NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CA4 MA4704 AEROELASTICITY

RISHAV IYER

Chapter 1

P-K Method

Theodorsen

Lift

$$L = \rho\pi b^2\left[U\dot{\alpha} - a\ddot{\alpha} + \ddot{h}\right] + 2\pi\rho b U C(k)\left[U\alpha + \dot{h} - \left(a - \frac{b}{2}\right)\dot{\alpha}\right]$$

Moment

$$M = \rho\pi b^2\left[a\ddot{h} + U\dot{h} + U^2\alpha - \left(a^2 + \frac{b^2}{8}\right)\ddot{\alpha}\right] - \rho\pi b U\left[b - (2a+b)C(k)\right]\left[U\alpha + \dot{h} - \left(a - \frac{b}{2}\right)\dot{\alpha}\right]$$

Complete equations of motion:

$$\begin{bmatrix} m & S \\ S & I_\alpha \end{bmatrix}\begin{Bmatrix} \ddot{h} \\ \ddot{\alpha} \end{Bmatrix} + \begin{bmatrix} K_h & 0 \\ 0 & K_\alpha \end{bmatrix}\begin{Bmatrix} h \\ \alpha \end{Bmatrix} = \begin{Bmatrix} -L \\ M \end{Bmatrix}$$

Substituting equations of lift and moment in:

$$\begin{bmatrix} m & S \\ S & I_\alpha \end{bmatrix}\begin{Bmatrix} \ddot{h} \\ \ddot{\alpha} \end{Bmatrix} + \begin{bmatrix} K_h & 0 \\ 0 & K_\alpha \end{bmatrix}\begin{Bmatrix} h \\ \alpha \end{Bmatrix}$$

$$= \begin{Bmatrix} -\rho\pi b^2\left[U\dot{\alpha} - a\ddot{\alpha} + \ddot{h}\right] - 2\pi\rho b U C(k)\left[U\alpha + \dot{h} - \left(a - \frac{b}{2}\right)\dot{\alpha}\right] \\ \rho\pi b^2\left[a\ddot{h} + U\dot{h} + U^2\alpha - \left(a^2 + \frac{b^2}{8}\right)\ddot{\alpha}\right] - \rho\pi b U\left[b - (2a+b)C(k)\right]\left[U\alpha + \dot{h} - \left(a - \frac{b}{2}\right)\dot{\alpha}\right] \end{Bmatrix}$$

$$\Rightarrow \overbrace{\begin{bmatrix} m & S \\ S & I_\alpha \end{bmatrix}}^{[M]}\begin{bmatrix} \ddot{h} \\ \ddot{\alpha} \end{bmatrix} + \overbrace{\begin{bmatrix} K_h & 0 \\ 0 & K_\alpha \end{bmatrix}}^{[K]}\begin{bmatrix} h \\ \alpha \end{bmatrix} = \rho\pi b^2 \overbrace{\begin{bmatrix} -1 & a \\ a & -\left(a^2 + \frac{b^2}{8}\right) \end{bmatrix}}^{[D]}\begin{bmatrix} \ddot{h} \\ \ddot{\alpha} \end{bmatrix}$$

$$\underbrace{+ \rho\pi b U \begin{Bmatrix} -2C(k) & -b + 2C(k)\left(a - \frac{b}{2}\right) \\ b - \left[b - (2a+b)C(k)\right] & \left[b - (2a+b)C(k)\right]\left(a - \frac{b}{2}\right) \end{Bmatrix}}_{[E(k)]}\begin{bmatrix} \dot{h} \\ \dot{\alpha} \end{bmatrix}$$

$$+\underbrace{\rho\pi b U^2 \begin{Bmatrix} 0 & -2C(k) \\ 0 & b-[b-(2a+b)C(k)] \end{Bmatrix} \begin{bmatrix} h \\ \alpha \end{bmatrix}}_{[F(k)]}$$

General Form:

$$[M]\{\ddot{q}\}+[K]\{q\}=[D]\{\ddot{q}\}+[E(k)]\{\dot{q}\}+[F(k)]\{q\}$$

where $\{q\}=\begin{bmatrix} h \\ \alpha \end{bmatrix}$

$$\Rightarrow [[M]-[D]]\{\ddot{q}\}=[E(k)]\{\dot{q}\}+[[F(k)]-[K]]\{q\}$$

Pre multiplying $[[M]-[D]]^{-1}$ throughout:

$$\{\ddot{q}\}=[[M]-[D]]^{-1}[E(k)]\{\dot{q}\}+[[M]-[D]]^{-1}[[F(k)]-[K]]\{q\}$$

$$\Rightarrow \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}\begin{Bmatrix} \dot{q} \\ \ddot{q} \end{Bmatrix}=\begin{bmatrix} 0 & I \\ [[M]-[D]]^{-1}[[F(k)]-[K]] & [[M]-[D]]^{-1}[E(k)] \end{bmatrix}\begin{Bmatrix} q \\ \dot{q} \end{Bmatrix}$$

$$\Rightarrow \{\dot{z}\}=[A(k)]\{z\} \quad \text{where} \quad \{z\}=\begin{Bmatrix} q \\ \dot{q} \end{Bmatrix}$$

$$\Rightarrow [[A(k)]-\lambda[I]]\{z_0\}=\{0\}$$

Where $C(k)$ is the Theodorsen Function with $C(k)=\frac{K_1(ik)}{K_0(ik)+K_1(ik)}$

The approximate expression for the Theodorsen Function used is

$$C(k)=1-\frac{0.165}{1-\left(\frac{0.0455}{k}\right)i}-\frac{0.335}{1-\left(\frac{0.30}{k}\right)i}$$

This becomes an eigenvalue problem which can be solved in MATLAB to obtain the frequencies and damping ratio for each airspeed U.

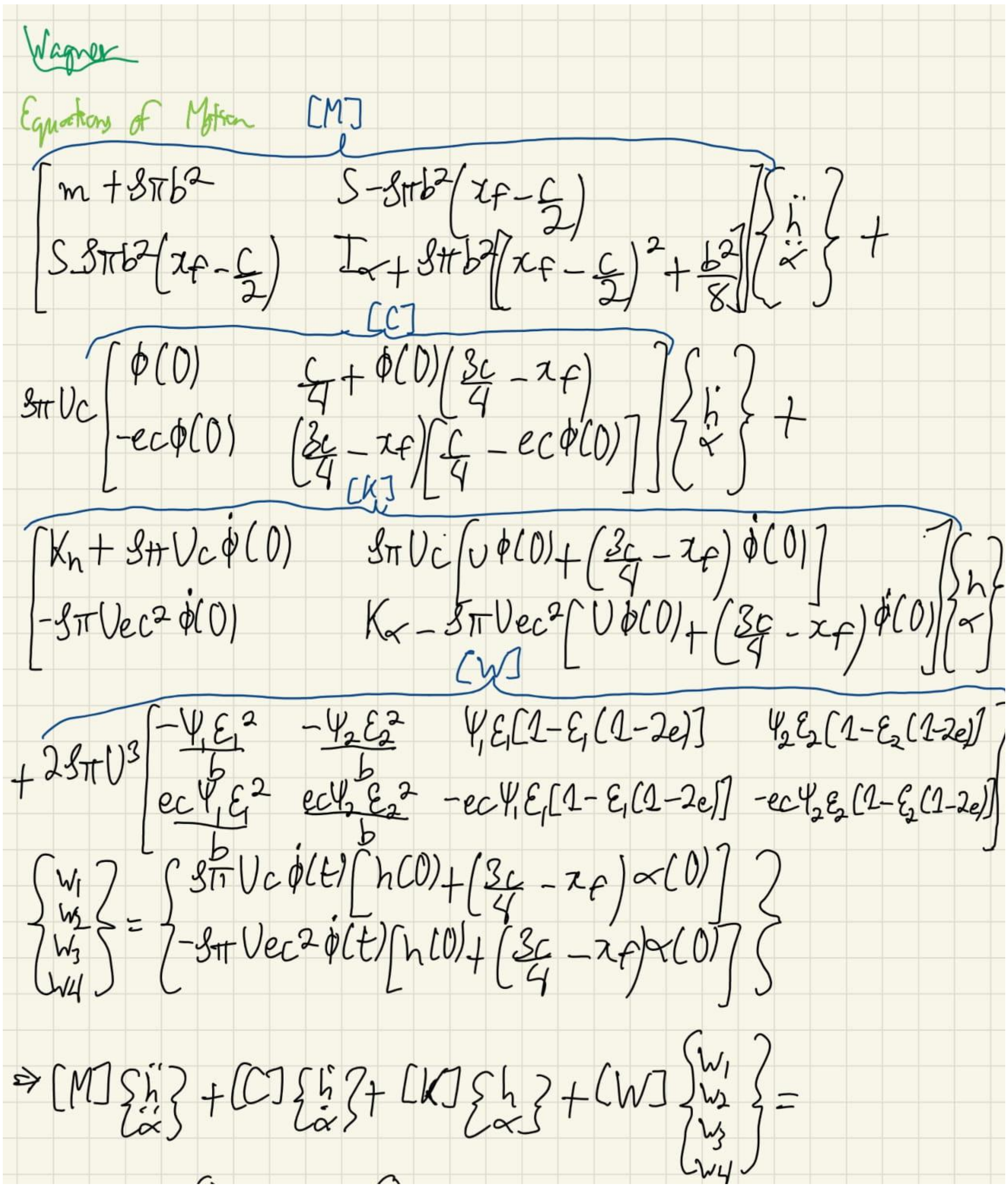
The eigenvalues are complex numbers of the form $\lambda = a + i\,b$.

Frequencies are obtained using $\lambda=\sqrt{a^2+b^2}$.

Damping ratios are obtained using $-\frac{Re(\lambda)}{\omega_n}=\frac{\zeta\omega_n}{\omega_n}=\zeta$

Flutter occurs at the speed where the damping ratio of either degree of freedom (pitch and plunge) becomes zero and then becomes negative.

Wagner Method



Wagner

Equations of Motion

$$\overbrace{\begin{bmatrix} m+\rho\pi b^2 & S-\rho\pi b^2\left(x_f-\frac{c}{2}\right) \\ S-\rho\pi b^2\left(x_f-\frac{c}{2}\right) & I_\alpha+\rho\pi b^2\left[\left(x_f-\frac{c}{2}\right)^2+\frac{b^2}{8}\right] \end{bmatrix}}^{[M]} \begin{Bmatrix} \ddot{h} \\ \ddot{\alpha} \end{Bmatrix} +$$

$$\overbrace{\rho\pi Vc \begin{bmatrix} \phi(0) & \frac{c}{4}+\phi(0)\left(\frac{3c}{4}-x_f\right) \\ -ec\phi(0) & \left(\frac{3c}{4}-x_f\right)\left[\frac{c}{4}-ec\phi(0)\right] \end{bmatrix}}^{[C]} \begin{Bmatrix} \dot{h} \\ \dot{\alpha} \end{Bmatrix} +$$

$$\overbrace{\begin{bmatrix} K_h+\rho\pi Vc\dot{\phi}(0) & \rho\pi Vc\left[V\phi(0)+\left(\frac{3c}{4}-x_f\right)\dot{\phi}(0)\right] \\ -\rho\pi Vec^2\dot{\phi}(0) & K_\alpha-\rho\pi Vec^2\left[V\phi(0)+\left(\frac{3c}{4}-x_f\right)\dot{\phi}(0)\right] \end{bmatrix}}^{[K]} \begin{Bmatrix} h \\ \alpha \end{Bmatrix}$$

$$+\overbrace{2\rho\pi V^3 \begin{bmatrix} -\frac{\Psi_1\varepsilon_1^2}{b} & -\frac{\Psi_2\varepsilon_2^2}{b} & \Psi_1\varepsilon_1[1-\varepsilon_1(1-2e)] & \Psi_2\varepsilon_2[1-\varepsilon_2(1-2e)] \\ \frac{ec\Psi_1\varepsilon_1^2}{b} & \frac{ec\Psi_2\varepsilon_2^2}{b} & -ec\Psi_1\varepsilon_1[1-\varepsilon_1(1-2e)] & -ec\Psi_2\varepsilon_2[1-\varepsilon_2(1-2e)] \end{bmatrix}}^{[W]}$$

$$\begin{Bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{Bmatrix} = \begin{Bmatrix} \rho\pi Vc\dot{\phi}(t)\left[h(0)+\left(\frac{3c}{4}-x_f\right)\alpha(0)\right] \\ -\rho\pi Vec^2\dot{\phi}(t)\left[h(0)+\left(\frac{3c}{4}-x_f\right)\alpha(0)\right] \end{Bmatrix}$$

$$\Rightarrow [M]\begin{Bmatrix} \ddot{h} \\ \ddot{\alpha} \end{Bmatrix}+[C]\begin{Bmatrix} \dot{h} \\ \dot{\alpha} \end{Bmatrix}+[K]\begin{Bmatrix} h \\ \alpha \end{Bmatrix}+[W]\begin{Bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{Bmatrix}=$$

$$\begin{Bmatrix} \rho\pi U c \dot{\phi}(t)\left[h(0)+\left(\frac{3c}{4}-x_f\right)\alpha(0)\right] \\ -\rho\pi U e c^2 \dot{\phi}(t)\left[h(0)+\left(\frac{3c}{4}-x_f\right)\alpha(0)\right] \end{Bmatrix}$$

Premultiplying by $[M]^{-1}$

$$\begin{Bmatrix} \ddot{h} \\ \ddot{\alpha} \end{Bmatrix} + [M]^{-1}[C]\begin{Bmatrix} \dot{h} \\ \dot{\alpha} \end{Bmatrix} + [M]^{-1}[K]\begin{Bmatrix} h \\ \alpha \end{Bmatrix} + [M]^{-1}[W]\begin{Bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{Bmatrix}$$

$$= [M]^{-1}\begin{Bmatrix} \rho\pi U c \dot{\phi}(t)\left[h(0)+\left(\frac{3c}{4}-x_f\right)\alpha(0)\right] \\ -\rho\pi U e c^2 \dot{\phi}(t)\left[h(0)+\left(\frac{3c}{4}-x_f\right)\alpha(0)\right] \end{Bmatrix} \qquad - (1)$$

Now,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} \dot{w}_1 \\ \dot{w}_2 \\ \dot{w}_3 \\ \dot{w}_4 \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}\begin{Bmatrix} h \\ \alpha \end{Bmatrix} + \begin{bmatrix} -\varepsilon_1 U/b & 0 & 0 & 0 \\ 0 & -\varepsilon_2 U/b & 0 & 0 \\ 0 & 0 & -\varepsilon_1 U/b & 0 \\ 0 & 0 & 0 & -\varepsilon_2 U/b \end{bmatrix}$$

$$\begin{Bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{Bmatrix}$$

$$\Rightarrow [I]\begin{Bmatrix} \dot{w}_1 \\ \dot{w}_2 \\ \dot{w}_3 \\ \dot{w}_4 \end{Bmatrix} = [B]\begin{Bmatrix} h \\ \alpha \end{Bmatrix} + [G]\begin{Bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{Bmatrix} \qquad - (2)$$

Additionally: $[I]\begin{Bmatrix}\dot{h}\\ \dot{\alpha}\end{Bmatrix} = [I]\begin{Bmatrix}\dot{h}\\ \dot{\alpha}\end{Bmatrix}$ — $\textcircled{3}$

Assembling the matrix

$$\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{Bmatrix} \ddot{h} \\ \ddot{\alpha} \\ \dot{h} \\ \dot{\alpha} \\ \dot{w}_1 \\ \dot{w}_2 \\ \dot{w}_3 \\ \dot{w}_4 \end{Bmatrix} = \left[\begin{array}{c:c:c} -[M]^{-1}[C] & -[M]^{-1}[K] & -[M]^{-1}[W] \\ \hdashline I & 0 & 0 \\ \hdashline 0 & B & G \end{array}\right] \begin{Bmatrix} \dot{h} \\ \dot{\alpha} \\ h \\ \alpha \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{Bmatrix} + \begin{bmatrix} [M]^{-1} \begin{Bmatrix} \rho\pi V c\,\dot{\phi}(t)\left[h(0) + \left(\frac{3c}{4} - x_f\right)\alpha(0)\right] \\ -\rho\pi V e c^2 \dot{\phi}(t)\left[h(0) + \left(\frac{3c}{4} - x_f\right)\alpha(0)\right] \end{Bmatrix} \\ 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow \{\dot{z}\} = [Q]\{z\} + [D]$$

This becomes $[[Q] - \lambda[I]]\{z\} = \{0\}$, which is an eigenvalue problem which can be solved in MATLAB to obtain the frequencies and damping ratio for each airspeed U.

The eigenvalues are complex numbers of the form $\lambda = a + i\,b$.

Frequencies are obtained using $\lambda = \sqrt{a^2 + b^2}$.

Damping ratios are obtained using $-\frac{Re(\lambda)}{\omega_n} = \frac{\zeta\omega_n}{\omega_n} = \zeta$

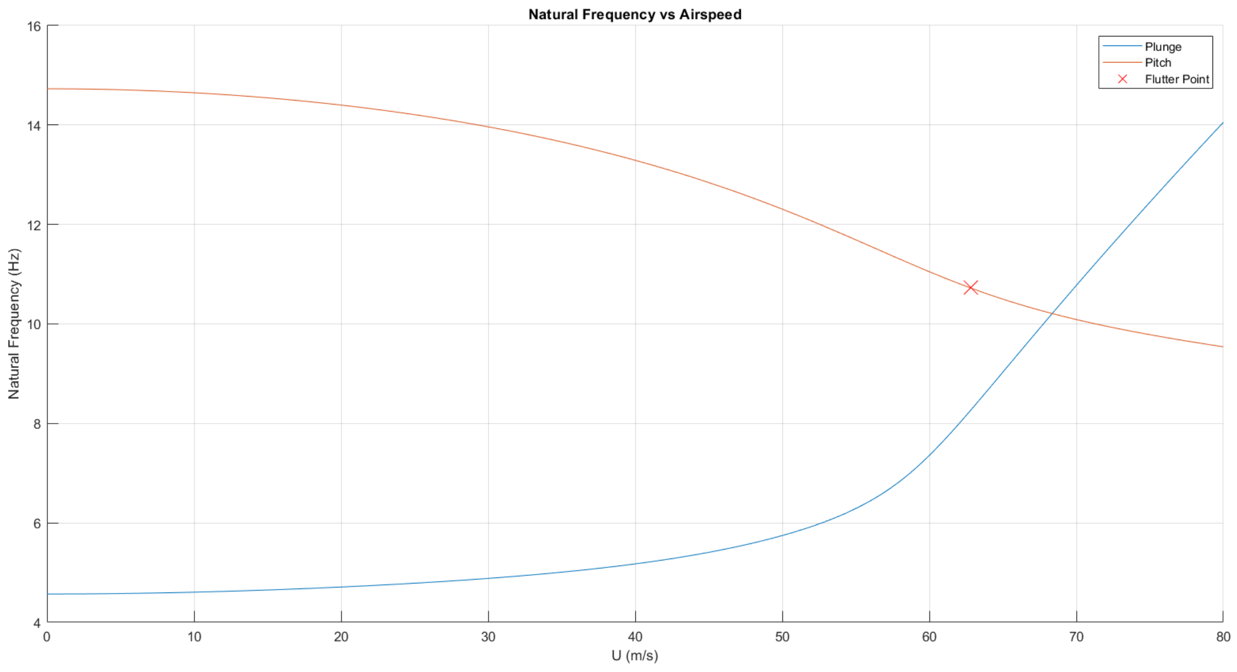
Flutter occurs at the speed where the damping ratio of either degree of freedom (pitch and plunge) becomes zero and then goes to negative.

Chapter 2

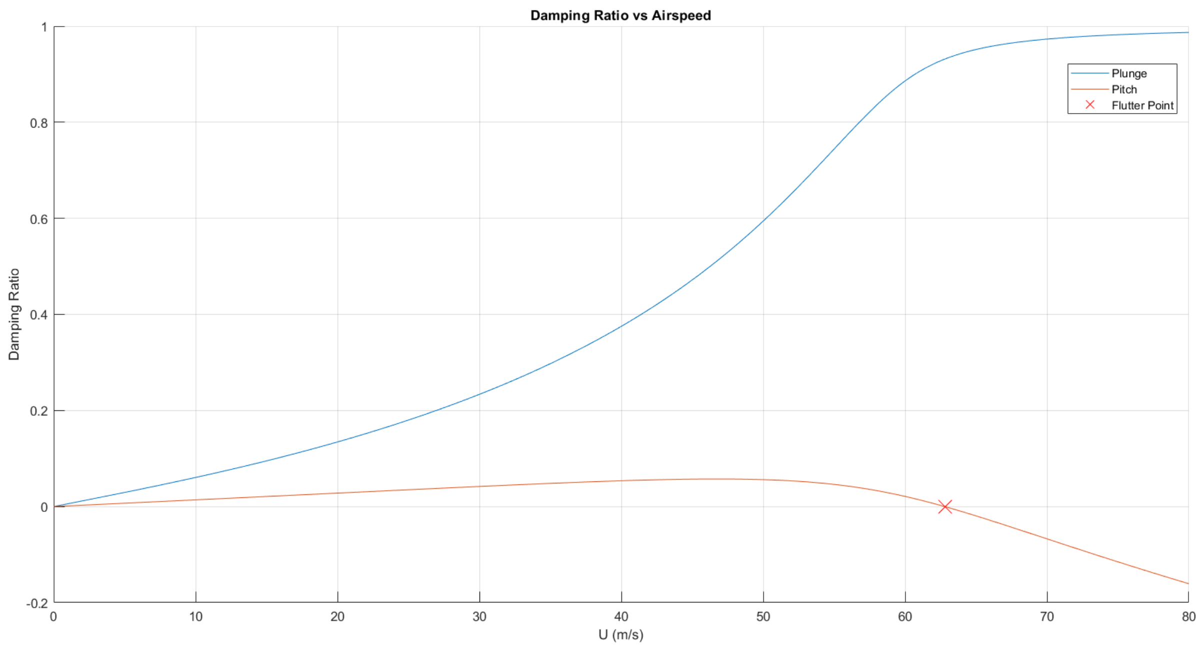
The flutter speed and frequency using the P-K Method is tabulated below.

Flutter Speed (m/s)	62.8
Flutter Frequency (Hz)	10.725

The figure below shows the plot between the natural frequencies and the airspeed.



The figure below shows the plot between the damping ratios and the airspeed.

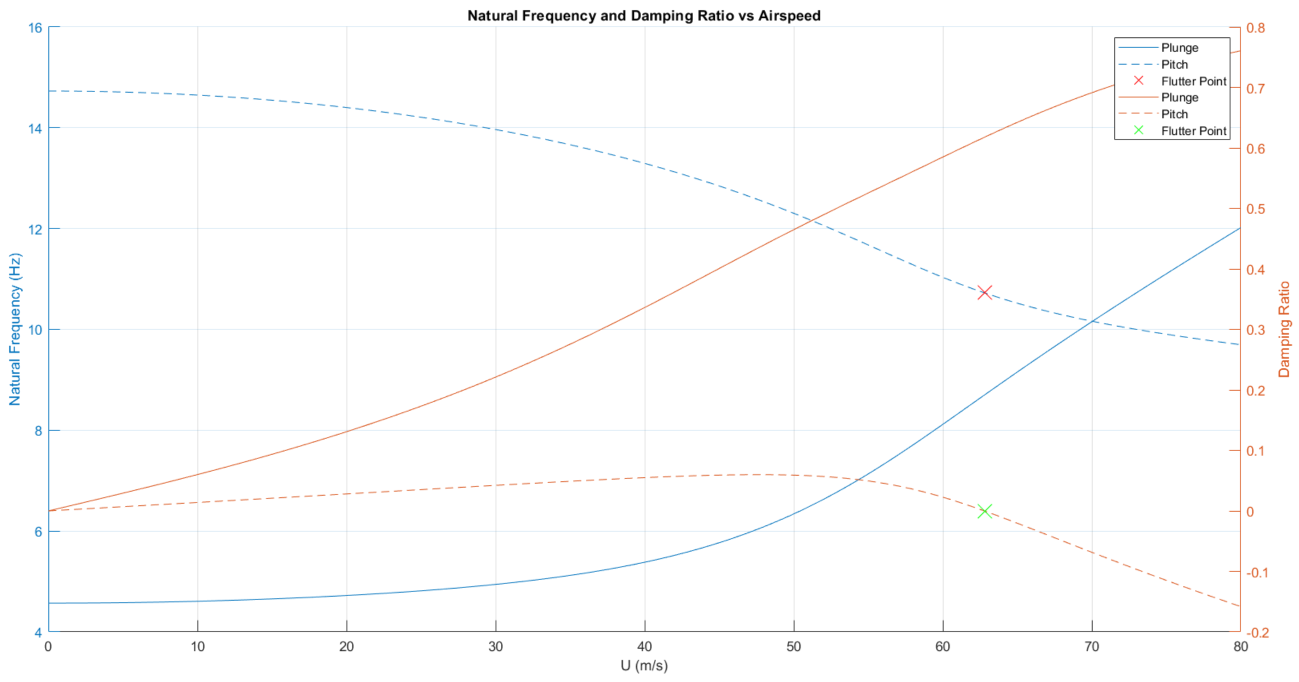

Damping Ratio vs Airspeed
Damping Ratio
U (m/s)
Plunge
Pitch
Flutter Point
1
0.8
0.6
0.4
0.2
0
-0.2
0
10
20
30
40
50
60
70
80

Chapter 3

The flutter speed using Wagner's method is given below.

Flutter Speed (m/s)	62.8
Flutter Frequency (Hz)	10.725

The figure below shows the plot of frequency and damping against airspeed.



Chapter 4

Observations from Task 2 (Chapter 2 – P-K Method)

The pitch frequency follows a steady decline for increasing airspeed U. The plunge frequency increases slightly between $0 \leq U \leq 50\ m/s$ and has a higher increase for $U > 50\ m/s$. The pitch and plunge frequencies approach each other as the airspeed increases.

The plunge damping ratio (ζ) has a steep increase between $0 \leq U \leq 60\ m/s$, and a slower increase for $U > 60\ m/s$ where ζ almost converges to 1. The pitch damping ratio slightly increases between $0 \leq U \leq 47\ m/s$. It decreases until it becomes zero and then becomes negative, at which flutter occurs.

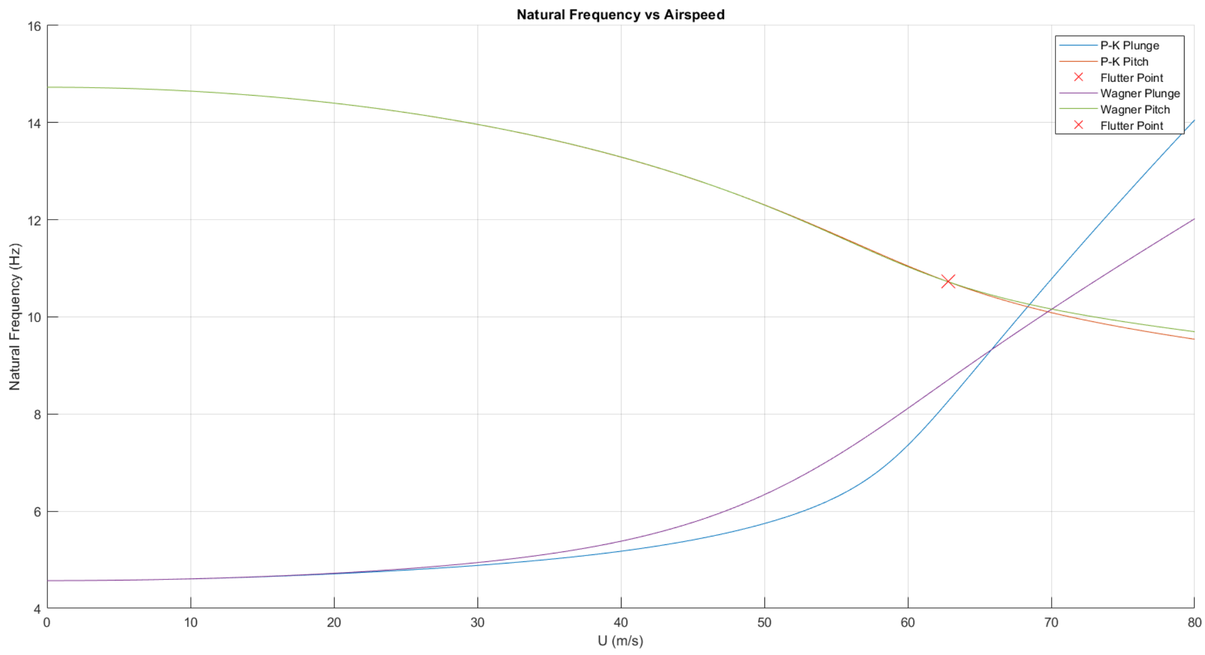
At this speed, the amplitude of the oscillations of the airfoil increase over time.

Observations from Task 3 (Chapter 3 – Wagner's Method)

Overall, the Wagner method natural frequencies and damping ratios follow a very similar trend as the P-K method natural frequencies and damping ratios.

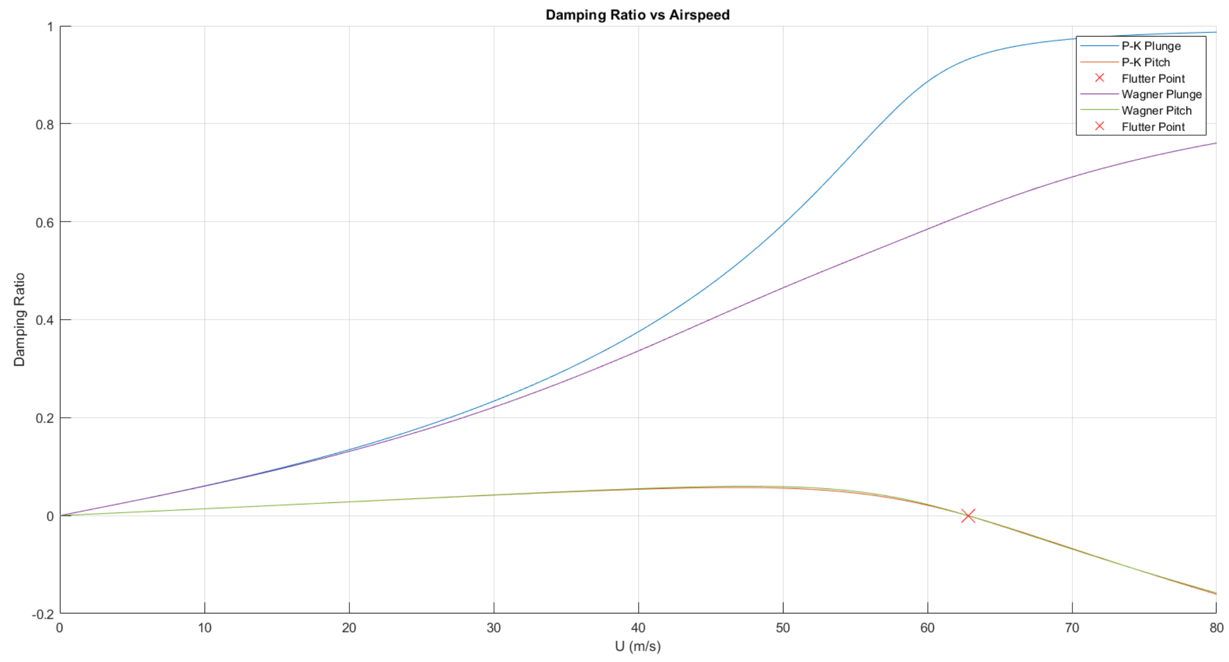
The plunge and pitch frequency have similar natural frequencies for airspeed $U = 0$ compared to the ones obtained from P-K method. The pitch frequency has a similar steady decline as the P-K method until $U = 62.8\ m/s$ after which there is a slight difference between the frequencies from both methods, with the Wagner frequencies being higher.

The plunge frequency overlaps with the P-K method plunge frequency plot for $0 \leq U \leq 20\ m/s$, after which the Wagner method plunge frequency increases and intersects with the P-K method plunge frequency plot at U = $65.9\ m/s$. After this point, it increases while still being slower than the corresponding P-K method frequency.



The Wagner method pitch damping ratios and P-K method pitch damping ratios are similar throughout.

The Wagner method plunge damping ratios and P-K method plunge damping ratios are similar for $0 \leq U \leq 17\ m/s$ after which the P-K method plunge damping ratios are higher than the ones from Wagner's method.



Discussion

The U and U^2 terms add damping and stiffness in the expressions of both Theodorsen and Wagner, making plunge stronger and pitch weaker, which causes the pitch frequency to decrease steadily and the plunge frequency to increase slightly and then rapidly. The frequencies then coalesce with increasing U.

The plunge damping ratio increases with U as the aerodynamic forces provide damping to the motion of the airfoil. As the pitch and plunge modes start to couple, the aerodynamic forces overcome the structural damping for the pitch dominated mode. Thus, the pitch damping ratio decreases, crosses zero and becomes negative, where flutter occurs. At the flutter speed, the airfoil will have unstable oscillations, the amplitude of which increases over time.

P-K Method (Theodorsen) and Wagner Method give the same results for flutter speed and flutter frequency. There are minor numeric differences in the flutter values and damping ratios due to their different unsteady aerodynamic representations, where P-K method is a frequency domain method, and Wagner's method is a time domain method. This shifts their phases and the aerodynamic stiffness and damping slightly.

Chapter 5

Conclusion

The plunge and the pitch modes of the airfoil show increasing aerodynamic coupling as airspeed U increases. The P-K Method and Wagner method both evaluate a similar physical phenomenon with small numerical differences which are due to their different representations, with P-K method (Theodorsen) being in the frequency domain and Wagner method being in the time domain.

The modes of the airfoil are representative of flutter phenomena in aircraft wings, where as the speed increases, aerodynamic loads start to interact dynamically with the structures, with transfer of energy between bending and torsional modes. At flutter, the aerodynamic forces overcome the damping of the system, which causes oscillations of increasing amplitude which can lead to structural failure.

Finally, the P-K method gives slightly more accurate results, whereas the Wagner method is computationally simpler and approximate the aerodynamic time lag with exponential terms.

Chapter 6

Declaration of Honesty

I did this assignment by myself and referred to the code provided by the Professor in class.

Appendix

P-K Method Code

```
close all;
clear all;

mu = 500;  % airfoil density (kg/m3)
c = 1;  % chord length (m)
x_f = 0.45 * c; % Location of elastical axis measured from leading edge of airfoil
a = -0.05 * c;  % Location of elastic axis measured from center of airfoil
t = 0.01;   % Airfoil thickness (m)
b = c / 2;  % Airfoil mid chord (m)



omega_h = 5 * 2 * pi;    % Plunge natural frequency (rad/s)
omega_alpha = 15 * 2 * pi;   % Pitch natural frequency (rad/s)

m = mu * c * t; % Mass per unit span of airfoil (kg/m)

S = -m * a;

K_h = m * (omega_h ^ 2);    % Plunge stiffness

I_c = (1 / 12) * m * (c ^ 2);   % Moment of inertia of airfoil about center of
airfoil
I_alpha = I_c + (m * (a ^ 2));  % Parallel axis theorem to obtain moment of
inertia about elastic axis

K_alpha = I_alpha * (omega_alpha ^ 2);  % Pitch stiffness

rho = 1.225; % Density of air (kg/m3)

V1 = [];    %Heave eigenvalue
d1 = [];    % Heave damping
V2 = [];    % Pitch eigenvalue
d2 = [];    % Pitch damping
d_all = [];

U_min = 0;  % (m/s)
U_max = 80; % (m/s)
dU = 0.1; % (m/s)

tolerance = 1e-3;


U_range = U_min:dU:U_max;   % Range of velocity values from 0 to 80 m/s in steps
of 0.1 m/s


for U = U_range

    ii = 3; % Eigenvalue index to extract

    for omega = [sqrt(K_h / m) sqrt(K_alpha / I_alpha)] % Initial estimate of
flutter frequency
```

```
        error = 10; % Arbitrary number to determine convergence to stop p-k method
iteration

        while error > tolerance     % While loop for eigenvalue iteration, stops
when eigenvalue converges

            k = (omega * b) / U;
            C_k = 1 - (0.165 ./ (1 - 1i * (0.0455 ./ k))) - (0.335 ./ (1 - 1i *
(0.30 ./ k)));  % Reduced frequency k using Theodorsen Function approximation

            M_matrix = [m, S; S, I_alpha];
            K_matrix = [K_h, 0; 0, K_alpha];
            D_matrix = rho * pi * (b ^ 2) * [-1, a; a, -((a ^ 2) + ((b ^ 2) /
8))];
            E_k_matrix = rho * pi * b * U * [-2 * C_k, -b + (2 * C_k * (a - (b /
2))); b - (b - (2 * a + b) * C_k), (b - (2 * a + b) * C_k) * (a - (b / 2))];
            F_k_matrix = rho * pi * b * (U ^ 2) * [0, -2 * C_k; 0, b - (b - (2 * a
+ b) * C_k)];

            A_k = [zeros(2), eye(2); (M_matrix - D_matrix) \ (F_k_matrix -
K_matrix), (M_matrix - D_matrix) \ E_k_matrix];

            Eigenvalues = eig(A_k); % Eigenvalue analysis

            [Eigenvalues_Imag, Index] = sort(imag(Eigenvalues), 'ascend');  %
Arranging the eigenvalues in ascending order by their imaginary parts

            Eigenvalue_Imag_1 = Eigenvalues_Imag(ii);   % Extracting imaginary
part of largest/2nd largest eigenvalue for checking against initial assumed
frequency (omega)
            error = abs(Eigenvalue_Imag_1 - omega);     % Calculate error between
imaginary part of eigenvalue against initial assumed frequency (omega)
            omega = Eigenvalue_Imag_1;  % Initial assumed frequency is updated to
the new eigenvalue imaginary part

        end

        Eigenvalues_sorted = Eigenvalues(Index);

        % Section to obtain and store frequencies and damping
        if ii == 3

            V1 = [V1 abs(Eigenvalues_sorted(ii))];
            d1 = [d1 -1 * real(Eigenvalues_sorted(ii)) /
abs(Eigenvalues_sorted(ii))];

        end

        if ii == 4

            V2 = [V2 abs(Eigenvalues_sorted(ii))];

            d2 = [d2 -1 * real(Eigenvalues_sorted(ii)) /
abs(Eigenvalues_sorted(ii))];


        end

        d_all = [d_all Eigenvalues_sorted];
```

```matlab
        ii = ii + 1;

    end

end

% Loop to find the flutter speed and frequency at the point where damping
% goes negative
for i = 1:1:length(U_range)

    if d1(i)<0
        Flutter_speed = U_range(i);
        Flutter_frequency = V1(i)/2/pi;
        Flutter_damping = d1(i);
        break;
    end

    if d2(i)<0
        Flutter_speed = U_range(i);
        Flutter_frequency = V2(i)/2/pi;
        Flutter_damping = d2(i);
        break;
    end

end

fprintf("Flutter Speed: %.1f m/s \n", Flutter_speed);
fprintf("Flutter Frequency: %.3f Hz \n", Flutter_frequency);

figure(1)
title("Natural Frequency vs Airspeed")
xlabel("U (m/s)")
ylabel("Natural Frequency (Hz)")
hold on
plot(U_range,V1/2/pi,"DisplayName","Plunge");
plot(U_range,V2/2/pi,"DisplayName","Pitch");
plot(Flutter_speed, Flutter_frequency,"x","DisplayName", "Flutter Point", "Color",
"red", "MarkerSize", 15)
legend show
hold off
grid on

figure(2)
title("Damping Ratio vs Airspeed")
xlabel("U (m/s)")
ylabel("Damping Ratio")
hold on
plot(U_range,d1,"DisplayName","Plunge");
plot(U_range,d2,"DisplayName","Pitch");
plot(Flutter_speed, Flutter_damping,"x","DisplayName", "Flutter Point", "Color",
"red", "MarkerSize", 15)
legend show
hold off
grid on
```

Wagner Method Code

```
close all;
clear all;

syms tt

mu = 500;  % airfoil density (kg/m3)
c = 1;  % chord length (m)
x_f = 0.45 * c; % Location of elastical axis measured from leading edge of airfoil
a = -0.05 * c;  % Location of elastic axis measured from center of airfoil
t = 0.01;   % Airfoil thickness (m)
b = c / 2;  % Airfoil mid chord (m)
e = x_f - 0.25 * c;   % Elastic axis distance fraction from aerodynamic centre

omega_h = 5 * 2 * pi;    % Plunge natural frequency (rad/s)
omega_alpha = 15 * 2 * pi;   % Pitch natural frequency (rad/s)

m = mu * c * t; % Mass per unit span of airfoil (kg/m)

S = -m * a;

K_h = m * (omega_h ^ 2);    % Plunge stiffness

I_c = (1 / 12) * m * (c ^ 2);   % Moment of inertia of airfoil about center of
airfoil
I_alpha = I_c + (m * (a ^ 2));  % Parallel axis theorem to obtain moment of
inertia about elastic axis

K_alpha = I_alpha * (omega_alpha ^ 2);  % Pitch stiffness

rho = 1.225; % Density of air (kg/m3)

% Wagner function parameters
psi_1 = 0.165;
psi_2 = 0.335;
epsilon_1 = 0.0455;
epsilon_2 = 0.3;

U_min = 0;  % (m/s)
U_max = 80; % (m/s)
dU = 0.1; % (m/s)

U_range = U_min:dU:U_max;   % Range of velocity values from 0 to 80 m/s in steps
of 0.1 m/s

EV = [];
Frequency = [];
Damping_Ratio = [];

Q = zeros(8);

for U = U_range

    phi = 1 - (psi_1 * exp((-epsilon_1 * U * tt) / b)) - (psi_2 * exp((-epsilon_2
* U * tt) / b));    % Wagner Function
    phi_dot = diff(phi, tt);

    phi_0 = subs(phi, tt, 0);
    phi_dot_0 = subs(phi_dot, tt, 0);

    M = [m + (rho * pi * (b ^ 2)), S - (rho * pi * (b ^ 2) * (x_f - (c / 2))); S - 
(rho * pi * (b ^ 2) * (x_f - (c / 2))), I_alpha + (rho * pi * (b ^ 2)) * (((x_f - 
(c/2)) ^ 2) + ((b ^ 2) / 8))];
    C = rho * pi * U * c * [phi_0, (c / 4) + phi_0 * (0.75 * c - x_f); -e * c * 
phi_0, (0.75 * c - x_f) * (0.25 * c - (e * c * phi_0))];
    K = [K_h + (rho * pi * U * c * phi_dot_0), rho * pi * U * c * ((U * phi_0) + 
(0.75 * c - x_f) * phi_dot_0); -rho * pi * U * e * (c ^ 2) * phi_dot_0, K_alpha - 
(rho * pi * U * e * (c ^ 2)) * ((U * phi_0) + (0.75 * c - x_f) * phi_dot_0)];
    W = 2 * rho * pi * (U ^ 3) *[-psi_1 * (epsilon_1) ^ 2 / b, -psi_2 * 
(epsilon_2) ^ 2 / b, psi_1 * epsilon_1 * (1 - epsilon_1 * (1 - 2 * e)), psi_2 * 
epsilon_2 * (1 - epsilon_2 * (1 - 2 * e)); e * c * psi_1 * (epsilon_1) ^ 2 / b, e 
* c * psi_2 * (epsilon_2) ^ 2 / b, -e * c * psi_1 * epsilon_1 * (1 - epsilon_1 * 
(1 - 2 * e)), -e * c * psi_2 * epsilon_2 * (1 - epsilon_2 * (1 - 2 * e))];

    B = [1 0; 1 0; 0 1; 0 1];
    G = [-epsilon_1 * U / b, 0, 0, 0; 0, -epsilon_2 * U / b, 0, 0; 0, 0, -
epsilon_1 * U / b, 0; 0, 0, 0, -epsilon_2 * U / b];

    Q(1:2,1:2) = -M \ C;
    Q(1:2,3:4) = -M \ K;
    Q(1:2,5:8) = -M \ W;
    Q(3:4,1:2) = eye(2);
    Q(5:8,3:4) = B;
    Q(5:8,5:8) = G;

    Eigenvalues = eig(Q);   % Eigenvalue analysis
    [Eigenvalues_imag, Index] = sort(imag(Eigenvalues), 'ascend');
    Eigenvalues_sorted = Eigenvalues(Index, :);

    EV = [EV; Eigenvalues_sorted(7) Eigenvalues_sorted(8)];

end

found = false;

tolerance = 1e-6;

for i = 1:1:size(U_range,2)


    Frequency(i,1) = abs(EV(i,1));
    Frequency(i,2) = abs(EV(i,2));
    Damping_Ratio(i,1) = -real(EV(i,1))/abs(EV(i,1));
    Damping_Ratio(i,2) = -real(EV(i,2))/abs(EV(i,2));

    if Damping_Ratio(i,1)<0 && found == false && abs(Damping_Ratio(i,1)) > 
tolerance
        Flutter_speed = U_range(i);
        Flutter_frequency = Frequency(i,1)/2/pi;
        Flutter_damping = Damping_Ratio(i,1);
        found = true;
    end

    if Damping_Ratio(i,2)<0 && found == false && abs(Damping_Ratio(i,2)) > 
tolerance
        Flutter_speed = U_range(i);
        Flutter_frequency = Frequency(i,2)/2/pi;
        Flutter_damping = Damping_Ratio(i,2);
        found = true;
    end

end

fprintf("Flutter Speed: %.1f m/s \n", Flutter_speed);
fprintf("Flutter Frequency: %.3f Hz \n", Flutter_frequency);

figure(1)
title("Natural Frequency vs Airspeed")
xlabel("U (m/s)")
ylabel("Natural Frequency (Hz)")
hold on
plot(U_range, Frequency(:,1)/2/pi,"DisplayName","Wagner Plunge");
plot(U_range,Frequency(:,2)/2/pi,"DisplayName","Wagner Pitch");
plot(Flutter_speed, Flutter_frequency,"x","DisplayName", "Flutter Point", "Color",
"red", "MarkerSize", 15)
legend show
hold off
grid on

figure(2)
title("Damping Ratio vs Airspeed")
xlabel("U (m/s)")
ylabel("Damping Ratio")
hold on
plot(U_range,Damping_Ratio(:,1),"DisplayName","Wagner Plunge");
plot(U_range,Damping_Ratio(:,2),"DisplayName","Wagner Pitch");
plot(Flutter_speed, Flutter_damping,"x","DisplayName", "Flutter Point", "Color",
"red", "MarkerSize", 15)
legend show
hold off
grid on

figure(3)
title("Natural Frequency and Damping Ratio vs Airspeed")
xlabel("U (m/s)")
hold on
yyaxis left
ylabel("Natural Frequency (Hz)")
plot(U_range,Frequency(:,1)/2/pi,"DisplayName","Plunge");
plot(U_range,Frequency(:,2)/2/pi,"DisplayName","Pitch");
plot(Flutter_speed, Flutter_frequency,"x","DisplayName", "Flutter Point", "Color",
"red", "MarkerSize", 15)
yyaxis right
ylabel("Damping Ratio")
plot(U_range,Damping_Ratio(:,1),"DisplayName","Plunge");
plot(U_range,Damping_Ratio(:,2),"DisplayName","Pitch");
plot(Flutter_speed, Flutter_damping,"x","DisplayName", "Flutter Point", "Color",
"green", "MarkerSize", 15)
legend show
hold off
grid on
```