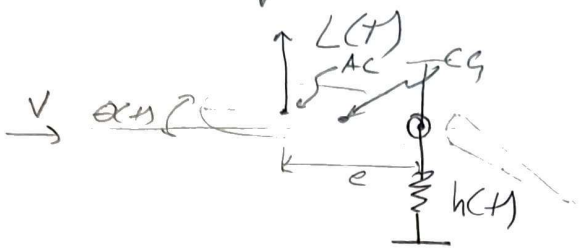
Problem No. 1




$\omega_h = 50$ rad/s
$\omega_\theta = 100$ rad/s
$\omega_\beta = 300$ rad/s
$a = -0.4$
$c = 0.6$
$b = 1$ m
$\bar{x}_\theta = 0.2$
$\bar{x}_\beta = 0.0125$
$r_\theta^2 = 0.25$
$r_\beta^2 = 0.00625$
$\mu = 40$
Sea Level

a) Find the flutter and divergence speed
Show graphs V-w and V-g

So my code derives everything then plots it.
I will show some of the steps here.

We are given F, M, and $M_\beta$. We can write them as so:

$$\begin{bmatrix} F/b \\ M/b^2 \\ M_\beta/b^2 \end{bmatrix} \xrightarrow{e^{i\omega t}} \begin{bmatrix} \bar{F}/b \\ \bar{M}/b^2 \\ \bar{M}_\beta/b^2 \end{bmatrix} e^{i\omega t}$$

We know $z = h + \theta(x-ab) + \beta(x-bc)$
which $T = \frac{1}{2}\int \dot{z}\,\rho\,dx$

Which after expanding, we get some terms

$\int \rho\,\dot{\theta}^2(x-ab)^2 dx \longrightarrow I_\theta \dot{\theta}^2 \longrightarrow m r_\theta^2\,\dot{\theta}^2$

$\int \rho\,\dot{\beta}^2(x-bc)^2 dx \longrightarrow I_\beta \dot{\beta}^2 \longrightarrow m r_\beta^2 \dot{\beta}^2$

$\int \rho\,\dot{h}\dot{\theta}(x-ab)\,dx \longrightarrow m x_\theta \dot{h}\dot{\theta}$

$\int \rho\,\dot{h}\dot{\beta}(x-bc)\,dx \longrightarrow m x_\beta \dot{h}\dot{\beta}$ ← Book error

$\int \rho\,\dot{\theta}\dot{\beta}(x-ab)(x-bc)\,dx \longrightarrow m r_\beta^2 + m x_\beta b(c-a)$

$\omega_h^2 = \frac{K_h}{m}$ $\quad \omega_\theta^2 = \frac{K_\theta}{mb^2} = \frac{K_\theta}{m r_\theta^2}\left(\frac{r_\theta^2}{b^2}\right)$ $\quad \omega_\beta^2 = \frac{K_\beta}{mb^2} = \frac{K_\beta}{m r_\beta^2}\left(\frac{r_\beta^2}{b^2}\right)$ } Used after dividing over m (next page)

We can define

$$U = \frac{1}{2}\left(K_h h^2 + K_\theta \theta^2 + K_\beta \beta^2\right)$$

Then we have Lagrange's Eq:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}}\right) - \frac{\partial T}{\partial q} + \frac{\partial U}{\partial q} = Q$$

My code does these calculations... so I won't do it all by hand...

So we eventually end up with an equation that sort of looks like (after dindy over m) we get something that looks like this:

$$[M]\{\ddot{x}_s\} + [K]\{x_s\} = \begin{bmatrix} F/b \\ M/b^2 \\ M_\beta/b^2 \end{bmatrix}$$

Probably where I begin to differ from the book...

So the book broke up everything since it easier to hardcode... I wanted Matlab to do it for me so I didn't.

So I plugged in the right side of the equation and let matlab create the matricies for me.

I ended up with 3 different matricies

$$[M_{nc}]\{\ddot{x}_s\} + [C_{nc}]\{\dot{x}_s\} + [K_{nc}]\{\ }\}$$

So the part that got sort of tricky was that I was missing terms outside of the matrieres like the book.

I would simply divide them out and keep track of their existence.

The book also forgets divide out $\pi$ from the aero matricies.

So we have something like this:

$$[M]\{\ddot{x}_s\} + [K]\{x_s\} = \frac{\pi q}{m}\left(\frac{2b^2}{V^2}[M_{nc}]\{\ddot{x}_s\} + \frac{2b}{V}[C_{nc}]\{\dot{x}\} + 2[K_{nc}]\{x_s\}\right)$$

If we make $(-i)^2 \Rightarrow -1$ $\quad x_s = x_{s_0} e^{i\omega t}$ (purely harmonic)

$$-\omega^2[M]\{x_s\} + [K]\{x_s\} = \frac{\pi q}{m}\left(-\frac{2b^2}{V^2}\omega^2[M_{nc}]\{x_s\} + \frac{2b}{V} i\omega[C_{nc}]\{x_s\} + 2[K_{nc}]\{x_s\}\right)$$

We can divide over the $\omega^2$, and add artificial structural damping to $[K]$ $(1+ig)$

$$\underbrace{\frac{(1+ig)}{\omega^2}}_{\lambda}[K]\{x_s\} = [M]\{x_s\} + \frac{\pi q}{m}\left(-\frac{2b^2}{V^2}[M_{nc}]\{x_s\} + \frac{2b}{V}\frac{i}{\omega}[C_{nc}]\{x_s\} + \frac{2}{\omega^2}[K_{nc}]\{x_s\}\right)$$

($\lambda$ eigenvalue)

Reduced frequency $\Rightarrow K = \frac{\omega b}{V}$

Quickly define (important for graphing later)

$$\frac{1}{\lambda_{Re(i)}} = \omega_i^2 \qquad g_i = \frac{\lambda_{Im(i)}}{\lambda_{Re(i)}}$$

$$q \rightarrow \tfrac{1}{2}\rho V^2 \qquad \mu = \frac{m}{\pi\rho b^2} \rightarrow \frac{1}{\mu} = \frac{\pi\rho b^2}{m}$$

$$\pi\left(\tfrac{1}{2}\rho V^2\right)\left(\frac{2b^2}{V^2}\right) \rightarrow \frac{1}{\mu}$$

$$\frac{b}{b}\pi\left(\tfrac{1}{2}\rho V^2\right)\left(\frac{2b}{V}\left(\frac{i}{\omega}\right)\right) \rightarrow \frac{1}{K} i \frac{1}{\mu}$$

$$\frac{b^2}{b^2}\pi\left(\tfrac{1}{2}\rho V^2\right)\left(\frac{2}{\omega^2}\right) \rightarrow \frac{1}{K^2}\frac{1}{\mu}$$

Rewrite w/ new variables

$$\underbrace{\frac{(1+ig)}{\omega^2}}_{\Omega}[\bar{K}]\{X_s\} = [\bar{M}]\{X_s\} + \frac{1}{\mu}\left(-[M_{nc}]\{X_s\} + \frac{i}{k}[C_{nc}]\{X_s\} + \frac{1}{k^2}[K_{nc}]\{X_s\}\right)$$

Then multiply both sides by $[\bar{K}]^{-1}$

Solve for eigenvalues of that matrix.

$$\sqrt{\frac{1}{\Omega_{Re(i)}}} = \omega_i \quad \text{and} \quad g_i = \frac{\Omega_{Im(i)}}{\Omega_{Re(i)}}$$

The rest is all Matlab from here.

Graphs attached in code (Publish)

$V_F = 301.5$ m/s ← where g crosse 0

$V_D = 635$ m/s ← where ω crosses 0

(I increased the number of points & lower step size)

b) Place a weight 5% chord position

Want to increase flutter speed by 10%, 15%, and 20%

With $V_F = 301.5$ m/s, we want:

10% → $V_F = 331.65$ m/s
15% → $V_F = 346.73$ m/s
20% → $V_F = 361.8$ m/s

We have the follow initial parameters
(will note that hw diagram is wrong with ab def)

- $\omega_h = 50$ rad/s
- $\omega_\theta = 100$ rad/s
- $\omega_\beta = 300$ rad/s
- $a = -0.4$
- $c = 0.6$
- $b = 1$ m
- $\frac{}{x_\theta} = 0.2$
- $\frac{}{x_\beta} = 0.0125$
- $\frac{}{r_\theta^2} = 0.25$
- $\frac{}{r_\beta^2} = 0.00625$
- $\mu = 40$
- $\rho = 1.225$ Kg/m$^3$ (Sea level)

First, we need to figure out what the initial mass is

$$\mu = \frac{m}{\pi \rho b^2} \longrightarrow M_0 = \mu \pi \rho b^2$$

The new mass will act shift the c.g location. We need to solve for said location. Pg. 372 is a good reference.

$$X_{cg} = \frac{m_0 X_1 + X_2 \Delta m}{m_0 + \Delta m}$$

$\Delta m$ is the variable that I will change later once this is all derived.

We need to solve for the original c.g point. We have $a$ and $\bar{X}_\theta$

$$a = \underbrace{\bar{X}_{1/2}}_{b/b} - \bar{X}_{sc} \qquad \bar{X}_\theta = \bar{X}_{cg} - \bar{X}_{sc}$$

$$\bar{X}_{sc} = \bar{X}_{1/2} - a \qquad b\bar{X}_{cg} = b\bar{X}_\theta + b\bar{X}_{sc}$$

$$X_{cg} = b\bar{X}_0 + b(\bar{X}_{1/2} - a)$$

$$X_1 = X_{cg} = b\bar{X}_0 + \underbrace{X_{1/2}}_{b} - ab$$

Solve for $X_{cg_{new}}$ now.

Need to solve for

$$\bar{X}_\theta = \frac{X_{cg} - X_{sc}}{b} \qquad \underbrace{\bar{X}_\beta = \frac{X_{cg} - X_H}{b}}_{\text{not dependent on new c.g.}}$$

$$a = \underbrace{\bar{X}_{1/2} - \bar{X}_{sc}}_{\text{these don't change either}}$$

$\bar{X}_\theta$ only variable that changes here

Solve for the new $\mu$

$$\mu = \frac{m_o + \Delta m}{\pi \rho b^2}$$

c and b don't change since characteristics of wing.

Need $r_\theta$, $r_\beta$, $\omega_h$, $\omega_\theta$, $\omega_\beta$

$m r_\theta^2 = I_\theta \quad m r_\beta^2 = I_\beta \quad \omega_h^2 = \frac{K_h}{m} \quad \omega_\theta^2 = \frac{K_\theta}{I_\theta} \quad \omega_\beta^2 = \frac{K_\beta}{I_\beta}$

We don't know $K_h$, $K_\theta$, $K_\beta$ (but they are const)

We can solve for $\omega_h$

$$(\omega_{h,0}^2)^2 = \left(\frac{K_h}{m_o}\right)^2 \rightarrow (\omega_{h,0})^2 (m_o) = K_h$$

$$\omega_h = \sqrt{\frac{K_h}{m_o + \Delta m}}$$

We need to solve for the inertias now ($I_\theta = I_o + m \bar{x}_\theta^2$)

$$\underbrace{m_o (b \bar{r}_{\theta,0})^2}_{\text{Known}} = \underbrace{I_{\theta,0}}_{\rightarrow} \qquad \underbrace{I_{\theta,0}}_{} - \underbrace{m_o (\bar{x}_{\theta,0} b)^2}_{\text{Known}} = I_o$$

We need this (constant)

$$I_{\theta_{new}} = I_o + (m_o + \Delta m)(\bar{x}_\theta b)^2$$

We can solve for $\bar{r}_\theta$ now

$$\bar{r}_\theta = \sqrt{\frac{I_{\theta new}}{(m_o + \Delta m) b^2}}$$

$I_\beta$ is not effected by the change in wing c.g. - only aileron c.g. $\therefore I_\beta$, $r_\beta$, and $\omega_\beta$ remain constant

Need $\omega_\theta$ now

$$\omega_\theta = \sqrt{\frac{K_\theta}{I_{\theta new}}}$$

Things I had to be careful about when adapting the code...

→ For some reason the problem assumes the wing and aileron have the same mass... this is no longer true.

Define $M_w = m_o + \Delta m$ and
$m_a = m_o$

→ The variables outside of the aero matrix will not change. Really only effects the M and K matrix

→ $\frac{K_e}{m_o + \Delta m} \Rightarrow \frac{K_e}{m_o(1 + \Delta m/m_o)}$

(nothing changed here)

→ Others will have a ratio of $\frac{M_a}{M_s}$ in it

Which is fine since we know those values.

| | $V_F$ | | |
|---|---|---|---|
| 10% → | 331.65 m/s | ⇒ ~5.25 Kg | } $\Delta m$ values |
| 15% → | 346.73 m/s | ⇒ ~7.15 Kg | |
| 20% → | 361.80 m/s | ⇒ ~8.76 Kg | |

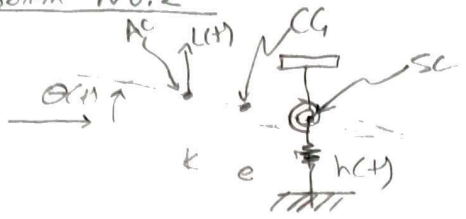
→ only graphed last one since graphs weren't asked for; that would have made publishing take much-much longer (it already takes a while)

→ I added a fun graph plotting the $V_F$ speeds and $\Delta m$ values. I wanted to see if any shape came of it. Was really surprised to see it had a very linear relationship.

→ It's hard to get super accurate to the 100th. So hence the ~ signs in front.

→ $V_F$ and $\Delta m$ can be approximated linearly... which is super useful for the design process.

Problem No. 2




$\omega_h = 10$ rad/s
$\omega_\theta = 25$ rad/s
$a = -0.2$ → Sea Level
$d = 0.4$
$b = 3m$
$\overline{x}_\theta = 0.1$
$r_\theta^2 = 0.5$
$C_{L_\alpha} = 2\pi$
$\mu = 20$

g is really with P-K graphs

a. Calc V-ω and V-g graphs. Flutter speed?

Do some ground work...

$$z(t) = h + x\theta \longrightarrow \dot{z} = \dot{h} + x\dot{\theta} \qquad T = \int \tfrac{1}{2}\rho \dot{z}\, dx$$

$$T = \underbrace{\tfrac{1}{2}\int \rho \dot{h}^2\, dx}_{\frac{1}{2}m\dot{h}^2} + \underbrace{\tfrac{1}{2}\int \rho x^2 \dot{\theta}^2\, dx}_{\frac{1}{2}I_\theta \dot{\theta}^2} + \underbrace{\int \rho \dot{h}\dot{\theta} x\, dx}_{mx_\theta \dot{\theta}\dot{h}}$$

We have the following equations for forces & moments:

$$P = \frac{\rho V^2}{2} b(2\pi)\frac{b^2\omega^2}{V^2}\left\{L_h\left(\frac{\overline{h}}{b}\right) + \left[L_\alpha - \left(\tfrac{1}{2}+a\right)L_h\right]\overline{\theta}\right\}e^{i\omega t}$$

$$\frac{\pi\rho b^3}{mb} = \frac{1}{\mu}$$

$$\frac{P}{m\cdot b} = \frac{\omega^2}{\mu}\left\{L_h\left(\frac{\overline{h}}{b}\right) + \left[L_\alpha - \left(\tfrac{1}{2}+a\right)L_h\right]\overline{\theta}\right\}e^{i\omega t}$$

$$\frac{m_{sc}}{mb^2} = \frac{mb}{mb}\pi\rho b^4\omega^2\left[\left(M_h - \left(\tfrac{1}{2}+a\right)L_h\right)\frac{\overline{h}}{b} + \left(M_\alpha - \left(\tfrac{1}{2}+a\right)(L_\alpha + M_h) + \left(\tfrac{1}{2}+a\right)^2 L_h\right)\overline{\theta}\right]e^{i\omega t}$$

$$M_h = \tfrac{1}{2} \quad \text{and} \quad M_\alpha = \tfrac{3}{8} - i\left(\tfrac{1}{k}\right)$$

$$\frac{m_{sc}}{mb^2} = \frac{\omega^2}{\mu}\left[\quad\right]$$

$$U = \frac{1}{2}\left(K_h h^2 + K_\theta \theta^2\right)$$

We have Lagrange's Eqs:

$$\frac{\partial}{\partial t}\left(\frac{\partial T}{\partial \dot{q}}\right) - \frac{\partial T}{\partial q} + \frac{\partial U}{\partial q} = Q$$

Yet again, the code will calculate the matrices.

First row divided by $b$
Second row divided by $b^2$

$$\omega_h^2 = \frac{K_h}{m} \qquad \omega_\theta^2 = \frac{K_\theta}{mb^2} = \frac{K_\theta}{mr_\theta^2}\left(\frac{r_\theta^2}{b^2}\right)$$

$$Q = \begin{bmatrix} P/mb \\ M_{sc}/mb^2 \end{bmatrix} = A$$

We have:

$$[M]\{\ddot{x}_s\} + [K]\{x_s\} = \frac{\omega^2}{\mu}[A]\{x_s\}$$

$$x_s = x_{s_i} e^{i\omega t}$$

$$-\omega^2[\bar{M}]\{x_s\} + [\bar{K}]\{x_s\} = \frac{\omega^2}{\mu}[\bar{A}]\{x_s\}$$

↑ add artificial damping $(1 + jg)$

Also now going to allow the motion to be $e^{pt} = e^{\sigma t} e^{i\omega t}$ so that $p = \sigma + j\omega$

Convert all real numbers...

$$[\bar{A}_{ij}] = \frac{\omega^2}{\mu}\,\text{Real}\left[\bar{A}_{ij}\right] + \frac{\omega^2}{\mu} j\,\text{Imag}\left[\bar{A}_{ij}\right]$$

Multiply · divide real elements by $q = \frac{1}{2}\rho V^2$

Multiply imaginary by $\frac{p}{j\omega}$ assuming ratio identical to 1.

$$\left(\frac{p}{j\omega} \approx 1\right) \rightarrow \text{near flutter}$$

1st $K = \frac{\omega b}{V} \rightarrow \omega = \frac{KV}{b}$ * Didn't need to mult & divide by $\frac{1}{2}$

$\frac{\omega^2}{\mu} \rightarrow \left(\frac{KV}{b}\right)^2 \rightarrow \left(\frac{V}{b}\right)^2\left(\frac{K^2}{\mu}\right)$ (first)

2nd $\frac{1}{2}\frac{\omega^2}{\mu}\left(\frac{\rho}{\frac{1}{2}\rho}\right) \rightarrow \frac{1}{2}\left(\frac{V^2}{b^2}\right)\left(\frac{K^2}{\mu}\right)\left(\frac{\rho}{\frac{1}{2}\rho}\right)\left(\frac{\omega b}{V}\right) \rightarrow \left(\frac{V}{b}\right)\left(\frac{K}{\mu}\right)(\rho)$ ($\omega \rightarrow K$)

$[\bar{A}] = [A_{ij}] + p[B_{ij}]$

$A_{ij} = \left(\frac{V}{b}\right)^2\left(\frac{K^2}{\mu}\right)\mathrm{Re}([\bar{A}])$

$B_{ij} = \left(\frac{V}{b}\right)\left(\frac{K}{\mu}\right)\mathrm{Im}([\bar{A}])$

$e^{pt} = e^{\sigma t}e^{i\omega t}$ → redo the eqs from before (instead of $\omega^2$)

$p^2[M]\{x_s\} - p[B_{ij}]\{x_s\} + \underbrace{([K] - [A_{ij}])}_{[K_{ij}]}\{x_s\} = 0$

$z = \begin{Bmatrix} x \\ v \end{Bmatrix} \quad x = \begin{Bmatrix} h/b \\ \theta \end{Bmatrix} \quad V = \{\dot{x}\}$

$[M]\{\dot{v}\} - [B_{ij}]\{v\} + [K_{ij}]\{x_s\} = 0$

$\{\dot{v}\} = -[M]^{-1}[K_{ij}]\{x\} + [M]^{-1}[B_{ij}]\{v\}$

$\{\dot{v}\} = \left[-[M]^{-1}[K_{ij}] \quad [M]^{-1}[B_{ij}]\right]\begin{Bmatrix} x \\ v \end{Bmatrix}$

$Q = \underbrace{\begin{bmatrix} 0 & I \\ -M^{-1}K_{ij} & M^{-1}B_{ij} \end{bmatrix}}_{\text{Plant matrix}}$ inputs of V and K

$P[Q]\{z\} = \{z\}$

Then we proceed similar to problem #1

$\boxed{V_F = 207.5 \text{ m/s}}$

b) See Matlab for graphs.

The V-g $V_f$ = 207.1 m/s

The quasi-steady $V_f$ = 197.5 m/s

The quasi-steady method will produce a lower estimate of flutter compared to the unsteady dynamic methods. It was also different by ~10 m/s... which while a bit off, it is not a bad estimate. Especially if doing some rough calculations early on in the design process. The quasi-steady model requires less work.

The V-g method and p-k method are both unsteady dynamic methods. I was actually quite impressed how close these answers were to each-other. It makes sense though because the two methods were not that different from each-other. P-k added only really a small change where the motion was no longer restricted to being only harmonic motion. This probably make p-k more accurate since it is one less assumption being levied on it. I also like graphing p-k more because I can use lines instead of *'s.

# Hw #4 - Aeroelasticity - ME597/AAE556

Victoria Nagorski - 12/2/22

## Contents

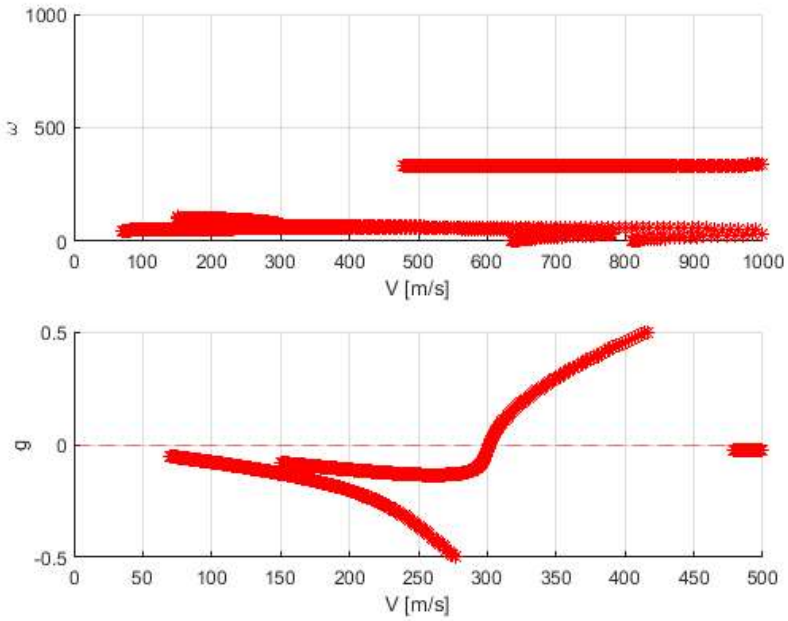

- Problem No. 1 Part a
- Problem No. 1 Part b
- Problem No. 2


## Problem No. 1 Part a

```
clear; clc; close all;
syms h(t) theta(t) beta(t) t x_theta x_beta rho m x y I_theta I_beta c a...
    T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15 T16 T17 T18 T19...
    C(k) b V Kh Kt Kb r_theta r_beta omega_h omega_t omega_b mu

variables = [omega_h omega_t  omega_b  a    c   b x_theta x_beta   r_theta   r_beta mu];
values =       [50    100      300   -0.4  0.6  1  0.2    0.0125  sqrt(0.25) sqrt(0.00625) 40];

B = 1;

% Define z
z = h + theta*x_theta + beta*x_beta;

% Give force and moment definitions
Q = V*theta + diff(h,t) + diff(theta,t)*b*((1/2)-a)+...
    (V/pi)*T10*beta + (b/(2*pi))*T11*diff(beta,t);
F = -pi*rho*b^2 *(diff(h,t,2)+V*diff(theta,t)-b*a*diff(theta,t,2)-...
    (V/pi)*T4*diff(beta,t)-(b/pi)*T1*diff(beta,t,2)) - 2*pi*rho*V*b*Q*C(k);
M = pi*rho*b^2*(b*a*diff(h,t,2)-V*b*((1/2)-a)*diff(theta,t)-...
    b^2*((1/8)+a^2)*diff(theta,t,2)-(V^2/pi)*(T4+T10)*beta+...
    ((V*b)/pi)*(-T1+T8+(c-a)*T4-(1/2)*T11)*diff(beta,t)+...
    b^2/pi*(T7+(c-a)*T1)*diff(beta,t,2)) + 2*pi*rho*V*b^2*(a+(1/2))*Q*C(k);
M_beta = pi*rho*b^2*((b/pi)*T1*diff(h,t,2)+((V*b)/pi)*(2*T9 +T1-...
    (a-(1/2))*T4)*diff(theta,t)-((2*b^2)/pi)*T13*diff(theta,t,2)-...
    (V/pi)^2*(T5-T4*T10)*beta+ ((V*b)/(2*pi*pi))*T4*T11*diff(beta,t)+...
    (b/pi)^2*T3*diff(beta,t,2)) - rho*V*b^2*T12*Q*C(k);

AeroTemp = expand([F/(m*b) M/(m*b^2) M_beta/(m*b^2)]);

% Solve for T (Kinetic Energy)
z_dot = diff(z,t);
temp = (1/2) * rho * int(int(expand(z_dot^2),x),y);
temp2 = expand(subs(expand(temp),rho*x*y,m));
temp3 = expand(subs(temp2,m*x_theta^2,m*r_theta^2));
T = subs(temp3,x_beta^2*m,m*r_beta^2);

% Solve for U (Potential Energy)
U = (1/2)*(Kh*h^2 + Kt*theta^2 + Kb*beta^2);

% Solve Lagranges Equations
mode = [h theta beta];
mode = mode(t);
eq = sym(zeros(length(mode),1));
for i = 1:length(mode)
    M = mode(i);
    dot = diff(M,t);
    eq(i,1) = simplify(diff(diff(T,dot),t) - diff(T,M) + diff(U,M));
end

damping = false;

% Create the matrices
sys.M = jacobian(eq,diff(mode,t,2)); % Mass
sys.K = jacobian(eq,mode); %
if damping == false
    sys.C = zeros(length(mode),length(mode));
else
    sys.C = jacobian(eq,diff(mode,t));
end

% Plug in values for M and K (constants)
```

```
M = subs(sys.M,x_beta*x_theta,(r_beta^2 + x_beta*b*(c-a)));
M = double(subs(M/m,variables,values));
K = subs(sys.K,[Kh Kt Kb],[omega_h^2 omega_t^2*r_theta^2 omega_b^2*r_beta^2]);
K = double(subs(K,variables,values));

% Create the Aero matrices
Mu = m/(pi*rho*b^2);
Kk = b/V;
TempM = simplify(Mu*jacobian(AeroTemp,diff(mode,t,2))*diag([b 1 1]));
TempC = simplify(Kk*Mu*jacobian(AeroTemp,diff(mode,t))*diag([b 1 1]));
TempK = simplify(Kk^2*Mu*jacobian(AeroTemp,mode)*diag([b 1 1]));

% Definitions for the T values
c = 0.6;
a = -0.4;
t1 = -((2+c^2)/3)*sqrt(1-c^2)+c*acos(c);
t3 = -((1-c^2)/8)*(5*c^2+4)+(1/4)*c*(7+2*c^2)*sqrt(1-c^2)*acos(c)-((1/8)+c^2)*acos(c)^2;
t4 = c*sqrt(1-c^2)-acos(c);
t5 = -(1-c^2)-acos(c)^2+2*c*sqrt(1-c^2)*acos(c);
t7 = c*((7+2*c^2)/8)*sqrt(1-c^2)-((1/8)+c^2)*acos(c);
t8 = -(1/3)*(1+2*c^2)*sqrt(1-c^2)+c*acos(c);
t9 = (1/2)*((sqrt(1-c^2)*(1-c^2))/3 + a*t4);
t10 = sqrt(1-c^2) + acos(c);
t11 = (2-c)*sqrt(1-c^2)+(1-2*c)*acos(c);
t12 = (2+c)*sqrt(1-c^2)-(1+2*c)*acos(c);
t13 = -(1/2)*(t7+(c-a)*t1);
t15 = t4+t10;
t16 = t1-t8-(c-a)*t4+(1/2)*t11;
t17 = -2*t9-t1+(a-(1/2))*t4;
t18 = t5-t4*t10;
t19 = -(1/2)*t4*t11;
TNums = [T1 T3 T4 T5 T7 T8 T9 T10 T11 T12 T13 T15 T16 T17 T18 T19];
tNums = [t1 t3 t4 t5 t7 t8 t9 t10 t11 t12 t13 t15 t16 t17 t18 t19];

% Plug in the T values
TTempM = simplify(subs(TempM,TNums,tNums));
TTempC = simplify(subs(TempC,TNums,tNums));
TTempK = simplify(subs(TempK,TNums,tNums));

% Condense to one matrix before looping
AeroT = (-TTempM + (j/k)*TTempC + (1/(k^2))*TTempK)/mu;

% Begin Iteration
num = 700;
rst1 = zeros(3,num);
rst2 = zeros(3,num);
vel = zeros(3,num);
for int = num:-1:1
    rk = int*0.001;      % Iterate on the k value

    % Plug in values for aero
    if rk == 0
        CC = 1;
    else
        CC = besselk(1,j*rk)/(besselk(0,j*rk)+besselk(1,j*rk));
    end
    z = double(subs(AeroT,[variables k C(k)],[values rk CC]));
    f = real(z);
    g = imag(z);
    Aero = f + j*g;

    % Solve eigenvalue problem
    eigen = eig(inv(K) * (M + Aero));
    Reig = abs(real(eigen));
    Ieig = imag(eigen);
    rst1(:,int) = sqrt(1./Reig);
    rst2(:,int) = Ieig./Reig;
    vel(:,int) = sqrt(1./Reig)*B/rk;
end

% Graph results
figure
subplot(2,1,1)
hold on
plot(vel,rst1,'*r')
grid on
xlabel('V [m/s]')
```

```
ylabel('\omega')
axis([0 1000 0 1000])
hold off
subplot(2,1,2)
hold on
plot(vel,rst2,'*r')
grid on
xlabel('V [m/s]')
ylabel('g')             % Eq 4.8.13b in textbook
axis([0 500 -0.5 0.5])
line([0 500],[0 0],'Color','red','LineStyle','--')
```



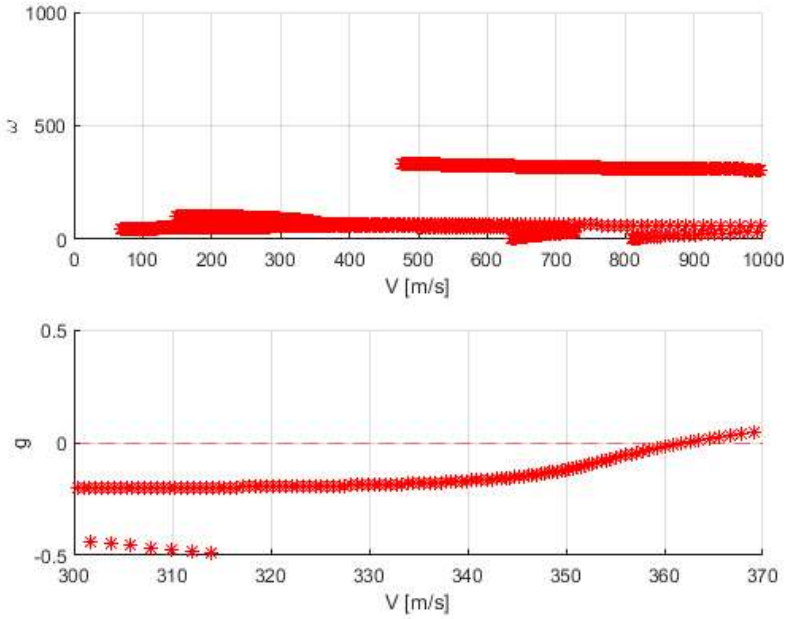

### Problem No. 1 Part b

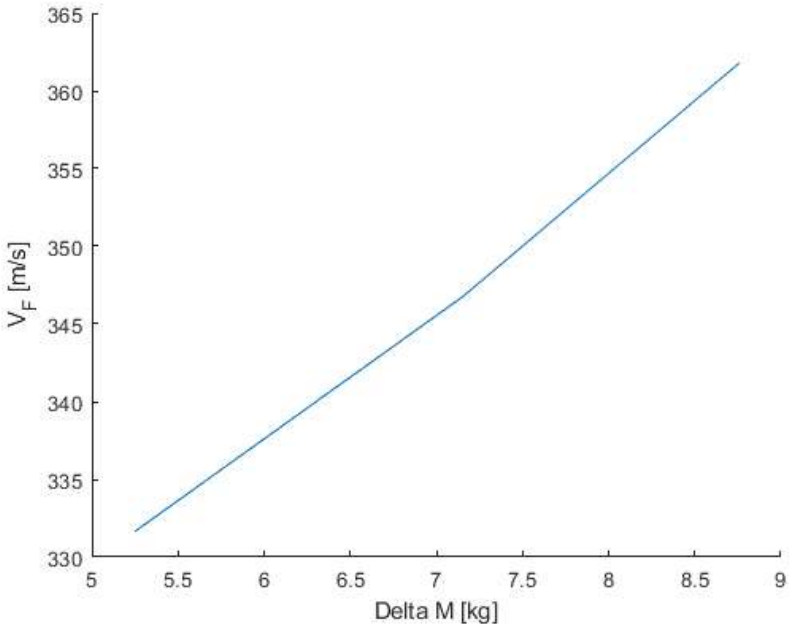
```
clear; clc; close all;

% ======= Calc New Values ========
Vf = 301.5;
Vf10 = Vf*1.10;
Vf15 = Vf*1.15;
Vf20 = Vf*1.20;
delta_M = 8.75; % kg

omega_H = 50;
omega_T = 100;
omega_B = 300;
A = -0.4;
C= 0.6;
B = 1;
x_thetaB = 0.2;
x_betaB = 0.0125;
r_thetaB = sqrt(0.25);
r_betaB = sqrt(0.00625);
MU = 40;
RHO = 1.225;    % Density at Sea-Level [Kg/m3]

% Solve for the new c.g. w/point mass
m0 = MU*pi*RHO*B^2; % Solve for the initial mass
x_12 = B;       % Halfway mark
x_sc = x_12 - A*B;
x_cg0 = x_thetaB*B - A*B + x_12;
x2= 0.05*2*B;   % 5% of the cord (2*b)
x_cg_new = (m0*x_cg0 + x2*delta_M)/(m0 + delta_M);

% Calc the new x_theta bar
x_thetaB_new = (x_cg_new - x_sc)/B;

% Calc the new MU
```

```
MU_new = (m0 + delta_M)/(pi*RHO*B^2);

% Calc inertias and new r_theta bar
I_theta0 = m0*(B*r_thetaB)^2;
I0 = I_theta0 - m0*(x_thetaB*B)^2;
I_theta_new = I0 + (m0 + delta_M)*B^2*x_thetaB_new^2;
r_thetaB_new = sqrt(I_theta_new/(B^2*(m0 + delta_M)));

% Calc Kh and Ktheta (these do not change)
Kh = omega_H^2*m0;
Ktheta = omega_T^2*I_theta0;

% Calculate the new omegas
omega_H_new = sqrt(Kh/(m0 + delta_M));
omega_T_new = sqrt(Ktheta/I_theta_new);

% ========== Start V-g code Prep ==============

syms h(t) theta(t) beta(t) t x_theta x_beta rho m x y I_theta I_beta c a...
    T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15 T16 T17 T18 T19...
    C(k) b V Kh Kt Kb r_theta r_beta omega_h omega_t omega_b mu Mw ma

variables = [omega_h omega_t  omega_b  a    c   b x_theta x_beta   r_theta   r_beta mu Mw ma];
values = [omega_H_new  omega_T_new  300 -0.4  0.6  1  x_thetaB_new    0.0125  r_thetaB_new sqrt(0.00625) MU_new (m0+delta_M) m0];

% Define z
z = h + theta*x_theta + beta*x_beta;

% Give force and moment definitions
Q = V*theta + diff(h,t) + diff(theta,t)*b*((1/2)-a)+...
    (V/pi)*T10*beta + (b/(2*pi))*T11*diff(beta,t);
F = -pi*rho*b^2 *(diff(h,t,2)+V*diff(theta,t)-b*a*diff(theta,t,2)-...
    (V/pi)*T4*diff(beta,t)-(b/pi)*T1*diff(beta,t,2)) - 2*pi*rho*V*b*Q*C(k);
M = pi*rho*b^2*(b*a*diff(h,t,2)-V*b*((1/2)-a)*diff(theta,t)-...
    b^2*((1/8)+a^2)*diff(theta,t,2)-(V^2/pi)*(T4+T10)*beta+...
    ((V*b)/pi)*(-T1+T8+(c-a)*T4-(1/2)*T11)*diff(beta,t)+...
    b^2/pi*(T7+(c-a)*T1)*diff(beta,t,2)) + 2*pi*rho*V*b^2*(a+(1/2))*Q*C(k);
M_beta = pi*rho*b^2*((b/pi)*T1*diff(h,t,2)+((V*b)/pi)*(2*T9 +T1-...
    (a-(1/2))*T4)*diff(theta,t)-((2*b^2)/pi)*T13*diff(theta,t,2)-...
    (V/pi)^2*(T5-T4*T10)*beta+ ((V*b)/(2*pi*pi))*T4*T11*diff(beta,t)+...
    (b/pi)^2*T3*diff(beta,t,2)) - rho*V*b^2*T12*Q*C(k);

AeroTemp = expand([F/(Mw*b) M/(Mw*b^2) M_beta/(Mw*b^2)]);

% Solve for T (Kinetic Energy)
z_dot = diff(z,t);
temp = expand((1/2) * rho * z_dot^2);
temp2 = expand(subs(expand(temp),rho*x_theta^2,Mw*r_theta^2));
temp3 = expand(subs(temp2,rho*x_beta^2,ma*r_beta^2));
temp4 = expand(subs(temp3,rho*diff(h(t), t)^2,Mw*diff(h(t), t)^2));
temp5 = expand(subs(temp4,rho*x_beta*diff(beta(t), t)*diff(h(t), t),ma*x_beta*diff(beta(t), t)*diff(h(t), t)));
temp6 = expand(subs(temp5,rho*x_theta*diff(h(t), t)*diff(theta(t), t),Mw*x_theta*diff(h(t), t)*diff(theta(t),t)));
T = subs(temp6,rho*x_beta*x_theta*diff(beta(t), t)*diff(theta(t), t),(ma*r_beta^2 + ma*x_beta*(c-a))*diff(beta(t), t)*diff(theta(t), t));

% Solve for U (Potential Energy)
U = (1/2)*(Kh*h^2 + Kt*theta^2 + Kb*beta^2);

% Solve Lagranges Equations
mode = [h theta beta];
mode = mode(t);
eq = sym(zeros(length(mode),1));
for i = 1:length(mode)
    M = mode(i);
    dot = diff(M,t);
    eq(i,1) = simplify(diff(diff(T,dot),t) - diff(T,M) + diff(U,M));
end

damping = false;

% Create the matrices
sys.M = jacobian(eq,diff(mode,t,2)); % Mass
sys.K = jacobian(eq,mode); %
if damping == false
    sys.C = zeros(length(mode),length(mode));
else
    sys.C = jacobian(eq,diff(mode,t));
end
```

```
% Plug in values for M and K (constants)
% M = subs(sys.M,x_beta*x_theta,(r_beta^2 + x_beta*b*(c-a)));
M = double(subs(expand(sys.M/Mw),variables,values));
K = subs(sys.K,[Kh Kt Kb],[omega_h^2 omega_t^2*r_theta^2 (omega_b^2*r_beta^2)/(1 + delta_M/m0)]);
K = double(subs(K,variables,values));

% Create the Aero matrices
Mu = Mw/(pi*rho*b^2);
Kk = b/V;
TempM = simplify(Mu*jacobian(AeroTemp,diff(mode,t,2))*diag([b 1 1]));
TempC = simplify(Kk*Mu*jacobian(AeroTemp,diff(mode,t))*diag([b 1 1]));
TempK = simplify(Kk^2*Mu*jacobian(AeroTemp,mode)*diag([b 1 1]));

% Definitions for the T values
c = 0.6;
a = -0.4;
t1 = -((2+c^2)/3)*sqrt(1-c^2)+c*acos(c);
t3 = -((1-c^2)/8)*(5*c^2+4)+(1/4)*c*(7+2*c^2)*sqrt(1-c^2)*acos(c)-((1/8)+c^2)*acos(c)^2;
t4 = c*sqrt(1-c^2)-acos(c);
t5 = -(1-c^2)-acos(c)^2+2*c*sqrt(1-c^2)*acos(c);
t7 = c*((7+2*c^2)/8)*sqrt(1-c^2)-((1/8)+c^2)*acos(c);
t8 = -(1/3)*(1+2*c^2)*sqrt(1-c^2)+c*acos(c);
t9 = (1/2)*((sqrt(1-c^2)*(1-c^2))/3 + a*t4);
t10 = sqrt(1-c^2) + acos(c);
t11 = (2-c)*sqrt(1-c^2)+(1-2*c)*acos(c);
t12 = (2+c)*sqrt(1-c^2)-(1+2*c)*acos(c);
t13 = -(1/2)*(t7+(c-a)*t1);
t15 = t4+t10;
t16 = t1-t8-(c-a)*t4+(1/2)*t11;
t17 = -2*t9-t1+(a-(1/2))*t4;
t18 = t5-t4*t10;
t19 = -(1/2)*t4*t11;
TNums = [T1 T3 T4 T5 T7 T8 T9 T10 T11 T12 T13 T15 T16 T17 T18 T19];
tNums = [t1 t3 t4 t5 t7 t8 t9 t10 t11 t12 t13 t15 t16 t17 t18 t19];

% Plug in the T values
TTempM = simplify(subs(TempM,TNums,tNums));
TTempC = simplify(subs(TempC,TNums,tNums));
TTempK = simplify(subs(TempK,TNums,tNums));

% Condense to one matrix before looping
AeroT = (-TTempM + (j/k)*TTempC + (1/(k^2))*TTempK)/mu;

% Begin Iteration
num = 700;
rst1 = zeros(3,num);
rst2 = zeros(3,num);
vel = zeros(3,num);
for int = num:-1:1
    rk = int*0.001;      % Iterate on the k value

    % Plug in values for aero
    if rk == 0
        CC = 1;
    else
        CC = besselk(1,j*rk)/(besselk(0,j*rk)+besselk(1,j*rk));
    end
    z = double(subs(AeroT,[variables k C(k)],[values rk CC]));
    f = real(z);
    g = imag(z);
    Aero = f + j*g;

    % Solve eigenvalue problem
    eigen = eig(inv(K) * (M + Aero));
    Reig = abs(real(eigen));
    Ieig = imag(eigen);
    rst1(:,int) = sqrt(1./Reig);
    rst2(:,int) = Ieig./Reig;
    vel(:,int) = sqrt(1./Reig)*B/rk;
end

% Graph results
figure
subplot(2,1,1)
hold on
plot(vel,rst1,'*r')
```

```
grid on
xlabel('V [m/s]')
ylabel('\omega')
axis([0 1000 0 1000])
hold off
subplot(2,1,2)
hold on
plot(vel,rst2,'*r')
grid on
xlabel('V [m/s]')
ylabel('g')              % Eq 4.8.13b in textbook
axis([300 370 -0.5 0.5])
line([0 500],[0 0],'Color','red','LineStyle','--')

figure
hold on
plot([5.25 7.15 8.76],[331.65 346.73 361.80])
xlabel('Delta M [kg]')
ylabel('V_F [m/s]')
hold off
```



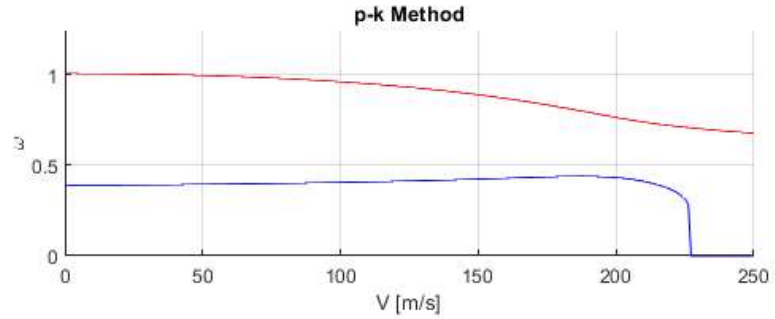
## Problem No. 2

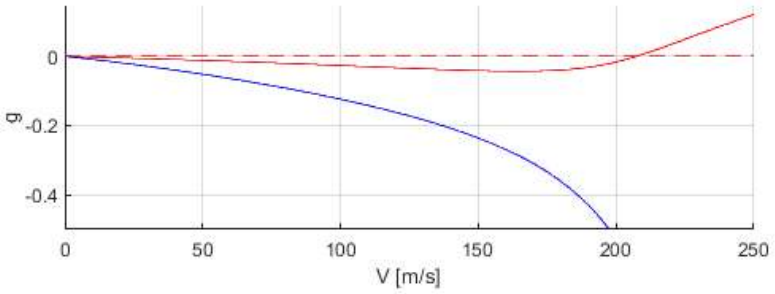
```
clear;clc;close all;

% ================== Set-up the Problem ======================
syms h(t) theta(t) beta(t) t x_theta x_beta rho m x y I_theta I_beta c a...
    T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15 T16 T17 T18 T19...
    C(k) b V Kh Kt Kb r_theta r_beta omega k omega_h omega_t mu Lh La ...
    Ma d_bar Cla

% Define z
z = h + theta*x;

% Solve for T (Kinetic Energy)
z_dot = diff(z,t);
temp = expand((1/2) * rho * z_dot^2);
temp2 = expand(subs(temp,rho*x^2,m*r_theta^2));
temp3 = expand(subs(temp2,rho*x,m*x_theta));
T = expand(subs(temp3,rho,m));

% Define U
U = (1/2)*Kh*h^2 + (1/2)*Kt*theta^2;

% Define the modes (h is really h/b)
mode = [h theta];

% Solve Lagrange's Equation
mode = mode(t);
eq = sym(zeros(length(mode),1));
for i = 1:length(mode)
    M = mode(i);
    dot = diff(M,t);
    eq(i,1) = simplify(diff(diff(T,dot),t) - diff(T,M) + diff(U,M));
end

% Change out defined terms in equations
eq2 = expand(subs(expand(eq/m),Kh/m,omega_h^2));
eq3 = expand(subs(eq2,Kt/m,omega_t^2*r_theta^2));

damping = false;

% Create the matrices
sys.M = jacobian(eq3,diff(mode,t,2)); % Mass
sys.K = jacobian(eq3,mode); %
if damping == false
    sys.C = zeros(length(mode),length(mode));
else
    sys.C = jacobian(eq3,diff(mode,t));
end

% Solve for the aero matrix (h is h/b and theta)
Mh = 1/2;
P_mb = (1/mu)* (Lh*h + (La - (1/2 + a)*Lh)*theta);
M_mbb = (1/mu)* ((Mh-(1/2 + a)*Lh)*h + (Ma -(1/2 + a)*(La+Mh)+(1/2+a)^2*Lh)*theta);
aero_eq = [P_mb;M_mbb];
sys.Aero = jacobian(aero_eq,mode);

% ====================== The p-k Code ========================
% Parameters
variables = [omega_h omega_t  a   d_bar b x_theta r_theta Cla mu];
values =       [10      25  -0.2   0.4  3  0.1  sqrt(0.5) 2*pi 20];
MU = 20;
B = 3;
OMEGAT = 25;

% Plug Into Mass and K matrices (Constant)
M = double(subs(sys.M,variables,values));
KK = double(subs(sys.K,variables,values));

% Replace Lh, La, and Ma in Aero pg. 415 and 417
lh = 1 - j*2*C(k)/k;
la = (1/2) - j*(1+2*C(k))/k - (2*C(k)/k^2);
ma = (3/8)- j/k;
AeroT = subs(mu*sys.Aero,[Lh La Ma],[lh la ma]);

% Begin Iteration Code
```

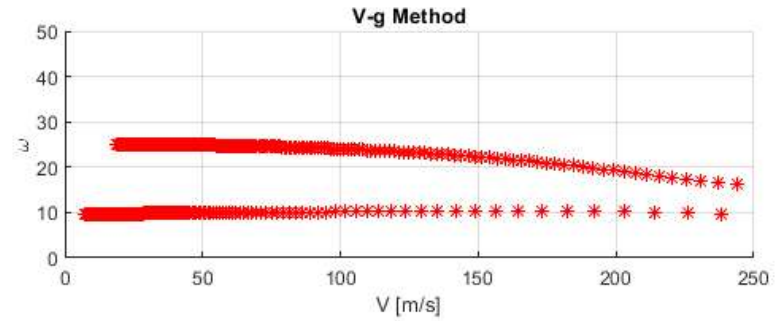
```
Nv = 1000;
Vel = linspace(0.001,1000,Nv);
sigma1 = zeros(1,Nv);
w1 = zeros(1,Nv);
sigma2 = zeros(1,Nv);
w2 = zeros(1,Nv);
for iter = 1:Nv
    V = Vel(iter);
    K = 0.5;            % Initial guess for k
    e = 1;              % Initialize error for k
    while e > 0.0001
        Ck = besselh(1,2,K)/(besselh(1,2,K)+1i*besselh(0,2,K));

        % Plug in values to Aero matrix
        Aero = double(subs(AeroT,[C(k) mu a k],[Ck 20 -0.2 K]));
        % Break Aero into Aij and Bij matricies
        Aij = V^2*K^2/(MU*B^2) * real(Aero);
        Bij = V/B *(K/MU)*imag(Aero);
        % Combine Matricies
        Kij_bar =  KK- Aij;

        % Make the Plant
        A = [zeros(length(M)) eye(length(M));
             -inv(M)*Kij_bar   inv(M)*Bij];

        % Solve eigenvalues
        p = eig(A);

        [wu index] = max(abs(imag(p)));
        if wu == 0
            e = 1e-6;
        else
            knew = wu*B/V;
            e = abs(knew-K);
            K = knew;
        end
    end
    sigma1(:,iter) = real(p(index))/imag(p(index));
    w1(:,iter) = imag(p(index));


    e = 1;              % Initialize error for force
    K = 0.5;            % Initilaize k
    while e > 0.0001
        Ck = besselh(1,2,K)/(besselh(1,2,K)+1i*besselh(0,2,K));

        % Plug in values to Aero matrix
        Aero = double(subs(AeroT,[C(k) mu a k],[Ck 20 -0.2 K]));
        % Break Aero into Aij and Bij matricies
        Aij = V^2*K^2/(MU*B^2) * real(Aero);
        Bij = V/B *(K/MU)*imag(Aero);
        % Combine Matricies
        Kij_bar = KK - Aij;

        % Make the Plant
        A = [zeros(length(M)) eye(length(M));
             -inv(M)*Kij_bar   inv(M)*Bij];

        % Solve eigenvalues
        p = eig(A);

        [wu1 index] = min(abs(imag(p)));
        if wu1 == 0
            e = 1e-6;
        else
            knew = wu1*B/V;
            e = abs(knew-K);
            K = knew;
        end
    end
    sigma2(:,iter) = real(p(index))/imag(p(index));
    w2(:,iter) = imag(p(index));
end

figure
subplot(2,1,1)
hold on
```

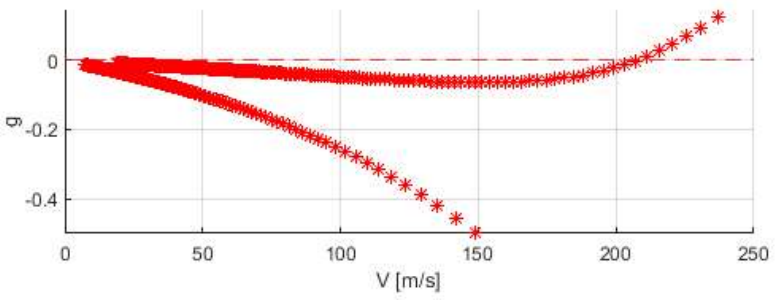
```
title('p-k Method')
plot(Vel,w1./OMEGAT,'r');
plot(Vel,w2./OMEGAT,'b')
grid on
xlabel('V [m/s]')
ylabel('\omega')
axis([0 250 0 1.25])
hold off
subplot(2,1,2)
hold on
plot(Vel,sigma1,'r');
plot(Vel,sigma2,'b')
grid on
xlabel('V [m/s]')
ylabel('g')             % Eq 4.8.13b in textbook
axis([0 250 -0.5 0.15])
line([0 250],[0 0],'Color','red','LineStyle','--')
hold off

% ====================== The v-g Code ==========================
% Plug in values for M and K (constants)
K = KK;

% Begin Iteration
num = 400;
rst1 = zeros(2,num);
rst2 = zeros(2,num);
vel = zeros(2,num);
for int = num:-1:1
    rk = int*0.01;      % Iterate on the k value

    % Plug in values for aero
    if rk == 0
        CC = 1;
    else
        CC = besselk(1,j*rk)/(besselk(0,j*rk)+besselk(1,j*rk));
    end
    z = double(subs(AeroT/MU,[variables k C(k)],[values rk CC]));
    f = real(z);
    g = imag(z);
    Aero = f + j*g;

    % Solve eigenvalue problem
    eigen = eig(inv(K) * (M + Aero));
    Reig = abs(real(eigen));
    Ieig = imag(eigen);
    rst1(:,int) = sqrt(1./Reig);
    rst2(:,int) = Ieig./Reig;
    vel(:,int) = sqrt(1./Reig)*B/rk;
end

% Graph results
figure
subplot(2,1,1)
hold on
title('V-g Method')
plot(vel,rst1,'*r')
grid on
xlabel('V [m/s]')
ylabel('\omega')
axis([0 250 0 50])
hold off
subplot(2,1,2)
hold on
plot(vel,rst2,'*r')
grid on
xlabel('V [m/s]')
ylabel('g')             % Eq 4.8.13b in textbook
axis([0 250 -0.5 0.15])
line([0 250],[0 0],'Color','red','LineStyle','--')

% ====================== Quasi-Steady ==========================
syms q V e
% Create the matrices
% e = 0.4;
rho = 1.225;    % Density at Sea-Level [Kg/m3]
LIFTq = q*Cla*b*(diff(h,t)/V + b/V*(1/2-a)*diff(theta,t) + theta);
```

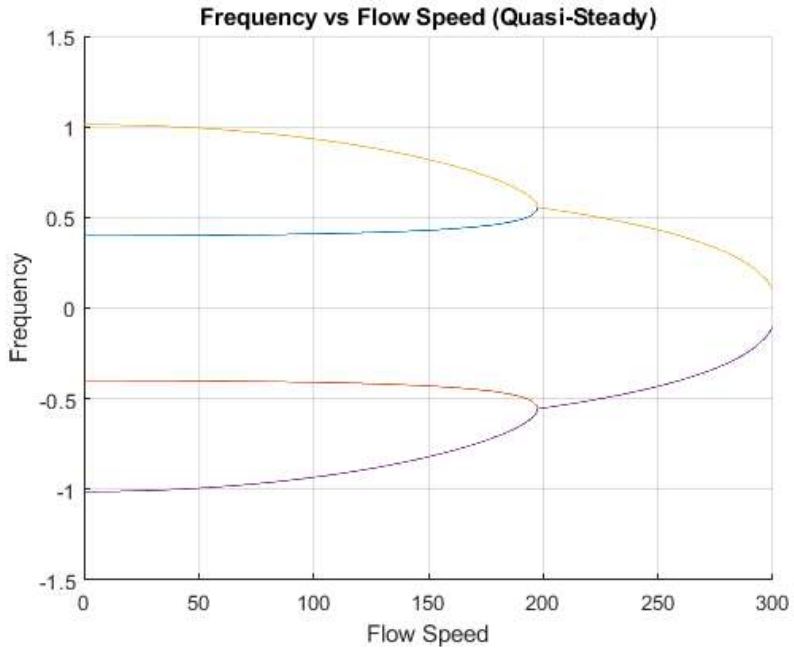
```
aero_q = expand([LIFTq;-e*LIFTq]);

K = sys.K/omega_t^2;

damping = false;    % Set C=0 so that motion is harmonic (pg 348)
AA = jacobian(aero_q,mode);
AA = AA*(Cla*V^2)/(Cla*q*b*b^2*omega_t^2*pi*mu);
if damping == false
    sys.C = zeros(length(mode),length(mode));
else
    sys.C = jacobian(aero_q,diff(mode,t));
end
A_quasi = [zeros(length(mode),length(mode))    eye(length(mode));
                -inv(sys.M)*(K+AA)         -inv(sys.M)*sys.C];

A_quasi = simplify(subs(A_quasi,q,(1/2)*rho*V^2));
A_quasi = subs(A_quasi,[variables e],[values 0.3]);


% Plug in velocities
Velocities = 0:0.1:300;
other.frequencies = zeros(length(mode)*2,length(Velocities));
other.damping = zeros(length(mode)*2,length(Velocities));
for i = 1:length(Velocities)
    v = Velocities(i);
    Temp = double(subs(A_quasi,V,v));
    p = eig(Temp);
    other.frequencies(:,i) = imag(p);
    other.damping(:,i) = real(p);
end

% Plot everything
figure
hold on
for i = 1:(length(mode)*2)
     plot(Velocities,other.frequencies(i,:))
end
title('Frequency vs Flow Speed (Quasi-Steady)')
xlabel('Flow Speed')
ylabel('Frequency')
grid('on')
hold off

figure
hold on
for i = 1:(length(mode)*2)
     plot(Velocities,other.damping(i,:))
end
title('Damping vs Flow Speed (Quasi-Steady)')
xlabel('Flow Speed')
ylabel('Damping')
grid('on')
hold off
```

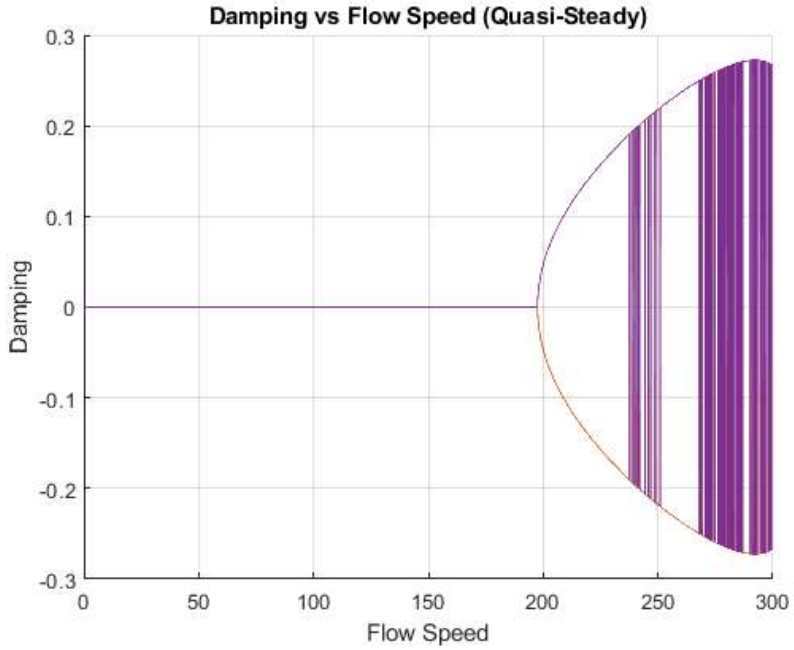

p-k Method
ω
V [m/s]


g
V [m/s]


V-g Method
ω
V [m/s]


g
V [m/s]




---

*Published with MATLAB® R2022a*