_{fn}h_{fn} + M_{fus} + M_{PL} - X_{PL}h_{PL} + Z_{PL}l_{PL} \\ N_{R1} + N_{R2} - Y_{R1}l_{R1} - Y_{R2}l_{R2} - Y_{fn}l_{fn} - Y_{fus}l_{fus} + N_{fus} + N_{PL} - Y_{PL}l_{PL} \end{bmatrix} \tag{2}$$

Notably, for mission legs without the slung load, all contributions due to the slung load go to 0. Following the method proposed in [11], empennage contributions are negligible at hover.

3 Equations of motion

The equations of motion of any rotorcraft can be broken down into contributions from Euler equations (equations 3 to 8) , kinematic equations (equations 9 to 12) and rotor equations detailed below. Detailed derivation is beyond the scope of this report, although the method is detailed in [15] and [3].

We have deemed it necessary to include the rotor flap angles in our state space model. Without these states, the helicopter behaves as a rigid body with 6 degrees of freedom. Preliminary engine specs was operation at 650 RPM; this gives a rotor rotation frequency of 68.06 rad/ sec. From [15], the upper bound of rotation frequency for an acceptable rigid body model is 10 rad/sec. It was hence initially deemed necessary to increase the number of states by including rotor flap angles.

We have also included the equations of motion for the payload dynamics as equations 16 to 19.

$$\dot{u} = -(wq - vr) + \frac{X}{M_a} - g\ sin\ \theta \tag{3}$$

$$\dot{v} = -(ur - wp) + \frac{Y}{M_a} + g\ \cos\ \theta\ sin\ \phi \tag{4}$$

$$\dot{w} = -(vp - uq) + \frac{Z}{M_a} + g\ \cos\ \theta\ \cos\ \phi \tag{5}$$

$$\dot{p} = I_{xx}^{-1}(L + qr(I_{yy} - I_{zz})) \tag{6}$$

$$\dot{q} = I_{yy}^{-1}(M + pr(I_{zz} - I_{xx}) \tag{7}$$

$$\dot{r} = I_{zz}^{-1}(N + pq(I_{xx} - I_{yy}) \tag{8}$$

$$\dot{\phi} = p + q\ sin\ \phi\ \tan\ \theta + r \cos\ \phi\ \tan\ \theta \tag{9}$$

$$\dot{\theta} = q\ cos\ \phi - r\ sin\ \phi \tag{10}$$

$$\dot{\psi} = q\ sin\ \phi\ sec\ \theta + r\ \cos\ \phi\ sec\ \theta \tag{11}$$

$$\dot{\beta}_{1s} = f(u, v, w, p, q, r, \beta_{1s}, \beta_{1c}, \theta_0, \theta_{1s}, \theta_{1c}) \tag{12}$$

$$\dot{\beta}_{1c} = g(u, v, w, p, q, r, \beta_{1s}, \beta_{1c}, \theta_0, \theta_{1s}, \theta_{1c}) \tag{13}$$

$$\dot{\beta}_{2s} = h(u, v, w, p, q, r, \beta_{2s}, \beta_{2c}, \theta_0, \theta_{2s}, \theta_{2c}) \tag{14}$$

$$\dot{\beta}_{2c} = j(u, v, w, p, q, r, \beta_{2s}, \beta_{2c}, \theta_0, \theta_{2s}, \theta_{2c}) \tag{15}$$

$$\dot{\phi_L} = \dot{\phi_L} \tag{16}$$

$$\dot{\theta_L} = \dot{\theta_L} \tag{17}$$

$$\ddot{\phi_L} = k(u, v, w, p, q, r, \phi, \theta, \phi_{L,L}\ ) \tag{18}$$

$$\ddot{\theta_L} = m(u, v, w, p, q, r, \phi, \theta, \phi_{L,L}\ ) \tag{19}$$

Figure 1: 17 equations of motion.

Due to the symmetry of our rotorcraft about the $x - z$ plane, I_{xz} can be neglected, allowing us to to arrive at the above system of equations.

4 State Space Model Formulation

To form our A and B matrices in the state space model, we conduct Jacobian Linearisation at selected equilibrium points. A known limitation of such an approach is that behaviour at extreme flight conditions cannot be accurately modelled, such as operation at V_{ne}[16].
2 methods were proposed to implement Jacobian Linearisation. The first involves using MATLAB's diff() function while the second involves implementing a 1st order central difference scheme as proposed in [15].
A preliminary analysis for the X_u derivative at the hover equilibrium point, using a tolerance of 10^{-1}. The percentage error between both methods was found to be 0.0022%, clearly a negligible difference. Hence, we elected to use the numerical method for efficient computation.
Initially we attempted to implement the state space model with the 4 flapping equations. However, it led to erronenous results. It was likely due to mishandling of symbolic variables in MATLAB. The result of this will be discussed in 6.1 The initial 17 state vector was:

$$x = \begin{bmatrix} u & v & w & p & q & r & \phi & \theta & \psi & \beta_{1s} & \beta_{1c} & \beta_{2s} & \beta_{2c} & \phi_L & \theta_L & \dot{\phi}_L & \dot{\theta}_L \end{bmatrix}^T \tag{20}$$

Since deciding to bypass the flap equations, we have identified the state vector x and control vector u below. These will be used for analysis.

$$x = \begin{bmatrix} u & v & w & p & q & r & \phi & \theta & \psi & \phi_L & \theta_L & \dot{\phi}_L & \dot{\theta}_L \end{bmatrix}^T \tag{21}$$

$$u = \begin{bmatrix} \theta_0 & \theta_{1s} & \theta_{1c} & \theta_{2s} & \theta_{2c} & \delta_e & \delta_R \end{bmatrix}^T \tag{22}$$

We will then define all 13 of our non-linear equations of motion, equations 3 to 12 and equations 16 to 19 into the variable $f(x, u)$.

$$\dot{x} = f(x, u) \tag{23}$$

Jacobian linearisation of equation 23 yields:

$$\dot{x} = Ax + Bu \tag{24}$$

$$A = \frac{df}{dx}|_{x_e,u_e} \qquad B = \frac{df}{du}|_{x_e,u_e}$$

These are written for the most general case. For mission legs without the slung load, the final 4 states are set to 0. A few simplifications were applied to the modelling process. These assumptions are:

1. CG position does not change with payload. Only the mass of the system changes. This assumption is made to simplify the model in the interests of time, although this is expected to have an impact on the various moment arms and the values of I_{xx}, I_{yy} and I_{zz} would likely have been undervalued.

2. Assume that azimuthal variations, do not give rise to changes in cyclic and collective trim control angle, especially in high speed forward flight. This phenomenon is chiefly due to compressibility effects. [3]

3. Empennage effects are negligible in hover, but must be considered at forward cruise.

A schematic of the A matrix is provided in A.3.

5 Trim point calculations

At the trim point, it is known that $\dot{x} = 0$. Equation 23 is reduced to

$$f(x, u) = 0 \tag{25}$$

There exists multiple approaches to computing a trim point. One approach is to express the equations of motion in terms of the design parameters and associated terms, compile them and eliminating variables as far as possible through direct substitution. An example of eliminating all variables associated to the main rotor is seen in [17]. Another method is to express all parameters in terms of the states and control inputs. It is the latter method that we adopt. The following method for trim point evaluation is based off that proposed in [18]. In all mission legs, we have 13 equations of motion, 13 states and 7 control inputs, for a total of 20 variables.
To describe any mission leg, we are able to start with 12 known variables. These are detailed in table 3.

Substituting these known variables into the 13 equations of motion, we are then left with 8 unknowns. We select 8 equations of motion which are functions of these 8 variables. Eqn 25 is solved using the fsolve() function in MATLAB, which utilises the Levenberg-Marquardt algorithm[19]. Consequently, our trim point is obtained.

Table 3: Initial known variables

Initial known variables	Units	Hover with Payload	Cruise with Payload
u_e	ms^{-1}	10^{-3}	30
v_e	ms^{-1}	10^{-3}	10^{-3}
w_e	ms^{-1}	10^{-3}	10^{-3}
p_e	$rads^{-1}$	10^{-3}	10^{-3}
q_e	$rads^{-1}$	10^{-3}	10^{-3}
r_e	$rads^{-1}$	10^{-3}	10^{-3}
ϕ_e	rad	10^{-3}	10^{-3}
ψ_e	rad	10^{-3}	10^{-3}
ϕ_{Le}	rad	10^{-3}	10^{-3}
θ_{Le}	rad	10^{-3}	0.6287
$\dot{\phi}_{Le}$	$rads^{-1}$	10^{-3}	10^{-3}
$\dot{\theta}_{Le}$	$rads^{-1}$	10^{-3}	10^{-3}

A high level overview of the code architecture in B.1 is shown in figure 2.

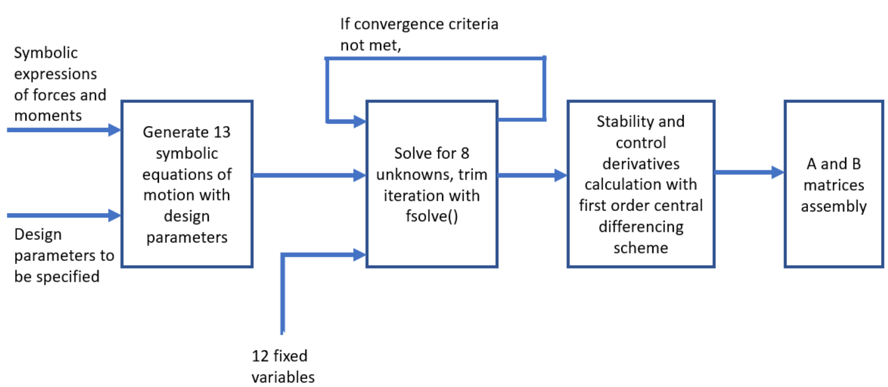

Figure 2: Code architecture to generate state space model

Part IV
Results

6 Model creation and verification

6.1 Initial state space model with flapping equations

The A matrix with flapping equations at hover can be found in A.4.1. It is clear that the derivatives related to flapping are highly suspect. The trim point calculation was also impossible due to the enormous symbolic expressions used for the flap states in MATLAB. After back substitution of trim variables to the $\dot{x}$ vector, the L2 norm was computed and was 27900: this was certainly not a trim point. In order to make progress, we decided to remove the 4 flapping equations from our model.

6.1.1 Modified state space model without flapping equations

Table 4 shows the variables that were produced after the fsolve() algorithm. Table 5 shows the residuals produced per step when iterating for the trim point for hover. The solution is well ordered and reaches convergence in 3 steps. This suggests that the point is a local minimum. Similar results are seen for the cruise case.

Table 4: Solved variables

Solved variables	Units	Hover with Payload	Cruise with Payload
θ_e	rad	0.0454	0.0394
θ_0	rad	0.147	0.0127
θ_{1s}	rad	2.51e-04	7.08e-05
θ_{1c}	rad	2.65e-04	-1.43e-05
θ_{2s}	rad	2.51e-04	7.08e-05
θ_{2c}	rad	2.65e-04	-1.43e-05
δ_e	rad	0	0
δ_R	rad	0	0

Table 5: Hover: fsolve() results for trim point calculations

Iteration	Func-count	Residual	First order optimality	Lambda	Norm of step
0	9	0.242	4.83	0.01	
1	18	3.99E-08	0.00196	0.001	0.0477
2	27	3.84E-18	1.92E-08	0.0001	1.94E-05
3	36	3.77E-30	1.9E-14	1E-05	1.90E-10

Back substitution of equilibrium points to the equation of motion yields the results below. The first derivative of all states are close to zero, save $\dot{w}$ and $\dot{p}$. This is likely due to helicopter dynamics rather than a mistake with the numerical solver, as the results in table 5 suggests that the numerical method has a low error.

The A and B matrices for hover are shown in equations 26 and 27. The corresponding matrices for cruise can be found in appendix A.4.2.

$$\dot{x}_{hover} = \begin{pmatrix} 2.0416e-15 \\ -2.253 \\ 18.1927 \\ 17.7143 \\ -0.0395 \\ -0.1627 \\ 0.1046 \\ 0.099 \\ 0.1011 \\ 0.001 \\ 0.001 \\ -0.0865 \\ -0.0057 \end{pmatrix} \qquad \dot{x}_{cruise} = \begin{pmatrix} 1.2943e-15 \\ -5.0122 \\ 20.5126 \\ 16.4716 \\ 0.1947 \\ -0.3458 \\ 0.104 \\ 0.099 \\ 0.1011 \\ 0.001 \\ 0.001 \\ 0.0009743 \\ -0.5366 \end{pmatrix}$$

1: Hover and cruise: Resulting $\dot{x}$ vector after back-substitution of all variables to non-linear equations of motion.

$$A_{hover} = \begin{pmatrix} -0.0142 & 0.11 & -0.122 & -0.0222 & -0.0999 & 0.1 & 0 & -9.8 & 0 & 0 & 0 & 0 & 0 \\ -0.0463 & -0.0206 & 0.221 & 0.1 & -0.0206 & -1 & 9.8 & -0.00446 & 0 & 0 & 0 & 0 & 0 \\ -0.122 & -0.0278 & -0.464 & -0.1 & 0.979 & -1.01 & 0.098 & 0.446 & 0 & 0 & 0 & 0 & 0 \\ -0.435 & -0.128 & -1.15 & 0.0304 & 46.7 & -46.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 134 & 0.0569 & -0.0947 & 134 & 0.269 & 134 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -181 & 0.0472 & -0.132 & -181 & -180 & -0.00449 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.000455 & 0.0455 & 0 & -0.0992 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -0.01 & -0.101 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 1 & 0.0991 & 0.0046 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0.026 & -0.941 & -1.11 & 2.71 & -0.274 & 0 & -0.217 & 0.487 & 0 & 0 \\ 0 & 0 & 0 & 0.073 & -0.009 & 0.122 & 0.029 & 0.277 & 0 & -0.009 & -0.302 & 0 & 0 \end{pmatrix} \tag{26}$$

$$B_{hover} = \begin{pmatrix} -3.17 & -0.0542 & -0.00573 & -0.0542 & -0.00573 & 0 & 0 \\ 14.6 & 0.0604 & -0.0408 & 0.0604 & -0.0408 & 0 & 0 \\ -58.8 & -0.222 & -0.0103 & -0.222 & -0.0103 & -0.00516 & 0 \\ -126 & 183 & 24.5 & -183 & 24.6 & 0 & 0 \\ 0.2 & -4.37 & 36.3 & 4.46 & 36.3 & -0.0172 & 0 \\ -1.18 & -3.91 & 11 & 3.91 & 11 & 0 & -1.39e-05 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{27}$$

7 Static Stability

Static stability is the immediate response of an aircraft to return to an equilibrium state after being perturbed. A preliminary look at the static stability characteristics of the helicopter would help the controls team better define their control goals. 2 methods are proposed for this analysis [20] and [21]. We shall adopt the latter approach. Starting off, the helicopter and slung load should be analysed as one system rather than separate components for static stability.

Unlike traditional fixed wing analysis, the concept of a static margin is not commonly used for helicopters [3]. Instead, analysis is done on the helicopter at different flight conditions; the behaviour of the helicopter at hover can be very different from cruise.

For rotorcraft, the following conditions must be true for positive static stability along that degree of freedom [22]:

$$\begin{aligned}
M_u &> 0 && \text{For positive forward speed static stability}\\
M_q &< 0 && \text{For positive pitch angle static stability}\\
L_p &< 0 && \text{For positive roll angle static stability}\\
N_r &< 0 && \text{For positive yaw angle static stability}
\end{aligned}$$

Figure 3 depicts the change in relevant stability derivatives with a change in u. This was selected because the only 2 mission legs we have chosen to study, hover and cruise, vary mainly in the value of u_e. The plots of figure 3 were made holding all other states and control inputs constant. This was done to be in accordance with small perturbation theory: to vary only 1 variable at a time.

One observation has to be made: M_u at trimmed hover is at a higher value than the extrapolated value suggested in the M_u vs u plot in figure 3.This is because the results in table 6 were taken at trim points at the specified u_e. This explains the discrepancy in the trend of M_u with the corresponding result in 6: Nonetheless, the plot in figure 3 gives a good understanding of how the static stability derivatives behave over a typical flight mission with the payload attached.

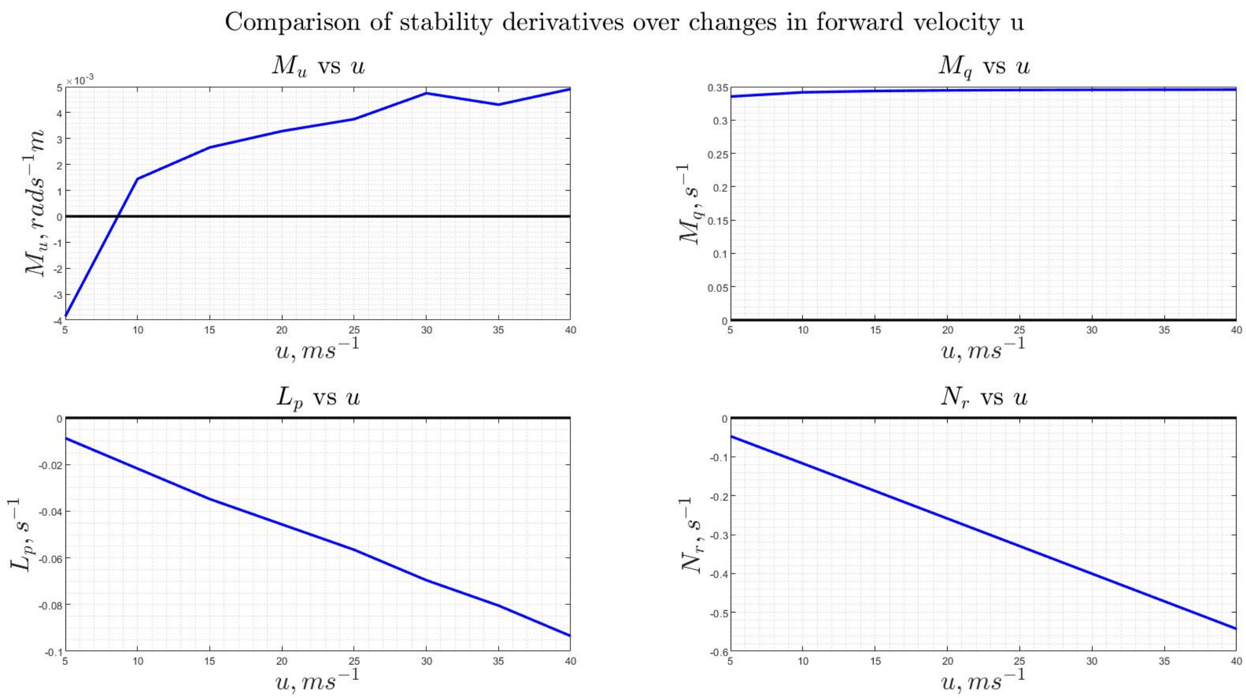

Figure 3: Change in static stability derivatives over forward speed

Table 6: Relevant stability derivatives for static stability

Mission Leg	Hover	Conclusion	Cruise	Conclusion
$M_u(rads^{-1}m)$	0.0333	Stable	0.00602	Stable
$M_q(s^{-1})$	0.289	Unstable	0.377	Unstable
$L_p(s^{-1})$	0.0304	Unstable	-0.0696	Stable
$N_r(s^{-1})$	0.00324	Unstable	-0.401	Stable

8 Dynamic Stability

Dynamic stability of an aircraft is the ability to return to an initial trimmed state upon receiving a small perturbation. [23] Immediately after the perturbation, the resulting motion will be influence by multiple modes. However, the most unstable mode will dominate the dynamic motion over time. Following the method in [2], we know that eigenvalues λ of the produced state-space model of our helicopter define these modes. The associated eigenvector determines the residues, indicating how much each mode contributes to an output, allowing us to classify it[24].
For a schematic representation of the root-locus diagram, refer to Appendix A.2. Key parameters are computed as:

$$\xi = \frac{\mathbb{R}(\lambda)}{|\lambda|} \tag{28}$$

$$\omega_n = |\lambda| \tag{29}$$

$$\omega_d = \mathbb{I}(\lambda) \tag{30}$$

$$t_{1/2} = -\frac{\ln 2}{\mathbb{R}(\lambda)} \tag{31}$$

$$t_2 = -\frac{\ln 2}{\mathbb{R}(\lambda)} \tag{32}$$

$$T = \frac{2\pi}{\omega_d} \tag{33}$$

Typical dynamic stability analysis of fixed wing aircraft consists of uncoupling longitudinal and lateral stability modes. However, upon closer inspection of our generated A matrix in A.4 reveals that this assumption is not applicable to our synchropter. Helicopter dynamics is intrinsically highly coupled and is supported by our findings. Hence, it has been decided to analyse the system as a whole, and to classify lateral and longitudinal modes only after generating the eigenvalues and eigenvectors of the full system.

8.1 Method to classify modes

There is merit in detailing how to use the eigenvectors to identify modes, especially since we will have to identify 13 eigenvalues in the full system. With reference to [2], we know that given a perturbation in states $\dot{x}$, the open loop response can be written as

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} \tag{34}$$

Then, using the general form

$$\boldsymbol{x} = \boldsymbol{x_i} e^{\lambda_i t} \tag{35}$$

We obtain

$$(\boldsymbol{A} - \lambda_i \boldsymbol{I})\boldsymbol{x_i} = 0 \tag{36}$$

where $\boldsymbol{x}_i$ is the eigenvector associated with the eigenvalue λ. The eigenvector can then be seen as the dynamic response of the states to the initial perturbation [25].
The amplitude of the eigenvector component is the corresponding amplitude of the perturbation of the associated state. Since our state vector contains velocities, angular rates and angular displacements, a good way to compare the effects of the perturbation on the steady state behaviour is to perform normalisation. Specifically like so:

$$q_i = \frac{\boldsymbol{x_i}}{\boldsymbol{x}} \tag{37}$$

where q_i refers to the relative effect of the perturbation and $\boldsymbol{x}$ refers to the steady state value before the perturbation. We will use $\boldsymbol{q_i}$ to identify the states that dominate a mode. The state corresponding to the maximum $\boldsymbol{q_i}$ is the state that is dominant. This method allows us to study even highly coupled systems. All the normalised eigenvectors are then stored as $\boldsymbol{Q}_{\text{mission leg}}$, and shown in A.5. Qualitative write ups of the expected dominant states can be found in [3], we cross reference our dominant states with these.

8.2 Eigenvalues identification

Tables 7 and 8 details the eigenvalues and corresponding eigenvectors. The corresponding eigenvectors and normalised eigenvector matrix can be found in A.5. We have used the method detailed above to identify the modes. The yaw heading mode has been omitted from the tables.

Table 7: Dynamic Stability Modes, Hover

Mode	Phugoid	Heave Subsidence	Pitch Subsidence	Dutch Roll	Roll subsidence	Spiral mode	Payload Mode	Payload Mode
λ	0.5707± 0.815i	-0.1737	-5.560	$0.542 \pm 97.9i$	-1.066	4.377	$\pm 0.0419i$	$\pm 3.340i$
$\omega_d(s^{-1})$	0.815	-	-	97.9	-	-	0.0419	3.340
$\omega_n(s^{-1})$	0.995	0.1737	5.560	97.9	1.066	4.377	0.0419	3.340
ξ	0.5735	1	1	0.000554	1	1	0	0
$t_{1/2}(s)$	-	3.990	0.12466	-	0.65023	-	-	-
$t_2(s)$	1.21	-	-	1.2788	-	0.15836	-	-
$T(s)$	7.7094	∞	∞	0.064179	∞	∞	150.0	1.88

Table 8: Dynamic Stability Modes, Cruise

Mode	Phugoid	Heave Subsidence	Pitch Subsidence	Dutch Roll	Roll subsidence	Spiral mode	Payload Mode	Payload Mode
λ	$-0.5501\pm$ 0.719i	-0.377	-7.93	0.5429± 98.16i	0.899	6.866	$\pm 0.0419i$	$\pm 3.340i$
$\omega_d(s^{-1})$	0.719	-	-	98.16	-	-	0.0419	3.340
$\omega_n(s^{-1})$	0.905	0.377	7.93	98.16	0.899	6.866	0.0419	3.340
ξ	0.608	1	1	0.000531	1	1	0	0
$t_{1/2}(s)$	-	1.84	0.0874	-	-	-	-	-
$t_2(s)$	1.26	-	-	1.2788	0.771	0.100	-	-
$T(s)$	8.738	∞	∞	0.0643	∞	∞	150.0	1.88

Figure 4 indicates that at the hover phase, there are 3 notable unstable modes: phugoid, dutch roll and spiral subsidence. The associated periods and time to double amplitude are shown in table 7. From [3], and from flight tests of various rotorcraft [26], an unstable phugoid mode is very much to be expected for helicopters. This has a moderate natural frequency. As highlighted in [3], the phugoid mode for rotorcraft is different from fixed wing aircraft in that it could lead to uncontrollable rolling moments if left unchecked for long enough. This mode becomes stable with increasing forward speed when we look at the root locus plot at cruise in figure 6. The eigenvalues in table 7 are of the expected order of magnitude as in [3] and [21]. The payload modes are discussed in [5]

The only difference is that our synchropter has a unstable spiral subsidence mode at hover, contrary to typical rotorcraft. The spiral subsidence mode is dominated by the N_r derivative. Interviews with manufacturers of the Karman K-Max highlighted that synchropters are prone to spiral instabilities when turning at high forward speed. The vertical tail is particularly important in generating the control inputs to counter such instabilities[7].

The eigenvalue related to the unstable dutch roll appears to be erroneous. The real part is of a correct magnitude but the imaginary part is far too high. An ω_d of 98 Hz is not feasible for such a mode.

To evaluate the behaviour at cruise, we see that the very oscillatory dutch roll mode is still present from hover, with the same high ω_n of 98Hz. We have concluded that this is also erroneous. Referring to figure 5, there are 2 pairs of purely oscillatory eigenvalues. These are related to payload dynamics. There is one large negative real pole, this

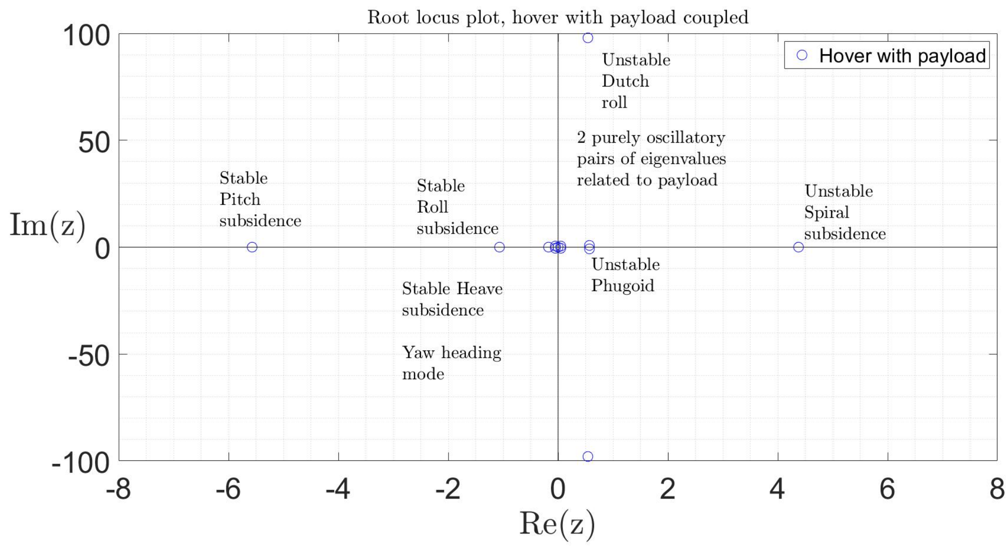

Figure 4: Root locus plot for hover

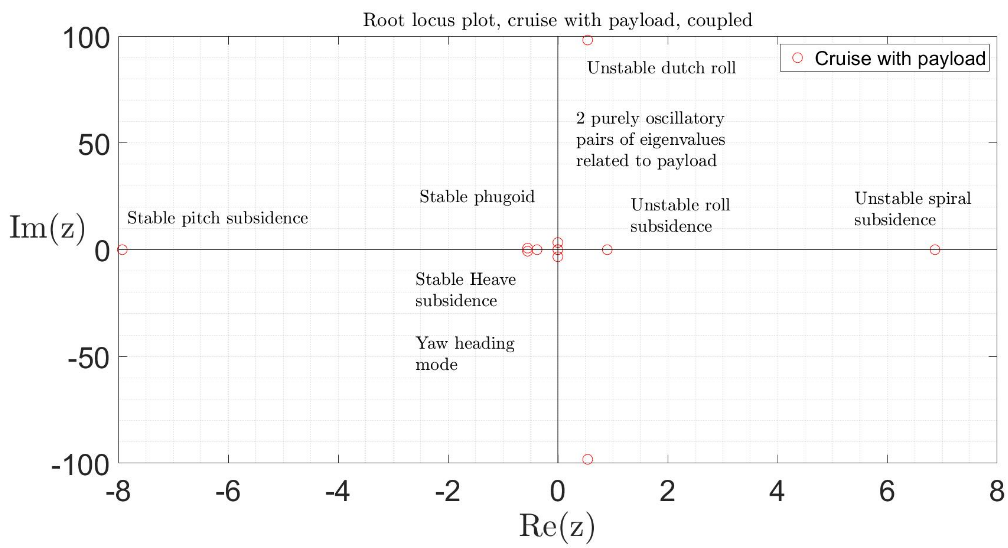

Figure 5: Root locus plot for cruise

is the pitch subsidence mode. There is one small negative real pole, related to the heave subsidence mode. There is one mode at the origin representing the yaw heading mode. The phugoid has moved over to the left half plane, becoming stable. The roll subsidence mode has moved over to the right half plane with an increase in forward velocity. The unstable spiral subsidence mode has moved even further right compared to at hover.
Critical modes to keep in mind would be modes that change behaviour with forward speed. These are the phugoid and the roll subsidence modes. The movement of the poles with forward speed is shown in figure 6.

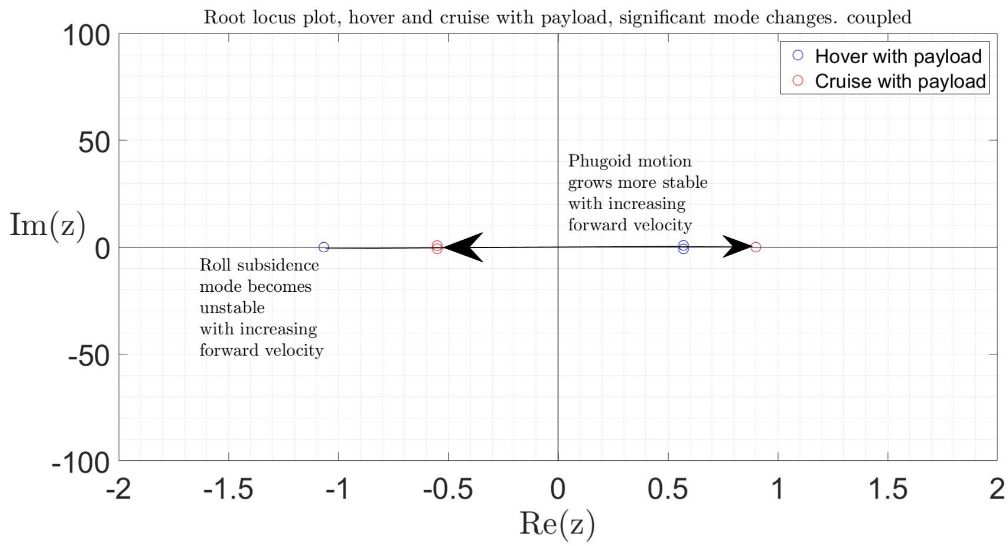

Figure 6: Identification of modes which change behaviours from hover to cruise

Part V
Discussion of methods and results

9 State space model, refinement

From the dynamic stability analysis, the erroneous eigenvalue for Dutch roll suggests that there has been a mistake in the formulation of the A matrix, likely in entries related to the states p or q, as Dutch roll is dominated by these. It is recommended to relook at the A matrix for future iterations.

The implementation of flap equations into the A matrices proved to be difficult on this occasion. However, the benefit of adding more states is clear, as the A matrix would be better able to model the blade dynamics, particularly crucial for our synchropter where there is potentially more inter-rotor interaction, complicating the wake and the aerodynamics near the rotor operation region.

It is recommended to implement the flap equations numerically rather than symbolicly in the future to reduce run time.

10 Static stability

The static stability of the helicopter is acceptable. Most helicopters are inherently unstable, but our design has proven to be speed stable in both flight regimes and experiences roll and yaw damping at cruise. This suggests that the current design is a decent model for implementing controllers.

Note that the static stability analysis only applies to the operating conditions that we have specified. Classification of static stability in any other flight regime requires a re adjustment of parameters.

11 Dynamic stability

Disregarding the erroneous dutch roll mode, the dynamic stability modes of our helicopter have been well identified. All stable modes have an acceptable $t_{1/2}$, whilst the unstable spiral mode is common to most helicopter designs. The unstable roll subsidence mode is unique to our synchropter, likely due to the rotor configuration and mass balance being different from most other helicopters.

Despite the unstable modes, these can be rectified with a well-designed control scheme. One added benefit of an unmanned vehicle is that we do not need to design for passenger comfort; we could theoretically design beyond human limits in terms of response rates or response amplitudes to oscillations. A concrete example is that dutch roll is seen as an annoyance to passengers and commercial aircraft are designed with related constraints in mind

[27]. Our unmanned concept is free from such constraints and it would be wise to keep this liberty in mind.

Part VI
conclusion

A state space model of the synchropter has been constructed, trim point calculation has been verified and has been used to study the open-loop response to small perturbations. Various modes were found to be stable, some which are common to all helicopters, while some proved to be unique to our novel design.

Our selection of analysing hover and cruise with the stage 2 payload is far from the most constraining case of the proposed mission profile, but this is a good stepping stone towards analysing more complex phases, such as the descent phase or to study how the synchropter behaves in a general trimmed turn. With the tools that have been developed, these goals are within reach.

Given more time, a study into the open loop response to step impulse inputs of the various control inputs would be highly beneficial to profiling the behaviour of the synchropter. This would generate a useful data set for the controls team to reference.

References

[1] C. Sultan T. Oktay. *Modeling and control of a helicopter slung-load system.* Virginia Polytechnic Institute and State University, 2012.

[2] E. Levis. *Chapter 9, AE2-211: Mechanics of Flight.* Imperial College London, Department of Aeronautics, 2019.

[3] Gareth D. Padfield. *Helicopter Flight Dynamics, The Theory and Application of Flying Qualities and Simulation Modelling, Second Edition.* Blackwell Publishing, 2007.

[4] Ethem Orhan (2020). *Generic Nonlinear Helicopter Model.* MATLAB Central File Exchange., Retrieved June 11, 2020.

[5] A. Dubey. *Flight Mechanics Modelling of a Novel Synchropter Concept for the Aerial-Recovery of Falling Rocket Stages.* GDP Coursework, 2020.

[6] I. Tavares. *Wholly recoverable launch system.* GDP Coursework, 2020.

[7] W. Horn V. Madhavan H. Park H. Prahlad P. Samuel M. Shaner N.Koratkar, N.Costes. *CALVERT High-Speed V/STOL Personal Transport.* University of Maryland, Alfred Gessow Rotorcraft Center, Department of Aerospace Engineering, 1999.

[8] C. Gliner. *Business Case and Vehicle Management Overview of a Novel Synchropter Concept for the Aerial-Recovery of Falling Rocket Stages.* GDP Coursework, 2020.

[9] H. Serpen. *Flight simulation of aerial capture and handling quality for a synchropter concept for the aerial-recovery of falling rocket stages.* GDP Coursework, 2020.

[10] F.Bastien. *Modelling synchropter flight dynamics in Aerial Recovery of Rocket Stage.* GDP Coursework, 2020.

[11] Le Yin. *Aerodynamic Design and Analysis of Empennage and Rotor Tip of Synchropter Concept for the Aerial-Recovery Mission.* GDP Coursework, 2020.

[12] K.K.T. Thanapalan. *Modelling of a Helicopter System.* Faculty of Engineering Sciences, University College London, 2010.

[13] Tiago D.T.Rita. *Model Helicopter Control.* IDMEC/IST, Universidade Técnica de Lisboa (TU Lisbon), 2009.

[14] K.S. Hon. *Flight performance analysis of the novel synchrocopter.* GDP Coursework, 2020.

[15] Celi. Roberto. *ENAE 635- Helicopter Stability and Control Class Notes.* University of Maryland, A James Clark School of Engineering, Department of Aerospace Engineering, 2020.

[16] C. Chen C-H. Fua T.H. Lee B.Ren, S.S. Ge. *Stability Analysis for Rotary-Wing Aircraft.* Springer, New York, NY, 2011. Retrieved from: `https://link.springer.com/chapter/10.1007/978-1-4614-1563-3_3`.

[17] I. Chopra B. Panda. *Flap-Lag-Torsion Stability in Forward Flight.* Center for Rotorcraft Education and Research, Department of Aerospace Engineering, University of Maryland College Park, 1985.

[18] G. M. El-Bayoumi M.M. Abdelrahman H.S. Hassan, A.M. Bayoumy. *Modeling, Trimming and Simulation of a Full Scale Helicopter.* Military Technical College, Kobry Elkobbah, Cairo, Egypt, 2017.

[19] J. J. More. *"The Levenberg-Marquardt Algorithm: Implementation and Theory," Lecture Notes in Mathmatics, Vol. 630.* 1978.

[20] A. M. Volodko A. N. Sviridenko. *Static Stability and controllability of helicopter with an external load.* CSTS Dinamika, 2007.

[21] Q. Qi. C. Yihua and D. Zhang. *Flight dynamic characteristics of tiltrotor aircraft slung-load system.* Journal of Aircraft, vol 54. no.3 pp 1221-1228, 2007.

[22] A. Gessow and K. B. Amer. *An introduction to the physical aspects of helicopter stability.* NACA, Langley Aeronautical Laboratory, Langley Air Force Base, Va, 1982.

[23] ESDU. *Introduction to aerodynamic derivatives, equations of motion and stability(including the classical criteria of longitudinal static stability and control, and description of the lateral modes of motion).*

[24] D. Andrisani John B. Davidson. *Lateral-directional eigenvector flying qualities guidelines for high performance aircraft.* NASA, Langley Research Center, 1996.

[25] *Longitudinal Aircraft Dynamics.* Virginia Tech, Kevin T Crofton Department of Aerospace and Ocean Engineering, 1996.

[26] C. Li R. Chen K. Lu, C.Liu. *Flight Dynamics Modeling and Dynamic Stability Analysis of Tilt-Rotor Aircraft.* Hindawi, International Journal of Aerospace Engineering, 2019.

[27] A.Chakravarty T.J. Goslin, F.H.Ansari. *An Optimized Yaw Damper for Enhanced Passenger Ride Comfort.* NACA, Langley Aeronautical Laboratory, Langley Air Force Base, Va, 1982.

A Appendicies

A.1 Payload angle free body diagram

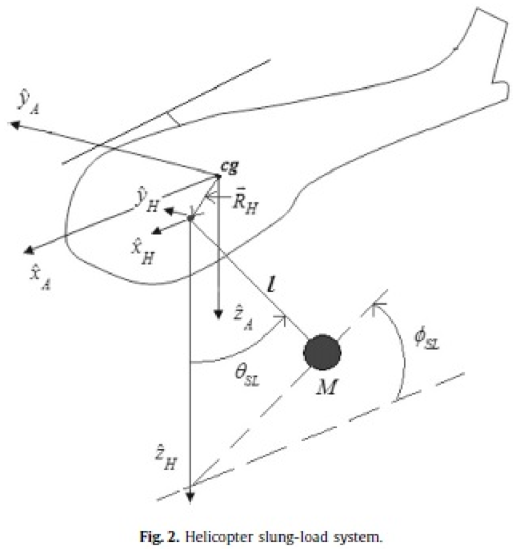

Figure 8: Slung load angle conventions [1]

The states ϕ_L and θ_L are referenced from the convention adopted in [1].ϕ_L corresponds to ϕ_{SL} while θ_L corresponds to θ_{SL}.

A.2 Root locus diagram convention

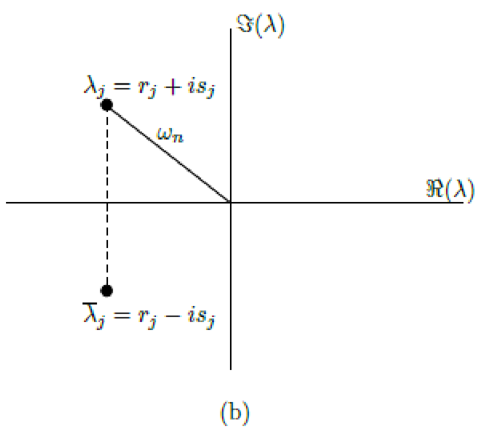

Figure 9: Root locus diagram [2]

Figure 9 shows the convention which will be used in section 8.

A.3 Schematic of A matrix

$$A = \begin{pmatrix} X_u & X_v + r_e & X_w - q_e & X_p & X_q - w_e & X_r + v_e & 0 & -g\cos(\theta_e) & 0 & X_{\phi_L} & X_{\theta_L} & X_{\dot{\phi} L} & X_{\dot{\theta} L} \\ Y_u - r_e & Y_v & Y_w + p_e & Y_p + w_e & Y_q & Y_r - u_e & g\cos(\theta_e)\cos(\phi_e) & -g\sin(\theta_e)\sin(\phi_e) & 0 & Y_{\phi_L} & Y_{\theta_L} & Y_{\dot{\phi} L} & Y_{\dot{\theta} L} \\ Z_u + q_e & Z_v - p_e & Z_w & Z_p - v_e & Z_q + u_e & Z_r & A_{3,7} & A_{3,8} & 0 & Z_{\phi_L} & Z_{\theta_L} & Z_{\dot{\phi} L} & Z_{\dot{\theta} L} \\ L_u & L_v & L_w & L_p & L_q + r_e(I_{yy} - I_{zz}) & L_r + q_e(I_{yy} - I_{zz} & 0 & 0 & 0 & L_{\phi_L} & L_{\theta_L} & L_{\dot{\phi} L} & L_{\dot{\theta} L} \\ A_{5,1} & M_v & M_w & M_p + r_e(I_{zz} - I_{xx}) & M_q & M_r + p_e(I_{zz} - I_{xx}) & 0 & 0 & 0 & M_{\phi_L} & M_{\theta_L} & M_{\dot{\phi} L} & M_{\dot{\theta} L} \\ A_{6,1} & N_v & N_w & N_p + q_e(I_{xx} - I_{yy}) & N_q + p_e(I_{xx} - I_{yy}) & N_r & 0 & 0 & 0 & N_{\phi_L} & N_{\theta_L} & N_{\dot{\phi} L} & N_{\dot{\theta} L} \\ 0 & 0 & 0 & 1 & tan(\theta_e)\sin(\phi_e) & \cos(\phi_e)\tan(\theta_e) & A_{7,7} & A_{7,8} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cos(\phi_e) & -\sin(\phi_e) & -A_{8,8} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin(\phi_e)\sec(\theta_e) & cos(\phi_e)\sec(\theta_e) & A_{9,7} & A_{9,8} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \dot{\phi}_{Lu} & \dot{\phi}_{Lv} & \dot{\phi}_{Lw} & \dot{\phi}_{Lp} & \dot{\phi}_{Lq} & \dot{\phi}_{Lr} & 0 & 0 & 0 & \dot{\phi}_{L\phi_L} & \dot{\phi}_{L\theta_L} & \dot{\phi}_{L\dot{\phi}_L} & \dot{\phi}_{L\dot{\theta}_L} \\ \dot{\theta}_{Lu} & \dot{\theta}_{Lv} & \dot{\theta}_{Lw} & \dot{\theta}_{Lp} & \dot{\theta}_{Lq} & \dot{\theta}_{Lr} & 0 & 0 & 0 & \dot{\theta}_{L\phi_L} & \dot{\theta}_{L\theta_L} & \dot{\theta}_{L\dot{\phi}_L} & \dot{\theta}_{L\dot{\theta}_L} \end{pmatrix} \tag{38}$$

Where

$A_{3,7} = \cos(\theta_e)\sin(\phi_e)$
$A_{3,8} = g\sin(\theta_e)\cos(\phi_e)$
$A_{5,1} = M_u + r_e(I_{zz} - I_{xx})$
$A_{6,1} = N_u + q_e(I_{xx} - I_{yy})$
$A_{7,7} = q_e\cos(\phi_e)\tan(\theta_e) - r_{\cos(\phi_e)\tan(\theta_e)}$
$A_{7,8} = q_e\sin(\phi_e)sec^2(\theta_e) - r_e$
$cos(\phi_e)sec^2$
$A_{8,8} = q_e\sin(\phi_e) - r_e\cos(\phi_e)$
$A_{9,7} = q_e\cos(\phi_e)\sec(\theta_e) - r_e\sin(\phi_e)\sec(\theta_e)$
$A_{9,8} = \tan(\theta_e)\sec(\theta_e)(q_e\sin(\phi_e) + r_e\cos(\phi_e))$

A.4 State Space Matrices

A.4.1 Hover with Payload, with flap

$$x = \begin{pmatrix} u_e & v_e & w_e & p_e & q_e & r_e & \phi & \theta & \psi & \beta_{1s} & \beta_{1c} & \beta_{2s} & \beta 2c \\ \phi_L & \theta_L & \dot{\phi}_L & \dot{\theta}_L \end{pmatrix} \tag{39}$$

$$= \begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0 & 0.1 & 0.1 & 0.05 & 0.1 & 0.05 \\ 0.01 & 0.01 & 0.001 & 0.001 \end{pmatrix} \tag{40}$$

$$
A = \begin{pmatrix}
-0.29344 & 0.6524 & -0.020624 & -0.0017958 & -0.1 & 0.10251 & 0 & -9.81 & 0 & 2657 & 3793 & 3532 & 3053 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
0.89927 & -1.1739 & 0.24837 & 0.098648 & 0.0017107 & -0.097286 & 9.71 & 0 & 0 & 3694 & -2548.549 & 2958.348 & -3411 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
-2.425 & -2.6357 & -2.6507 & -0.10359 & 0.10359 & -0.1 & 0.979 & 0 & 0 & -780 & 860 & 930 & -720 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
-3.9783 & -22.1304 & -10.5 & -0.41304 & 46.5583 & -46.8843 & 0 & 0 & 0 & 4120 & -3270 & 4660 & -5430 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
134.3001 & -1.0954 & -0.13461 & 133.8954 & 0.12049 & 133.9356 & 0 & 0 & 0 & -3491 & -4280 & -4333 & -3491 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
-178.9194 & 2.0483 & 1.5839 & -180.7078 & -180.6467 & 0.048333 & 0 & 0 & 0 & -232 & -356 & -250 & 992 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -0.08 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
0 & 0 & 0 & 0 & 0.995 & 0.995 & 0.089 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
0 & 0 & 0 & 0 & 0.099833 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
201390 & -70170 & -17790 & -16000 & -191580 & -10 & 0 & 0 & 0 & -157500 & -157500 & 0 & 0 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
201390 & -70170 & -17790 & -16000 & -191580 & -10 & 0 & 0 & 0 & -157500 & -157500 & 0 & 0 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
201390 & -70170 & -17790 & -16000 & -191580 & -10 & 0 & 0 & 0 & 0 & 0 & -157500 & -157500 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
201390 & -70170 & -17790 & -16000 & -191580 & -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & & & & & & & & & \\
0 & 0.013 & 0 & -0.123 & -1014.164 & -1014.041 & 2160 & -10.9 & 0 & 0 & 0 & 0 & 0 \\
-107 & -1167 & 0 & 0 & & & & & & & & & \\
0 & 0 & 0 & 0.102 & 0 & 0.102 & 0 & 0.219 & 0 & 0 & 0 & 0 & 0 \\
0.012 & -0.221 & 0 & 0 & & & & & & & & &
\end{pmatrix} \tag{41}
$$

$$
B = \begin{pmatrix}
3.9593 & -0.086461 & -0.042227 & -0.068724 & -0.085536 & 0 & 0 \\
0.69925 & -0.20611 & -0.45838 & 0.44975 & 0.1798 & 0 & 0 \\
-39.3492 & -1.2823 & -1.1989 & -1.1803 & -1.271 & 0 & 0 \\
-184.6092 & 125.2809 & 85.735 & 93.9843 & 105.0058 & 0 & 0 \\
-8.5962 & -17.0075 & 25.4686 & -23.3726 & 19.3445 & 0 & 0 \\
-3.4795 & 8.1588 & -4.5123 & -4.9275 & 9.8331 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-575160 & 0 & 0.0068466 & 0 & 0 & 0 & 0 \\
-575160 & -0.0068466 & 0 & 0 & 0 & 0 & 0 \\
-575160 & 0 & 0 & 0 & 0.0068466 & 0 & 0 \\
-575160 & 0 & -0.0068466 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \tag{42}
$$

A.4.2 A and B matrices, Cruise

$$A_{cruise} = \begin{pmatrix} -0.00164 & 0.107 & -0.129 & -0.0197 & -0.1 & 0.101 & 0 & -9.8 & 0 & 0 & 0 & 0 & 0 \\ -0.0969 & -0.0371 & 0.208 & 0.108 & -0.017 & -30 & 9.8 & -0.00387 & 0 & 0 & 0 & 0 & 0 \\ 0.0897 & 0.00835 & -0.423 & -0.131 & 30 & -30 & 0.098 & 0.387 & 0 & 0 & 0 & 0 & 0 \\ -0.0239 & 0.25 & -0.928 & -0.0696 & 46.5 & -46.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 134 & -0.00432 & 0.0294 & 134 & 0.377 & 134 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -181 & 0.124 & -0.0802 & -181 & -180 & -0.401 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.000394 & 0.0394 & 0 & -0.0991 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -0.01 & -0.101 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 1 & 0.0991 & 0.00399 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0.000233 & 1.16e-05 & 0 & 0.114 & -0.0719 & -0.154 & 0.46 & -0.0469 & 0 & -0.00182 & 0.0211 & 0 & 0 \\ -0.003 & 0 & 0 & -0.333 & 0.006 & 0.557 & 0.103 & 0.755 & 0 & 0.029 & -11.2 & 0 & 0 \end{pmatrix} \tag{43}$$

$$B_{cruise} = \begin{pmatrix} -3.16 & -0.176 & 0.00358 & -0.176 & 0.00358 & 0 & 0 \\ 15.1 & 1.65 & 0.175 & 1.65 & 0.175 & 0 & 0.00103 \\ -60.6 & -6.62 & 0.0464 & -6.62 & 0.0464 & 0.031 & 0 \\ -130 & 155 & 4.3 & -184 & 4 & 0 & 0 \\ -0.139 & -0.329 & 37 & -0.0115 & 37 & 0.2 & 0 \\ 0.00454 & -2.67 & 11.6 & 2.52 & 11.6 & 0 & -0.00831 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{44}$$

A.5 Dynamic Stability Analysis

A.5.1 Hover

$$\lambda_{hover} = \begin{pmatrix} 0 \\ 0.542928 + 97.9193i \\ 0.542928 - 97.9193i \\ -5.5692 \\ 4.3776 \\ -1.0663 \\ 0.57078 + 0.81565i \\ 0.57078 - 0.81565i \\ 0.049589 + 0.51182i \\ 0.049589 - 0.51182i \\ -0.049589 + 0.51182i \\ -0.049 - 0.51182i \\ -0.17376 \end{pmatrix} \tag{45}$$

Figure 10: Hover eigenvalues

$$\boldsymbol{Q}_{hover} = \begin{pmatrix} 0 & 0.014 & 0.014 & 2 & 2.8 & 1 & 10 & 2.3 & 5.6e-15 & 5.6e-16 & 4.2e-16 & 4.2e-14 & 0.027 \\ 0 & 0.073 & 0.073 & 0.057 & 2.7 & 9.4 & 92 & 20 & 1.5e-13 & 1.5e-14 & 1.2e-14 & 1.2e-12 & 990 \\ 0 & 0.073 & 0.073 & 0.1 & 0.15 & 0.58 & 1.7 & 0.38 & 1.1e-14 & 1.1e-15 & 5.3e-16 & 5.3e-14 & 79 \\ 0 & 3.4 & 3.4 & 4.1 & 2.8 & 1.1 & 9.4 & 2.1 & 1e-14 & 1e-15 & 4.9e-16 & 4.9e-14 & 3.9 \\ 0 & 6.4 & 6.4 & 6.3 & 5.5 & 0.092 & 1.1 & 0.23 & 7.2e-15 & 7.2e-16 & 2.7e-16 & 2.7e-14 & 3.6 \\ 0 & 6.9 & 6.9 & 5.8 & 5.8 & 0.096 & 1.1 & 0.23 & 7.1e-15 & 7.1e-16 & 1.4e-16 & 1.4e-14 & 4.4 \\ 0 & 0.032 & 0.032 & 0.67 & 0.6 & 1 & 9.4 & 2.1 & 6.1e-15 & 6.1e-16 & 6.6e-16 & 6.6e-14 & 14 \\ 0 & 0.065 & 0.065 & 1.1 & 1.3 & 0.014 & 0.34 & 0.074 & 2.6e-15 & 2.6e-16 & 2.1e-16 & 2.1e-14 & 12 \\ 1 & 0.071 & 0.071 & 1.1 & 1.3 & 0.19 & 2 & 0.43 & 1.9e-14 & 1.9e-15 & 2.6e-16 & 2.6e-14 & 33 \\ 0 & 0.0012 & 0.0012 & 0.33 & 0.7 & 1.9 & 26 & 5.6 & 88 & 8.8 & 8.8 & 880 & 110 \\ 0 & 6.6e-05 & 6.6e-05 & 0.0022 & 0.041 & 0.037 & 1.3 & 0.3 & 12 & 1.2 & 1.2 & 120 & 11 \\ 0 & 0.12 & 0.12 & 1.8 & 3.1 & 2 & 25 & 5.6 & 45 & 4.5 & 4.5 & 450 & 20 \\ 0 & 0.0065 & 0.0065 & 0.012 & 0.18 & 0.04 & 1.3 & 0.29 & 6.2 & 0.62 & 0.62 & 62 & 2 \end{pmatrix} \tag{46}$$

Figure 11: Hover Normalised Eigenvector Matrix

A.5.2 Cruise

$$\lambda_{cruise} = \begin{pmatrix} 0 \\ 0.542981 + 98.165i \\ 0.542981 - 98.165i \\ -7.9293 \\ 6.8665 \\ -4.996e-16 + 3.3408i \\ -4.996e-16 - 3.3408i \\ 0.8999 \\ -0.55014 + 0.71953i \\ -0.55014 - 0.71953i \\ -3.9506e-17 + 0.041972i \\ -3.9506e-17 - 0.041972i \\ -0.37795 \end{pmatrix} \tag{47}$$

Figure 12: Cruise eigenvalues

$$\boldsymbol{Q}_{cruise} = \begin{pmatrix} 0 & 0.015 & 0.015 & 0.56 & 0.52 & 1e-15 & 1e-14 & 7.3 & 19 & 1.9 & 2.5e-17 & 1.5e-14 & 17 \\ 0 & 2 & 2 & 9 & 9.4 & 6.5e-14 & 6.5e-13 & 22 & 91 & 9.1 & 5.2e-16 & 3.3e-13 & 930 \\ 0 & 2.1 & 2.1 & 1.1 & 0.72 & 2.4e-16 & 2.4e-15 & 0.19 & 12 & 1.2 & 1.2e-16 & 7.7e-14 & 290 \\ 0 & 3.2 & 3.2 & 2 & 1.5 & 6.9e-15 & 6.9e-14 & 5.9 & 21 & 2.1 & 3.1e-17 & 1.9e-14 & 27 \\ 0 & 6.1 & 6.1 & 2.7 & 1.9 & 4.9e-15 & 4.9e-14 & 1.5 & 4.7 & 0.47 & 1.1e-17 & 7.2e-15 & 9.2 \\ 0 & 6.6 & 6.6 & 2.4 & 2.1 & 5.3e-15 & 5.3e-14 & 1.5 & 4.7 & 0.47 & 1.3e-17 & 8.4e-15 & 9.7 \\ 0 & 0.031 & 0.031 & 0.24 & 0.21 & 9.6e-16 & 9.6e-15 & 6.5 & 23 & 2.3 & 7.9e-17 & 5e-14 & 69 \\ 0 & 0.062 & 0.062 & 0.34 & 0.28 & 1.1e-15 & 1.1e-14 & 0.9 & 2.6 & 0.26 & 5.3e-18 & 3.3e-15 & 5.7 \\ 0.033 & 0.068 & 0.068 & 0.31 & 0.31 & 1.5e-15 & 1.5e-14 & 2.4 & 7.7 & 0.77 & 5.8e-16 & 3.6e-13 & 44 \\ 0 & 0.00017 & 0.00017 & 0.011 & 0.016 & 0.0054 & 0.054 & 4 & 10 & 1 & 1.6 & 1000 & 180 \\ 0 & 0.00048 & 0.00048 & 0.025 & 0.032 & 2.9 & 29 & 0.031 & 0.95 & 0.095 & 0.0041 & 2.6 & 2 \\ 0 & 0.017 & 0.017 & 0.085 & 0.11 & 0.018 & 0.18 & 3.6 & 9.1 & 0.91 & 0.067 & 42 & 70 \\ 0 & 0.047 & 0.047 & 0.2 & 0.22 & 9.6 & 96 & 0.028 & 0.86 & 0.086 & 0.00017 & 0.11 & 0.75 \end{pmatrix} \tag{48}$$

Figure 13: Cruise Normalised Eigenvector Matrix

B Supporting MATLAB Code

B.1 Generation of equilibrium points, A and B matrices

This script solves for the equilibrium points and A and B matrices.

```
clc
clear
close all

tic
%% Enter forward velocity
u_e=40; %ms^-1

prompt='1 for hover, 2 for any cruise.';
answer=input(prompt);

if answer==1
     th_L_e=10^-1;
elseif answer==2
    th_L_e=deg2rad(52);
end


%% Empennage
u=u_e;%ms^-1. Numerical value for forward velocity
mu=u/(51.050*4.02);
altitude=1000; %ft
rho=0.9; %kg/m^3

S_h=1.40;
S_v=0.57;
c_h=0.59;
a_e=4.8;
d_e=1*10^-3;
a_R=3.490*10^-4;

Ctsigma = Ctsigma(mu,altitude);
w_i = downwash(mu,altitude);
syms u v w del_elevator del_rudder p q r

alpha_H=del_elevator+atan((w+q*(4+0.10019)-w_i)/u);

V_h=sqrt(u^2+(w+q*(4+0.10019)-w_i)^2);
C_ZH=-4.8*alpha_H;

Z_h=.5*rho*V_h*S_h*C_ZH*alpha_H;
M_h=(4+0.10019)*Z_h;


%%%%%Vertical tail
V_v=sqrt(u^2+(v-r*(4+0.10019))^2);
beta_v=0+asin((v-r*(4+0.10019))/V_v);
C_YV=-3*beta_v+0.008*del_rudder;
Y_v=.5*rho*V_v^2*S_v*C_YV;
N_v=-(4+0.10019)*Y_v;


fprintf("Empennage complete.")
```

```
save empennage_forces.mat Z_h M_h Y_v N_v


clearvars -except u_e th_L_e


%% External force generation
%load flap_eqns.mat
load design_constants.mat
load FusForces.mat
load FusMoments.mat
load PayloadEqn.mat
load Rotor1_ext.mat
load Rotor2_ext.mat
load empennage_forces.mat
%FusForces and FusMoments already contain contributions from payload_e


XPL=FH(1);
YPL=FH(2);
ZPL=FH(3);
LPL=MH(1);
MPL=MH(2);
NPL=MH(3);

X_total=Xrotor1+Xrotor2+Xfus;
Y_total=Yrotor1+Yrotor2+Yfus+Y_v;
Z_total=Zrotor1+Zrotor2+Zfus+Z_h;
L_total=Lrotor1+Lrotor2+Yrotor1*h_R1+Yrotor2*h_R2+Zrotor2*y_R1+Zrotor2*y_R2+Yfus*h_fus+Lfus+LPL+YPL*h_PL+
M_total=Mrotor1+Mrotor2-Xrotor1*h_R1-Xrotor2*h_R2+Zrotor1*l_R1+Zrotor2*l_R2+Mfus+MPL-XPL*h_PL+ZPL*l_PL+M_
N_total=Nrotor1+Nrotor2-Yrotor1*l_R1-Yrotor2*l_R2-Yfus*l_fus+Nfus+NPL-YPL*l_PL+N_v-Y_v*l_fn;

syms u v w p q r p_d q_d r_d b1s b1c b2s b2c b1s_d b1c_d b2s_d b2c_d phi_L th_L phi_L_d th_L_d

syms phi th psi
syms phi_euler th_euler psi_euler
syms u_d v_d w_d p_d q_d r_d  phi_euler_d  th_euler_d  psi_euler_d  b1s_d  b1c_d  b2s_d  b2c_d ...
     phi_L_d  th_L_d  th0_d  th1s_d  th1c_d  th2s_d  th2c_d  del_elevator_d  del_rudder_d





 syms u v w p q r p_d q_d r_d phi_L th_L phi_L_d th_L_d
% %Euler angles, from PayloadEqn.mat
 syms phi_euler
 syms th_euler
 syms psi_euler
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %Control inputs as variables
 syms th0 th1s th1c th2s th2c del_elevator del_rudder

first_derivatives=[u_d,v_d,w_d, p_d, q_d, r_d, phi_euler_d, th_euler_d, psi_euler_d, phi_L_d, ...
    th_L_d, th0_d, th1s_d, th1c_d, th2s_d, th2c_d, del_elevator_d, del_rudder_d];

X_total=subs(X_total, [phi th psi],[phi_euler th_euler psi_euler]);

Y_total=subs(Y_total, [phi th psi],[phi_euler th_euler psi_euler]);
Z_total=subs(Z_total, [phi th psi],[phi_euler th_euler psi_euler]);
```

```
L_total=subs(L_total, [phi th psi],[phi_euler th_euler psi_euler]);
M_total=subs(M_total, [phi th psi],[phi_euler th_euler psi_euler]);
N_total=subs(N_total, [phi th psi],[phi_euler th_euler psi_euler]);

phi_L_dd=subs(phi_L_dd, [phi th psi],[phi_euler th_euler psi_euler]);
th_L_dd=subs(th_L_dd, [phi th psi],[phi_euler th_euler psi_euler]);

X_total=vpa(subs(X_total,first_derivatives, zeros(size(first_derivatives)))); %Makes the ...
    calculations faster by removing derivatives.
toc
Y_total=vpa(subs(Y_total,first_derivatives, zeros(size(first_derivatives)))); %Makes the ...
    calculations faster by removing derivatives.
toc
Z_total=vpa(subs(Z_total,first_derivatives, zeros(size(first_derivatives)))); %Makes the ...
    calculations faster by removing derivatives.
toc
L_total=vpa(subs(L_total,first_derivatives, zeros(size(first_derivatives)))); %Makes the ...
    calculations faster by removing derivatives.
toc
M_total=vpa(subs(M_total,first_derivatives, zeros(size(first_derivatives)))); %Makes the ...
    calculations faster by removing derivatives.
toc
N_total=vpa(subs(N_total,first_derivatives, zeros(size(first_derivatives)))); %Makes the ...
    calculations faster by removing derivatives.
toc
phi_L_dd=vpa(subs(phi_L_dd,first_derivatives, zeros(size(first_derivatives)))); %Makes the ...
    calculations faster by removing derivatives.
th_L_dd=vpa(subs(th_L_dd,first_derivatives, zeros(size(first_derivatives)))); %Makes the ...
    calculations faster by removing derivatives.

% fprintf("X symbols:");symvar(X_total)
% fprintf("Y symbols:");symvar(Y_total)
% fprintf("Z symbols:");symvar(Z_total)
% fprintf("L symbols:");symvar(L_total)
% fprintf("M symbols:");symvar(M_total)
% fprintf("N symbols:");symvar(N_total)
% fprintf("phi_L_dd symbols:");symvar(phi_L_dd)
% fprintf("th_L_dd symbols:");symvar(th_L_dd)
fprintf('External forces generated complete.')

%% Generates EOM

    X_states_studied=[u, v, w, p, q, r, phi_euler, th_euler, psi_euler, phi_L, th_L, phi_L_d, ...
        th_L_d];
    u_control_inputs=[th0, th1s, th1c, th2s, th2c, del_elevator, del_rudder];
    x_EOM=sym('X_EOM',[13 1]);

    x_EOM(1)=subs(x_EOM(1),-(w*q-v*r)+X_total/m_aircraft-g*sin(th_euler));
    x_EOM(2)=subs(x_EOM(2),-(u*r-w*p)+Y_total/m_aircraft-g*cos(th_euler)*sin(phi_euler));
    x_EOM(3)=subs(x_EOM(3),-(v*p-u*q)+Z_total/m_aircraft+g*cos(th_euler)*cos(phi_euler));
    x_EOM(4)=subs(x_EOM(4),I_xx^-1*(L_total+q*r*(I_yy-I_zz)));
    x_EOM(5)=subs(x_EOM(5),I_yy^-1*(M_total+p*r*(I_zz-I_xx)));
    x_EOM(6)=subs(x_EOM(6),I_zz^-1*(N_total+p*q*(I_xx-I_yy)));
    x_EOM(7)=subs(x_EOM(7),p+q*sin(phi_euler)*tan(th_euler)+r*cos(phi_euler)*tan(th_euler));
    x_EOM(8)=subs(x_EOM(8),q*cos(phi_euler)-r*sin(phi_euler));
    x_EOM(9)=subs(x_EOM(9),q*sin(phi_euler)*sec(th_euler)+r*cos(phi_euler)*sec(th_euler));
    x_EOM(10)=subs(x_EOM(10),phi_L_d);
    x_EOM(11)=subs(x_EOM(11),th_L_d);
    x_EOM(12)=subs(x_EOM(12),phi_L_dd);
```

```
    x_EOM(13)=subs(x_EOM(13),th_L_dd);


  fprintf('Equations of motion generated.')
%% Generates equilibrium points

 %% Equilibrium values. INPUTS REQUIRED.
p_dot_e=10^-3;
q_dot_e=10^-3;
r_dot_e=10^-3;

%KNOWN VARIABLES. INPUT THESE 10 variables below.
%Equilibrium states
%u_e=1;
v_e=10^-1;
w_e=10^-1;
p_e=10^-1;
q_e=10^-1;
r_e=10^-1;

phi_euler_e=10^-2; %roll angle
%th_euler_e=10^-2;
psi_euler_e=10^-2; %Yaw angle

phi_L_d_e=10^-3;
th_L_d_e=10^-3;
phi_L_e=10^-1;


%all in radians

% th1s_e=0.05235;
% th1c_e=0;
% th2s_e=-0.05235;
% th2c_e=-0;
%

unknown_states=[th_euler;th0;th1s;th1c;th2s;th2c;del_elevator;del_rudder]; %Need to solve for ...
    this array
    known_states=[u; v; w; p; q; r; phi_euler;psi_euler; phi_L;th_L; phi_L_d; th_L_d];
    fixed_points=[u_e; v_e; w_e; p_e; q_e; r_e; phi_euler_e; psi_euler_e;phi_L_e;th_L_e; ...
        phi_L_d_e; th_L_d_e];

    x_EOM_subs=subs(x_EOM, known_states,fixed_points);%Replaces known states with values.
    toc
syms x1 x2 x3 x4 x5 x6 x7 x8

variables=sym(zeros(13,1));
% for i=1:13
%     symvar(x_EOM_subs(i))
% end
%% non linear solver
%These 2 lines are done so that we know what is produced by the fsolve()
%step.
placeholder=[x1;x2;x3;x4;x5;x6;x7;x8];
x_EOM_subs=subs(x_EOM_subs, unknown_states,placeholder);%Tags each variable.


fprintf("Appropriate form of x_EOM_subs produced, %s ", datestr(now,'HH:MM:SS.FFF'))
```

```

    f = ...
        matlabFunction(x_EOM_subs(1),x_EOM_subs(2),x_EOM_subs(3),x_EOM_subs(4),x_EOM_subs(5),x_EOM_subs(6
        'File','x_EOM_handle', 'Outputs',{'EOM1','EOM2','EOM3','EOM4','EOM5','EOM6','EOM7'});
    toc
    %%
    opt = optimset('Algorithm', 'levenberg-marquardt', 'Display','off');
    fprintf("Appropriate function handle produced, %s ", datestr(now,'HH:MM:SS.FFF'))

    fun=@x_EOM_handle;
    modf = @(x)fun(x(1),x(2), x(3),x(4),x(5),x(6),x(7),x(8));
    initial_points=[0;0;0;0;0;0;0;0];
    solution=fsolve(modf , initial_points, opt);

    th_euler_e=solution(1);
    th0_e=solution(2);
    th1s_e=solution(3);
    th1c_e=solution(4);
    th2s_e=solution(5);
    th2c_e=solution(6);
    del_elevator_e=solution(7);
    del_rudder_e=solution(8);

    %Function outputs the array e_point.
    e_point=[u_e, v_e, w_e, p_e, q_e, r_e, phi_euler_e, th_euler_e, psi_euler_e, phi_L_e, ...
        th_L_e, phi_L_d_e, th_L_d_e, th0_e, th1s_e, th1c_e, th2s_e, th2c_e, del_elevator_e, ...
        del_rudder_e];


    %% Stability derivatives

X_states=[u, v, w, p, q, r, phi_euler, th_euler, psi_euler,phi_L, th_L, phi_L_d, th_L_d, th0, ...
    th1s, th1c, th2s, th2c, del_elevator, del_rudder];
first_derivatives=[u_d,v_d,w_d, p_d, q_d, r_d, phi_euler_d, th_euler_d, psi_euler_d, phi_L_d, ...
    th_L_d, th0_d, th1s_d, th1c_d, th2s_d, th2c_d, del_elevator_d, del_rudder_d];


toc
toc
states_dimension=13;
tolerance=10^-1;
X_e_numerical=zeros(1,states_dimension);
for i=1:states_dimension

    if i>6 && i<12
        continue
    else
        perturbation=zeros(1,size(e_point,2));
        perturbation(1,i)=tolerance;

        X_e=double(subs(X_total,X_states, e_point));
        X_e_perturbed=double(subs(X_total,X_states, e_point+perturbation));

        X_e_numerical(i)=(X_e_perturbed-X_e)/tolerance;
        toc
        fprintf(" Finished %f derivative.", i)
    end

end
```

```

toc
X_vector_e=X_e_numerical;
X_vector_e(1:6)=X_vector_e(1:6)/m_aircraft;
fprintf("X stability derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc

Y_e_numerical=zeros(1,states_dimension);
for i=1:states_dimension

    if i>6 && i<12
        continue
    else
        perturbation=zeros(1,size(e_point,2));
        perturbation(1,i)=tolerance;

        Y_e=double(subs(Y_total,X_states, e_point));
        Y_e_perturbed=double(subs(Y_total,X_states, e_point+perturbation));
        Y_e_numerical(i)=(Y_e_perturbed-Y_e)/tolerance;
        toc
        fprintf(" Finished %f derivative.", i)
    end
end
toc
Y_vector_e=Y_e_numerical;
Y_vector_e(1:6)=Y_vector_e(1:6)/m_aircraft;
fprintf("Y stability derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc

Z_e_numerical=zeros(1,states_dimension);
for i=1:states_dimension

    if i>6 && i<12
        continue
    else
        perturbation=zeros(1,size(e_point,2));
        perturbation(1,i)=tolerance;

        Z_e=double(subs(Z_total,X_states, e_point));
        Z_e_perturbed=double(subs(Z_total,X_states, e_point+perturbation));

        Z_e_numerical(i)=(Z_e_perturbed-Z_e)/tolerance;
        toc
        fprintf(" Finished %f derivative.", i)
    end

end

toc
Z_vector_e=Z_e_numerical;
Z_vector_e(1:6)=Z_vector_e(1:6)/m_aircraft;
fprintf("Z stability derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc

L_e_numerical=zeros(1,states_dimension);
for i=1:states_dimension

    if i>6 && i<12
        continue
```

```
    else
        perturbation=zeros(1,size(e_point,2));
        perturbation(1,i)=tolerance;

        L_e=double(subs(L_total,X_states, e_point));
        L_e_perturbed=double(subs(L_total,X_states, e_point+perturbation));

        L_e_numerical(i)=(L_e_perturbed-L_e)/tolerance;
        toc
        fprintf(" Finished %f derivative.", i)
    end

end

L_vector_e=L_e_numerical;
L_vector_e(1:6)=L_vector_e(1:6)/I_xx;

fprintf("L stability derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc


M_e_numerical=zeros(1,states_dimension);
for i=1:states_dimension

    if i>6 && i<12
        continue
    else
        perturbation=zeros(1,size(e_point,2));
        perturbation(1,i)=tolerance;

        M_e=double(subs(M_total,X_states, e_point));
        M_e_perturbed=double(subs(M_total,X_states, e_point+perturbation));

        M_e_numerical(i)=(M_e_perturbed-M_e)/tolerance;
        toc
        fprintf(" Finished %f derivative.", i)
    end

end

toc
M_vector_e=M_e_numerical;
M_vector_e(1:6)=M_vector_e(1:6)/I_yy;
fprintf("M stability derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc
N_e_numerical=zeros(1,states_dimension);
for i=1:states_dimension

    if i>6 && i<12
        continue
    else
        perturbation=zeros(1,size(e_point,2));
        perturbation(1,i)=tolerance;

        N_e=double(subs(N_total,X_states, e_point));
        N_e_perturbed=double(subs(N_total,X_states, e_point+perturbation));

        N_e_numerical(i)=(N_e_perturbed-N_e)/tolerance;
        toc
```

```
        fprintf(" Finished %f derivative.", i)
    end

end
toc
N_vector_e=N_e_numerical;
N_vector_e(1:6)=N_vector_e(1:6)/I_zz;
fprintf("N stability derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc

phi_L_e_numerical=zeros(1,states_dimension);
for i=1:states_dimension
    perturbation=zeros(1,size(e_point,2));
    perturbation(1,i)=tolerance;

    phi_L_dd_e=double(subs(phi_L_dd,X_states, e_point));
    phi_L_dd_perturbed=double(subs(phi_L_dd,X_states, e_point+perturbation));

    phi_L_e_numerical(i)=(phi_L_dd_perturbed-phi_L_dd_e)/tolerance;
    toc
    fprintf(" Finished %f derivative.", i)


end

phi_L_vector_e=phi_L_e_numerical;

fprintf("phi_L_d stability derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc



th_L_e_numerical=zeros(1,states_dimension);
for i=1:states_dimension


        perturbation=zeros(1,size(e_point,2));
        perturbation(1,i)=tolerance;

        th_L_dd_e=double(subs(th_L_dd,X_states, e_point));
        th_L_dd_perturbed=double(subs(th_L_dd,X_states, e_point+perturbation));

        th_L_e_numerical(i)=(th_L_dd_perturbed-th_L_dd_e)/tolerance;
        toc
        fprintf(" Finished %f derivative.", i)


end

th_L_vector_e=th_L_e_numerical;

fprintf("th_L_d stability derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc
save stability_deriv_part1A.mat X_vector_e Y_vector_e Z_vector_e L_vector_e M_vector_e ...
    N_vector_e phi_L_vector_e th_L_vector_e
toc


%% Control derivatives
```

```

toc
toc
controls_dimension=7;
tolerance=10^-1;

%Row 1
X_e_numerical=zeros(1,controls_dimension);
for i=1:6

    j=13+i;

    perturbation=zeros(1,size(e_point,2));
    perturbation(1,j)=tolerance;

    X_e=double(subs(X_total,X_states, e_point));
    X_e_perturbed=double(subs(X_total,X_states, e_point+perturbation));

    X_e_numerical(i)=(X_e_perturbed-X_e)/tolerance;
    toc
    fprintf("Finished %f X derivative.", i)
end

toc
X_control_vector_e=X_e_numerical;
X_control_vector_e(1:7)=X_control_vector_e(1:7)/m_aircraft;
fprintf("X control derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc


tolerance=10^-1;

%Row 2
Y_e_numerical=zeros(1,controls_dimension);
for i=1:7
    j=13+i;
    if i==6
        continue
    else

        perturbation=zeros(1,size(e_point,2));
        perturbation(1,j)=tolerance;

        Y_e=double(subs(Y_total,X_states, e_point));
        Y_e_perturbed=double(subs(Y_total,X_states, e_point+perturbation));

        Y_e_numerical(i)=(Y_e_perturbed-Y_e)/tolerance;
        toc
        fprintf("Finished %f Y derivative.", i)

    end
end
toc
Y_control_vector_e=Y_e_numerical;
Y_control_vector_e(1:7)=Y_control_vector_e(1:7)/m_aircraft;
fprintf("Y control derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc


```

```

%Row 3
Z_e_numerical=zeros(1,controls_dimension);
for i=1:6
    j=13+i;
    perturbation=zeros(1,size(e_point,2));
    perturbation(1,j)=tolerance;

    Z_e=double(subs(Z_total,X_states, e_point));
    Z_e_perturbed=double(subs(Z_total,X_states, e_point+perturbation));

    Z_e_numerical(i)=(Z_e_perturbed-Z_e)/tolerance;
    toc
    fprintf("Finished %f Z derivative.", i)


end
toc
Z_control_vector_e=Z_e_numerical;
Z_control_vector_e(1:6)=Z_control_vector_e(1:6)/m_aircraft;
fprintf("Z control derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc

L_e_numerical=zeros(1,controls_dimension);
for i=1:7
    j=13+i;
    if i==6
        continue
    else

        perturbation=zeros(1,size(e_point,2));
        perturbation(1,j)=tolerance;

        L_e=double(subs(L_total,X_states, e_point));
        L_e_perturbed=double(subs(L_total,X_states, e_point+perturbation));

        L_e_numerical(i)=(L_e_perturbed-L_e)/tolerance;
        toc
        fprintf("Finished %f L derivative.", i)

    end
end
toc
L_control_vector_e=L_e_numerical;
L_control_vector_e(1:7)=L_control_vector_e(1:7)/I_xx;
fprintf("L control derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc

%Row 5
M_e_numerical=zeros(1,controls_dimension);
for i=1:6
    j=13+i;
    perturbation=zeros(1,size(e_point,2));
    perturbation(1,j)=tolerance;

    M_e=double(subs(M_total,X_states, e_point));
    M_e_perturbed=double(subs(M_total,X_states, e_point+perturbation));

    M_e_numerical(i)=(M_e_perturbed-M_e)/tolerance;
```

```
    toc
    fprintf("Finished %f M derivative.", i)
end
toc
M_control_vector_e=M_e_numerical;
M_control_vector_e(1:6)=M_control_vector_e(1:6)/I_yy;
fprintf("M control derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc

%Row 6
N_e_numerical=zeros(1,controls_dimension);
for i=1:7
    if i==6
        continue
    else
        j=13+i;
        perturbation=zeros(1,size(e_point,2));
        perturbation(1,j)=tolerance;

        N_e=double(subs(N_total,X_states, e_point));
        N_e_perturbed=double(subs(N_total,X_states, e_point+perturbation));

        N_e_numerical(i)=(N_e_perturbed-N_e)/tolerance;
        toc
        fprintf("Finished %f N derivative.", i)

    end
end
toc
N_control_vector_e=N_e_numerical;
N_control_vector_e(1:7)=N_control_vector_e(1:7)/I_zz;
fprintf("N control derivatives produced, %s ", datestr(now,'HH:MM:SS.FFF'))
toc

%%%%%Rows 10 & 11 are zeros
%%%%%%%%%%%%%%%Rows 12 & 13
phi_L_dd_numerical=zeros(1,controls_dimension);
th_L_dd_numerical=zeros(1,controls_dimension);
for i=1:5

    j=13+i;
    perturbation=zeros(1,size(e_point,2));
    perturbation(1,j)=tolerance;

    phi_L_dd_e=double(subs(phi_L_dd,X_states, e_point));
    phi_L_dd_perturbed=double(subs(phi_L_dd,X_states, e_point+perturbation));
    phi_L_dd_numerical(i)=(phi_L_dd_perturbed-phi_L_dd_e)/tolerance;
    toc
    fprintf("Finished %f th_L_dd derivative.", i)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    th_L_dd_e=double(subs(th_L_dd,X_states, e_point));
    th_L_dd_perturbed=double(subs(th_L_dd,X_states, e_point+perturbation));
    th_L_dd_numerical(i)=(th_L_dd_perturbed-th_L_dd_e)/tolerance;
    toc
    fprintf("Finished %f phi_L_dd derivative.", i)
end

phi_L_control_vector_e=phi_L_dd_numerical;
th_L_control_vector_e=th_L_dd_numerical;
```

```
save control_deriv_.mat X_control_vector_e Y_control_vector_e Z_control_vector_e ...
    L_control_vector_e M_control_vector_e N_control_vector_e phi_L_control_vector_e ...
    th_L_control_vector_e

toc



 %% State space formulation, A  Matrix, for the selected equilibrium point

 state_dimension=13;
    A_matrix=zeros(state_dimension);

    A_matrix(1,1)= X_vector_e(1);
    A_matrix(1,2)= X_vector_e(2)+r_e;
    A_matrix(1,3)= X_vector_e(3)-q_e;
    A_matrix(1,4)= X_vector_e(4);
    A_matrix(1,5)= X_vector_e(5)-w_e;
    A_matrix(1,6)= X_vector_e(6)+v_e;
    A_matrix(1,8)= -g *cos(th_euler_e);

    for i=10:state_dimension
        A_matrix(1,i)=X_vector_e(i);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    A_matrix(2,1)= Y_vector_e(1)-r_e;
    A_matrix(2,2)= Y_vector_e(2);
    A_matrix(2,3)= Y_vector_e(3)+p_e;
    A_matrix(2,4)= Y_vector_e(4)+w_e;
    A_matrix(2,5)= Y_vector_e(5);
    A_matrix(2,6)= Y_vector_e(6)-u_e;
    A_matrix(2,7)= g*cos(th_euler_e)*cos(phi_euler_e);
    A_matrix(2,8)= -g *sin(th_euler_e)*sin(phi_euler_e);

    for i=10:state_dimension
        A_matrix(2,i)=Y_vector_e(i);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    A_matrix(3,1)= Z_vector_e(1)+q_e;
    A_matrix(3,2)= Z_vector_e(2)-p_e;
    A_matrix(3,3)= Z_vector_e(3);
    A_matrix(3,4)= Z_vector_e(4)-v_e;
    A_matrix(3,5)= Z_vector_e(5)+u_e;
    A_matrix(3,6)= Z_vector_e(6)-u_e;
    A_matrix(3,7)= g*cos(th_euler_e)*sin(phi_euler_e);
    A_matrix(3,8)= g *sin(th_euler_e)*cos(phi_euler_e);

    for i=10:state_dimension
        A_matrix(3,i)=Z_vector_e(i);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    A_matrix(4,1)= L_vector_e(1);
    A_matrix(4,2)= L_vector_e(2);
    A_matrix(4,3)= L_vector_e(3);
    A_matrix(4,4)= L_vector_e(4);
    A_matrix(4,5)= L_vector_e(5)+r_e*(I_yy-I_zz);
    A_matrix(4,6)= L_vector_e(6)-q_e*(I_yy-I_zz);

    for i=10:state_dimension
```

```
        A_matrix(4,i)=L_vector_e(i);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    A_matrix(5,1)= M_vector_e(1)+r_e*(I_zz-I_xx);
    A_matrix(5,2)= M_vector_e(2);
    A_matrix(5,3)= M_vector_e(3);
    A_matrix(5,4)= M_vector_e(4)+r_e*(I_zz-I_xx);
    A_matrix(5,5)= M_vector_e(5);
    A_matrix(5,6)= M_vector_e(6)+p_e*(I_zz-I_xx);

    for i=10:state_dimension
        A_matrix(5,i)=M_vector_e(i);
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    A_matrix(6,1)= N_vector_e(1)+q_e*(I_xx-I_yy);
    A_matrix(6,2)= N_vector_e(2);
    A_matrix(6,3)= N_vector_e(3);
    A_matrix(6,4)= N_vector_e(4)+q_e*(I_xx-I_yy);
    A_matrix(6,5)= N_vector_e(5)+p_e*(I_xx-I_yy);
    A_matrix(6,6)= N_vector_e(6);

    for i=10:state_dimension
        A_matrix(6,i)=N_vector_e(i);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    A_matrix(7,4)= 1;
    A_matrix(7,5)= tan(th_euler_e)*sin(phi_euler_e);
    A_matrix(7,6)= cos(phi_euler_e)*tan(th_euler_e);
    A_matrix(7,7)= q_e*cos(phi_euler_e)*tan(th_euler_e)-r_e*cos(phi_euler_e)*tan(th_euler_e);
    A_matrix(7,8)= ...
        q_e*sin(phi_euler_e)*(sec(th_euler_e))^2-r_e*cos(phi_euler_e)*(sec(th_euler_e))^2;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    A_matrix(8,5)= cos(phi_euler_e);
    A_matrix(8,6)= -sin(phi_euler_e);
    A_matrix(8,7)= -q_e*sin(phi_euler_e)-r_e*cos(phi_euler_e);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    A_matrix(9,5)= sin(phi_euler_e)*sec(th_euler_e);
    A_matrix(9,6)= cos(phi_euler_e)*sec(th_euler_e);
    A_matrix(9,7)= q_e*cos(phi_euler_e)*sec(th_euler_e)-r_e*sin(phi_euler_e)*sec(th_euler_e);
    A_matrix(9,8)= tan(th_euler_e)*sec(th_euler_e)*(q_e*sin(phi_euler_e)+r_e*cos(phi_euler_e));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for i=1:state_dimension

        A_matrix(12,i)=phi_L_vector_e(i);
        A_matrix(state_dimension,i)=th_L_vector_e(i);
    end
    A_matrix(10,12)=1;
    A_matrix(11,state_dimension)=1;
    %% B matrix formulation, for selected equilibrium point
    B_matrix=zeros(state_dimension,7);
    for i=1:7
        B_matrix(1,i)=X_control_vector_e(i);
        B_matrix(2,i)=Y_control_vector_e(i);
        B_matrix(3,i)=Z_control_vector_e(i);
        B_matrix(4,i)=L_control_vector_e(i);
        B_matrix(5,i)=M_control_vector_e(i);
        B_matrix(6,i)=N_control_vector_e(i);

```

```
        B_matrix(12,i)=phi_L_control_vector_e(i);
        B_matrix(state_dimension,i)=th_L_control_vector_e(i);
    end
    %% Save function
    %Outputs to folder (mission_legs). Remember to properly label the .mat
    %file.
   save state_space_model.mat A_matrix B_matrix e_point;
```

B.2 Eigenvalue analysis

This script conducts eigenvalue analysis, plots figures and performs normalisation.

```
clc
clear

close all

load open_loop_analysis.mat
load e_point_analysis

[hover_vector,hover_values]=eig(A_matrix_hover);
[cruise_vector,cruise_values]=eig(A_matrix_cruise);


%% Normalise eigenvector elements to equilibrium states
hover_vector_magnitude=sqrt(real(hover_vector).^2+imag(hover_vector).^2);
normalised_hover_magnitude=zeros(13);
cruise_vector_magnitude=sqrt(real(cruise_vector).^2+imag(cruise_vector).^2);
normalised_cruise_magnitude=zeros(13);
for i=1:13
    for j=1:13
        normalised_hover_magnitude(i,j)=hover_vector_magnitude(i,j)./e_point_hover(j)';
        normalised_cruise_magnitude(i,j)=cruise_vector_magnitude(i,j)./e_point_cruise(j)';
    end
end


eigenvalues_hover=diag(hover_values);
eigenvalues_cruise=diag(cruise_values);


figure(1)
box on
grid minor
hold on
hold on

xlabel('Re(z)','interpreter', 'latex', 'fontsize', 30, 'Rotation', 0)
ylabel('Im(z)','interpreter', 'latex', 'fontsize', 30, 'Rotation', 0)
set(gca,'FontSize',30)
title('Root locus plot, hover with payload coupled', 'interpreter', 'latex', 'fontsize', 20, ...
    'Rotation', 0)

hover_root_locus=plot(eigenvalues_hover, 'bo', 'MarkerSize', 10);
plot([0 0], [-100 100], 'k') %Im axis
plot([-8 8], [0 0], 'k') %Im axis
L=legend([hover_root_locus],'Hover with payload');
L.FontSize=20;

```

```

figure(2)
box on
grid minor
hold on
hold on

xlabel('Re(z)','interpreter', 'latex', 'fontsize', 30, 'Rotation', 0)
ylabel('Im(z)','interpreter', 'latex', 'fontsize', 30, 'Rotation', 0)
set(gca,'FontSize',30)
title('Root locus plot, cruise with payload, coupled', 'interpreter', 'latex', 'fontsize', 20, ...
    'Rotation', 0)
cruise_root_locus=plot(eigenvalues_cruise, 'ro', 'MarkerSize', 10);
plot([0 0], [-100 100], 'k') %Im axis
plot([-8 8], [0 0], 'k') %Im axis
L=legend([cruise_root_locus], 'Cruise with payload');
L.FontSize=20;

figure(3)
box on
grid minor
hold on
hold on

xlabel('Re(z)','interpreter', 'latex', 'fontsize', 30, 'Rotation', 0)
ylabel('Im(z)','interpreter', 'latex', 'fontsize', 30, 'Rotation', 0)
set(gca,'FontSize',30)
title('Root locus plot, hover and cruise with payload, significant mode changes. coupled', ...
    'interpreter', 'latex', 'fontsize', 20, 'Rotation', 0)

eigenvalues_changed_hover(1)=eigenvalues_hover(6);
eigenvalues_changed_hover(2)=eigenvalues_hover(7);
eigenvalues_changed_hover(3)=eigenvalues_hover(8);


eigenvalues_changed_cruise(1)=eigenvalues_cruise(8);
eigenvalues_changed_cruise(2)=eigenvalues_cruise(9);
eigenvalues_changed_cruise(3)=eigenvalues_cruise(10);

hover_root_locus=plot(eigenvalues_changed_hover, 'bo', 'MarkerSize', 10);
cruise_root_locus=plot(eigenvalues_changed_cruise, 'ro', 'MarkerSize', 10);
plot([0 0], [-100 100], 'k') %Im axis
plot([-2 2], [0 0], 'k') %Im axis
L=legend([hover_root_locus cruise_root_locus],'Hover with payload', 'Cruise with payload');
L.FontSize=20;


[hover_vector_paper, A_str]=sd_round(normalised_hover_magnitude, 2, 1, 1, 0, '', '');
[cruise_vector_paper, A_str]=sd_round(normalised_cruise_magnitude, 2, 1, 1, 0, '', '');

matrix2latexmatrix(hover_vector_paper, 'eig_hover.tex')
matrix2latexmatrix(cruise_vector_paper, 'eig_cruise.tex')
```

B.3 Static stability plots

This script produces the plots of the static stability derivatives with respect to forward velocity.

```
clc
clear
```

```
close all

%%Code written by Andrew Ng, 01180665.
load stability_deriv_variation.mat

subplot(2,2,1) %M_u
box on
grid minor
hold on
hold on
plot(u_array,M_u_array, 'b','LineWidth',2)
plot([5 40], [0 0], 'k', 'LineWidth',2)

set(gca,'FontSize',5)
xlabel('$u,  ms^{-1}$','interpreter', 'latex', 'fontsize', 20, 'Rotation', 0)
ylabel('$M_u, rads^{-1}m$','interpreter', 'latex', 'fontsize', 20, 'Rotation', 90)
title('$M_u$ vs $u$', 'interpreter', 'latex', 'fontsize', 20, 'Rotation', 0)

subplot(2,2,2) %M_q
box on
grid minor
hold on
hold on
plot(u_array,M_q_array, 'b','LineWidth',2)
plot([5 40], [0 0], 'k', 'LineWidth',2)

set(gca,'FontSize',5)
xlabel('$u,  ms^{-1}$','interpreter', 'latex', 'fontsize', 20, 'Rotation', 0)
ylabel('$M_q, s^{-1}$','interpreter', 'latex', 'fontsize', 20, 'Rotation', 90)
title('$M_q$ vs $u$', 'interpreter', 'latex', 'fontsize', 20, 'Rotation', 0)


subplot(2,2,3) %N_r
box on
grid minor
hold on
hold on
plot(u_array,L_p_array, 'b','LineWidth',2)
plot([5 40], [0 0], 'k', 'LineWidth',2)

set(gca,'FontSize',5)
xlabel('$u,  ms^{-1}$','interpreter', 'latex', 'fontsize', 20, 'Rotation', 0)
ylabel('$L_p, s^{-1}$','interpreter', 'latex', 'fontsize', 20, 'Rotation', 90)
title('$L_p$ vs $u$', 'interpreter', 'latex', 'fontsize', 20, 'Rotation', 0)


subplot(2,2,4) %N_r
box on
grid minor
hold on
hold on
plot(u_array,N_r_array, 'b','LineWidth',2)
plot([5 40], [0 0], 'k', 'LineWidth',2)

set(gca,'FontSize',5)
xlabel('$u,  ms^{-1}$','interpreter', 'latex', 'fontsize', 20, 'Rotation', 0)
ylabel('$N_r, s^{-1}$','interpreter', 'latex', 'fontsize', 20, 'Rotation', 90)
title('$N_r$ vs $u$', 'interpreter', 'latex', 'fontsize', 20, 'Rotation', 0)

```

```
sgtitle('Comparison of stability derivatives over changes in forward velocity ...
    u','interpreter', 'latex', 'fontsize', 20, 'Rotation', 0)
```