Problem No. 1

$$\rho \frac{\partial^2 w}{\partial t^2} + EI \frac{\partial^4 w}{\partial x^4} + N_x \frac{\partial^2 w}{\partial x^2} + \frac{2q}{\sqrt{M^2-1}} \frac{\partial w}{\partial x} = 0$$

Consider the transverse displacement synchronous solution as:

$$w(x,t) = \phi(x) e^{i\omega t}$$

ϕ is the spatial solution

a) Sub $w(x,t)$ into PDE

$$\frac{\partial^2 w}{\partial t^2} \rightarrow (i\omega)^2 \phi e^{i\omega t} \rightarrow (-1)\omega^2 \phi e^{i\omega t}$$

$$\frac{\partial^4 w}{\partial x^4} \rightarrow \ddddot{\phi} e^{i\omega t} \qquad \frac{\partial w}{\partial x} \rightarrow \dot{\phi} e^{i\omega t}$$

$$\frac{\partial^2 w}{\partial x^2} \rightarrow \ddot{\phi} e^{i\omega t}$$

Cancel out $e^{i\omega t}$ & put together

$$\boxed{-\omega^2 \rho \phi + EI \ddddot{\phi} + N_x \ddot{\phi} + \frac{2q}{\sqrt{M^2-1}} \dot{\phi} = 0}$$

b) $\phi(x) = \sum_{n=1}^{N} c_n \sin\left(\frac{\pi n x}{a}\right)$ to approximate...

$$\dot{\phi} = \sum_{n=1}^{N} c_n \cos\left(\frac{\pi n x}{a}\right)\left(\frac{\pi n}{a}\right)$$

$$\ddot{\phi} = -\sum_{n=1}^{N} c_n \sin\left(\frac{\pi n x}{a}\right)\left(\frac{\pi n}{a}\right)^2$$

$$\dddot{\phi} = -\sum_{n=1}^{N} c_n \cos\left(\frac{\pi n x}{a}\right)\left(\frac{\pi n}{a}\right)^3$$

$$\ddddot{\phi} = \sum_{n=1}^{N} c_n \sin\left(\frac{\pi n x}{a}\right)\left(\frac{\pi n}{a}\right)^4$$

Plug in:

$$\sum_{n=1}^{N} c_n \left(-\omega^2 \rho + EI\left(\frac{\pi n}{a}\right)^4 - N_x\left(\frac{\pi n}{a}\right)^2\right) \sin\left(\frac{\pi n x}{a}\right) + \sum_{n=1}^{N} c_n \left(\frac{\pi n}{a}\right)\left(\frac{2q}{\sqrt{M^2-1}}\right) \cos\left(\frac{\pi n x}{a}\right) = 0$$

Divide by EI and define the following terms:

$$\lambda = \frac{2q}{EI\sqrt{\pi^2 - 1}}$$

$$\omega_{0,1}^2 = \left(\frac{\pi}{a}\right)^4 - \frac{N_x}{EI}\left(\frac{\pi}{a}\right)^2$$

$$\omega_{0,2}^2 = \left(\frac{2\pi}{a}\right)^4 - \frac{N_x}{EI}\left(\frac{2\pi}{a}\right)^2$$

First eq multiply by $\sin\left(\frac{\pi x}{a}\right)$ and integrate:

$$\int_0^a C_1\left(-\frac{\omega^2\rho}{EI} + \omega_{0,1}^2\right)\sin^2\left(\frac{\pi x}{a}\right)dx + \int_0^a C_2\left(-\frac{\omega^2\rho}{EI} + \omega_{0,2}^2\right)\sin\left(\frac{\pi x}{a}\right)\sin\left(\frac{2\pi x}{a}\right)dx$$

$$+ \int_0^a C_1\left(\frac{\pi}{a}\right)\lambda \sin\left(\frac{\pi x}{a}\right)\cos\left(\frac{\pi x}{a}\right)dx$$

$$+ \int_0^a C_2\left(\frac{2\pi}{a}\right)\lambda \sin\left(\frac{\pi x}{a}\right)\cos\left(\frac{2\pi x}{a}\right)dx$$

Second eq multiply by $\sin\left(\frac{2\pi x}{a}\right)$ and integrate:

$$\int_0^a C_1\left(\frac{-\omega^2\rho}{EI} + \omega_{0,1}^2\right)\sin\left(\frac{2\pi x}{a}\right)\sin\left(\frac{\pi x}{a}\right)dx$$

$$+ \int_0^a C_2\left(\frac{-\omega^2\rho}{EI} + \omega_{0,2}^2\right)\sin^2\left(\frac{2\pi x}{a}\right)dx$$

$$+ \int_0^a C_1\left(\frac{\pi}{a}\right)\lambda \sin\left(\frac{2\pi x}{a}\right)\cos\left(\frac{\pi x}{a}\right)dx$$

$$+ \int_0^a C_2\left(\frac{2\pi}{a}\right)\lambda \sin\left(\frac{2\pi x}{a}\right)\cos\left(\frac{2\pi x}{a}\right)dx$$

I used a calculator to calculate the integrals... I even checked two different ones and they both said differently than the book. So I think the textbook is wrong here...

I attached Matlab's outputs for the integration of the sines and cosines.

After breaking up the equations, I get the following matrix:

$$\begin{bmatrix} \frac{a}{2}\left(\omega_{0,1}^2 - \frac{\omega^2\rho}{EI}\right) & -\frac{4\lambda}{3} \\ \frac{4\lambda}{3} & \frac{a}{2}\left(\omega_{0,2}^2 - \frac{\omega^2\rho}{EI}\right) \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solve for C_1 and C_2

So the first thing to realize is that this is an eigenvalue problem.

If we define $\left(\frac{a}{2}\right)\left(\frac{\omega^2\rho}{EI}\right) = \Omega$, the problem begins to look like:

$$\begin{bmatrix} \frac{a}{2}\omega_{0,1}^2 - \Omega & -\frac{4\lambda}{3} \\ \frac{4\lambda}{3} & \frac{a}{2}\omega_{0,2}^2 - \Omega \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$(A - I\Omega)\hat{x} = 0$; $\hat{x} = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$

$$A\hat{x} = \Omega\hat{x}$$

I used Matlab to solve this since I had a hard time tracking all the variables.

I will solve for Ω by hand since I will use that later... (Matlab harder to decipher)

$$\det(A - I\Omega) = 0$$

$$\left(\frac{a}{2}\omega_{0,1}^2 - \Omega\right)\left(\frac{a}{2}\omega_{0,2}^2 - \Omega\right) + \left(\frac{4\lambda}{3}\right)^2 = 0$$

$$\left(\frac{a}{2}\right)^2\omega_{0,1}^2\omega_{0,2}^2 - \left(\frac{a}{2}\right)\omega_{0,1}^2\Omega - \left(\frac{a}{2}\right)\omega_{0,2}^2\Omega + \Omega^2 + \left(\frac{4\lambda}{3}\right)^2 = 0$$

$$\Omega^2 - \left(\frac{a}{2}\right)\left(\omega_{0,1}^2 + \omega_{0,2}^2\right)\Omega + \left(\frac{a}{2}\right)^2\omega_{0,1}^2\omega_{0,2}^2 + \left(\frac{4\lambda}{3}\right)^2 = 0$$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\Omega_{1,2} = \frac{(a/2)(\omega_{o,1}^2 + \omega_{o,2}^2) \pm \sqrt{(a/2)^2(\omega_{o,1}^2 + \omega_{o,2}^2)^2 - 4(a/2)^2\omega_{o,1}^2\omega_{o,2}^2 - 4\left(\frac{4\lambda}{3}\right)^2}}{2}$$

$$\Omega_{1,2} = (a/4)(\omega_{o,1}^2 + \omega_{o,2}^2) \pm (a/4)\sqrt{(\omega_{o,1}^2 + \omega_{o,2}^2)^2 - 4\omega_{o,1}^2\omega_{o,2}^2 - 4\left(\frac{4\lambda}{3}\right)^2\left(\frac{2}{a}\right)^2}$$

We'll come back to this answer...

Need to solve for the eigenvector $\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \hat{x}$

Trying to minimize, so use the smaller eigenvalue (the one that subtracts, Ω_2)

Set $c_2 = 1$ ... solve for c_1 in $A\hat{x} = \Omega\hat{x}$

$$\begin{bmatrix} \frac{a}{2}\omega_{o,1}^2 & -\frac{4\lambda}{3} \\ \frac{4\lambda}{3} & \frac{a}{2}\omega_{o,2}^2 \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \Omega_2 c_1 \\ \Omega_2 c_2 \end{bmatrix}$$

Use the first equation to solve for c_1

$$\frac{a}{2}\omega_{o,1}^2 c_1 - \frac{4\lambda}{3} = \Omega_2 c_1$$

$$\left(\frac{a}{2}\omega_{o,1}^2 - \Omega_2\right)c_1 = \frac{4\lambda}{3}\frac{1}{\frac{a}{2}\omega_{o,1}^2 - \Omega_2}$$

$$\boxed{c_1 = \frac{4\lambda}{3}\left(\frac{1}{(a/2)\omega_{o,1}^2 - \Omega_2}\right)}$$

$$\boxed{c_2 = 1}$$

This was checked using Matlab. If variable "Bottom" equals 0, its because the 2nd equation is equal on both sides when using c_1 and c_2. I subtracted both sides to eachother - which would equal 0 if equal.

Put it all together:

$$C_1 = \frac{4\lambda}{3}\left(\frac{1}{(a/2)\omega_{011}^2 - \Omega_2}\right) \qquad \lambda = \frac{2q}{EI\sqrt{M^2-1}}$$

$$\omega_{011}^2 = \left(\frac{\pi}{a}\right)^4 - \frac{N_x}{EI}\left(\frac{\pi}{a}\right)^2$$

$$\Omega_2 = \frac{a}{4}(\omega_{011}^2 + \omega_{012}^2) - \frac{a}{4}\sqrt{(\omega_{011}^2 + \omega_{012}^2)^2 - 4\omega_{011}^2\omega_{012}^2 - 4\left(\frac{4\lambda}{3}\right)^2\left(\frac{2}{a}\right)^2}$$

The synchronous solution for eq(1) is

$$w(x,t) = \phi(x)e^{i\omega t}$$

ϕ approximated with $\phi(x) = \sum_{n=1}^{N} C_n \sin\left(\frac{\pi n x}{a}\right)$

$N = 2$

$$\phi(x) = C_1 \sin\left(\frac{\pi x}{a}\right) + \sin\left(\frac{2\pi x}{a}\right)$$

$$\boxed{w(x,t) = \left[C_1 \sin\left(\frac{\pi x}{a}\right) + \sin\left(\frac{2\pi x}{a}\right)\right]e^{i\omega t}}$$

c) Solve for V_F w/ two-mode expansion

Go back to eigenvalue equation...

Frequencies will merge when the terms under the radical go to 0.

$$\frac{(\omega_{0,1}^2 + \omega_{0,2}^2)^2}{4} - \frac{4\omega_{0,1}^2\omega_{0,2}^2}{4} - 4\left(\frac{4\lambda}{3}\right)^2\left(\frac{2}{a}\right)^2 = 0$$

$$\sqrt{\left(\frac{4\lambda}{3}\right)^2} = \sqrt{\left(\frac{a}{2}\right)^2 \frac{(\omega_{0,1}^2 + \omega_{0,2}^2)^2}{4}} - \omega_{0,1}^2\omega_{0,2}^2\left(\frac{a}{2}\right)^2$$

$$\lambda = \frac{3a}{8}\sqrt{\frac{(\omega_{0,1}^2 + \omega_{0,2}^2)^2}{4} - \omega_{0,1}^2\omega_{0,2}^2}$$

$$\lambda = \frac{2q}{EI\sqrt{M^2-1}} \leftarrow \frac{1}{2}\rho V^2$$

$$\frac{\rho V^2}{EI\sqrt{M^2-1}} = \frac{3a}{8}\sqrt{\frac{(\omega_{0,1}^2 + \omega_{0,2}^2)}{4} - \omega_{0,1}^2\omega_{0,2}^2}\left(\frac{EI\sqrt{M^2-1}}{\rho}\right)$$

$$\boxed{V_{Flutter}^2 = \frac{3a\,EI\sqrt{M^2-1}}{8\rho}\sqrt{\frac{(\omega_{0,1}^2 + \omega_{0,2}^2)}{4} - \omega_{0,1}^2\omega_{0,2}^2}}$$

d) Determine the flutter speed w/ three-mode expansion

Go back to page 1 and grab last eq:

$$\sum_{n=1}^{N} C_n \left(-\frac{\omega^2 \rho}{EI} + \left(\frac{\pi n}{a}\right)^4 - \frac{N_x}{EI}\left(\frac{\pi n}{a}\right)^2\right) \sin\left(\frac{\pi n x}{a}\right) + \sum_{n=1}^{N} C_n \left(\frac{\pi n}{a}\right)\underbrace{\left(\frac{2q}{EI\sqrt{M^2-1}}\right)}_{\lambda} \cos\left(\frac{\pi n x}{a}\right) = 0$$

$$\omega_{0,1}^2 = \left(\frac{\pi}{a}\right)^4 - \frac{N_x}{EI}\left(\frac{\pi}{a}\right)^2$$

$$\omega_{0,2}^2 = \left(\frac{2\pi}{a}\right)^4 - \frac{N_x}{EI}\left(\frac{2\pi}{a}\right)^2 \quad \Big\} \text{ Old}$$

$$\omega_{0,3}^2 = \left(\frac{3\pi}{a}\right)^4 - \frac{N_x}{EI}\left(\frac{3\pi}{a}\right)^2 \leftarrow \text{New}$$

We follow a similar procedure as last time. I coded this to make it automatic.

Expand Eq out by $N = 3$

1) Multiply by $\sin\left(\frac{\pi x}{a}\right)$ and integrate
2) Multiply by $\sin\left(\frac{2\pi x}{a}\right)$ and integrate
3) Multiply by $\sin\left(\frac{3\pi x}{a}\right)$ and integrate

eqs ↓ [$n = 1, 2, 3 \rightarrow$] ← how matrix was built

Matlab gives me the following:

$$\begin{bmatrix} AA & -4\lambda/3 & 0 \\ 4\lambda/3 & BB & -12\lambda/5 \\ 0 & 12\lambda/5 & CC \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$AA = \frac{-\rho a^4\omega^2 - N_x a^2\pi^2 + EI\pi^4}{2EIa^3}$$

$$BB = \frac{-\rho a^4\omega^2 - 4N_x a^2\pi^2 + 16EI\pi^4}{2EIa^3} \qquad 4 \to 2\cdot 2 \quad 16 \to 2\cdot 2\cdot 2\cdot 2$$

$$CC = \frac{-\rho a^4\omega^2 - 9N_x a^2\pi^2 + 81EI\pi^4}{2EIa^3} \qquad 9 \to 3\cdot 3 \quad 81 \to 3\cdot 3\cdot 3\cdot 3$$

Will only show work for first one since the others follow same pattern.

$$\frac{a}{2}\left(\frac{-\rho a^{\cancel{3}}\omega^2 - N_x a^{\cancel{2}1}\pi^2 + EI\pi^{\cancel{4}1}/a}{\cancel{2}EIa^{\cancel{3}}}\right)$$

$$\frac{a}{2}\left(-\frac{\rho\omega^2}{EI} - \underbrace{\frac{N_x}{EI}\frac{\pi^2}{a^2} + \left(\frac{\pi}{a}\right)^4}_{\omega_{0,1}^2}\right)$$

$$AA = \frac{a}{2}\left(\omega_{0,1}^2 - \frac{\rho\omega^2}{EI}\right) \quad BB = \frac{a}{2}\left(\omega_{0,2}^2 - \frac{\rho\omega^2}{EI}\right)$$

$$CC = \frac{a}{2}\left(\omega_{0,3}^2 - \frac{\rho\omega^2}{EI}\right)$$

Define $\Omega = \frac{a}{2}\left(\frac{\rho\omega^2}{EI}\right)$

Rewrite matrix:

$$\begin{bmatrix} (a/2)\omega_{0,1}^2 - \Omega & -4\lambda/3 & 0 \\ 4\lambda/3 & (a/2)\omega_{0,2}^2 - \Omega & -12\lambda/5 \\ 0 & 12\lambda/5 & (a/2)\omega_{0,3}^2 - \Omega \end{bmatrix}$$

$$(A - \lambda I) = 0$$

The eigenvalue problem

After adjusting the Matlab matrix to be more simplified, I assumed all the values were real... well actually done in beginning, but establishing now.

I then solved for the eigenvalue with

$$\det(A - I\lambda) = 0$$

The determinant written out by hand looks like:

$$\left((a/2)\omega_{0,1}^2 - \Omega^2\right)\left[\left((a/2)\omega_{0,2}^2 - \Omega^2\right)\left((a/2)\omega_{0,3}^2 - \Omega^2\right) + \left(\frac{12\lambda}{5}\right)^2\right] + \left(\frac{4\lambda}{3}\right)^2\left((a/2)\omega_{0,3}^2 - \Omega^2\right) = 0$$

→ It looks a lot messier in Matlab, but there's not good way to solve for Ω^2 by hand... at least easily

→ Defined syms variables as real.

→ Solve where frequencies collide. Set last two eigenvalues equal to eachother, and solved for λ.

→ Code gave 2 sets of complex and 1 set of real values for λ. Checked both real values when $\forall\lambda$ to make sure velocity term would not go imaginary in last step.

→ Left matlab output as just λ. Next page shows final calc.

→ Matlab's output is way to long to write down. I tried many different ways to get it to condense... gave up. I tried to write out the first few lines in the code because matlab would not publish the full output.
↳ Check code itself for first part of output

~~$2q$~~ $\leftarrow \frac{1}{2}\rho V^2$

~~$EI\sqrt{M^2-1}$~~

$$\frac{\cancel{2}\,V_{Flutter}^2}{\cancel{EI\sqrt{M^2-1}}} = \lambda_{new}\frac{EI\sqrt{M^2-1}}{\rho}$$

$$\boxed{V_{Flutter}^2 = \lambda_{new}\frac{EI\sqrt{M^2-1}}{\rho}}$$

→ Plug in λ_{new} solved for prior page

Compare results:

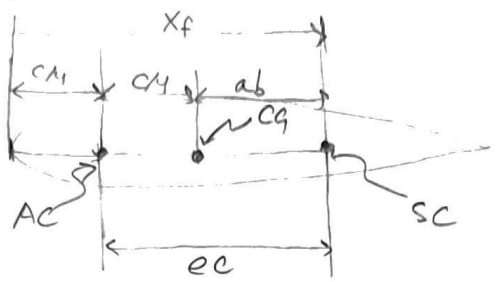
→ $n=1,2,3$ was much harder to solve for than $n=1,2$

→ $n=1,2,3$ is probably more accurate since it has a third expansion. However it does not feel realistic to use this number of expansions if not plugging in numbers. The output is long and has 0 meaning without numbers.

→ $n=1,2$ for quick design
$n=1,2,3$ for graphing and more accurate

→ I am curious <u>how</u> much more accurate $n=1,2,3$ is than $n=1,2$. There is probably a way to do it mathematically without numbers, but graphing and comparing would be more fun and intuitive.

Problem No. 2



→ Note: Mass/area ⇒ m not ρ... confusing

$$q = \frac{1}{2}\rho V^2$$

$$dL = q c a_w \left(\theta(y,t) + \frac{\dot{h}(y,t)}{V} + \frac{\dot{P}(t)}{V}\right) dy$$

$$dM = q c^2 \left(e a_w \left(\theta(y,t) + \frac{\dot{h}(y,t)}{V} + \frac{\dot{P}(t)}{V}\right) + M_{\dot{\theta}} \frac{\dot{\theta}(y,t)c}{4V}\right) dy$$

Givens

- $s = 7.5$ m
- $c = 2$ m
- $x_f = 0.48\,c$
- $x_{cg} = 0.5\,c$
- $m' = 200$ Kg/m²
- $EI = 2 \times 10^7$ N m²
- $GJ = 2 \times 10^6$ N m²
- $a_w = 2\pi$
- $\rho = 1.225$ Kg/m³
- $M_{\dot{\theta}} = -1.2$

Solve for e → $e = \frac{SC - AC}{c} = \frac{x_f - \frac{1}{4}c}{c}$ $0.48 - 0.25$

$e = 0.23$ → Assume one wing ⇒ see "A fixed root wing"

a) $z(x,y,t) = (y/s)^2 h_1(t) + (x - x_f)(y/s)\theta_1(t) + P(t)$

i) Solve for equations of motion w/ modal coordinates $\begin{bmatrix} h_1(t) \\ \theta_1(t) \\ P(t) \end{bmatrix}$

I wrote code from hw 4 that derives everything for me... so I will explain the concepts that go into this.

Need to solve for the total Kinetic energy of system. Remember, only solving for one dim.

$$T = \frac{1}{2}M\dot{p}^2 + \frac{1}{2}\int_0^s\int_0^c m(\dot{z})^2\,dx\,dy$$

Mass of fuselage... this needs to be added to the Kinetic energy

not much changed here other than the $\dot{z}$ def has the added term

U did not change at all since dependent on bendiy and torsion only

$$U = \frac{1}{2}\int_0^s EI\left(\frac{\partial^2 h}{\partial y^2}\right)^2 dy + \frac{1}{2}\int_0^s GJ\left(\frac{\partial\theta}{\partial y}\right)^2 dy$$

The δW was different now too.

$$\delta W = -\int_0^s dL\left((y/s)^2\delta h + \delta p\right)dy + \int_0^s dM\left((y/s)\delta\theta\right)dy$$

Once these parts are all solved for by my Matlab code, I solve for each equation and save it into a vector.

Lagrange's Eq:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} + \frac{\partial U}{\partial q_i} = \frac{\partial(\delta W)}{\partial(\delta q_i)}$$

Once I get an equation for each mode, I simply take their Jacobian w.r.t

$$\begin{bmatrix}\ddot{h}_i\\ \ddot{\theta}_i\\ \ddot{p}_i\end{bmatrix}, \quad \begin{bmatrix}\dot{h}_i\\ \dot{\theta}_i\\ \dot{p}_i\end{bmatrix}, \quad \begin{bmatrix}h_i\\ \theta_i\\ p_i\end{bmatrix}$$

$\Uparrow$ M $\quad$ $\Uparrow$ C $\quad$ $\Uparrow$ K

My code gives the following matrices:

$$M = \begin{bmatrix} m\frac{sc}{5} & m\frac{sc}{8}(c-2x_f) & m\frac{sc}{3} \\ m\frac{sc}{8}(c-2x_f) & m\frac{sc}{9}(c^2-3cx_f+3x_f^2) & m\frac{sc}{4}(c-2x_f) \\ m\frac{sc}{3} & m\frac{sc}{4}(c-2x_f) & msc + M \end{bmatrix}$$

$$C = \begin{bmatrix} \frac{a_w c q s}{5V} & 0 & \frac{a_w c q s}{3V} \\ \frac{-a_w c^2 e q s}{4V} & -\frac{M_\dot{o} c^3 q s}{12V} & -\frac{a_w c^2 e q s}{2V} \\ \frac{a_w c q s}{3V} & 0 & \frac{a_w c q s}{V} \end{bmatrix}$$

$$K = \begin{bmatrix} \frac{4EI}{s^3} & \frac{a_w c q s}{4} & 0 \\ 0 & \frac{GJ}{s} - \frac{a_w c^2 e q s}{3} & 0 \\ 0 & \frac{a_w c q s}{2} & 0 \end{bmatrix}$$

$$M\ddot{x} + C\dot{x} + Kx = 0$$

→ Matches the matrices found on page 200 of text (after some subs)

$q = \frac{1}{2}\rho V^2 \qquad b_w = a_w e$

→ Code will also print out the symbolic matrices (at end of print out... couldn't fix)

ii) The first step to graphing is to solve for the plant of the system, A.

$M\ddot{x} + C\dot{x} + Kx = 0 \rightarrow M\ddot{x} = -C\dot{x} - Kx$

$\ddot{x} = -M^{-1}C\dot{x} - M^{-1}Kx$

$y = \dot{x} \rightarrow \dot{y} = \ddot{x}$

$\dot{y} = -M^{-1}Cy - M^{-1}Kx$

$\ddot{x} = [0 \quad I]\,\dot{y}$

$$\dot{y} = \underbrace{\begin{bmatrix} [0] & [I] \\ -M^{-1}K & -M^{-1}C \end{bmatrix}}_{A} y$$

This part of the homework has two parts to it. Will define M now.

Part 1 $\Rightarrow$ $M = msc$

Part 2 $\Rightarrow$ $M = 10msc$

For both of these, I am creating

a) Frequency vs Flow

b) Damping vs Flow

c) Flutter Speed

Refer to Matlab handout for graphs.

$M = msc$

$\boxed{V_F = 77.13 \text{ m/s}}$

$\Delta V = 0.1$ m/s

$M = 10msc$

$\boxed{V_F = 81.67 \text{ m/s}}$

The flutter speed increases as we increase the mass of the fuselage.

Note

→ These are different than what book says... but I think book changed some parameters without saying. I spent extensive time making sure the matrices were correct, and checked the damping/frequency part of the code against other examples.

b) $$z(x,y,t) = \left(\frac{y}{s}\right)^2 h_1(t) + \left(\frac{y}{s}\right)^3 h_2(t) + (x - x_f)\left(\frac{y}{s}\right)\theta_1(t) + (x - x_f)\left(\frac{y}{s}\right)^2 \theta_2(t) + P(t)$$

We have the modal coordinates

$$\begin{bmatrix} h_1(t) \\ h_2(t) \\ \theta_1(t) \\ \theta_2(t) \\ P(t) \end{bmatrix}$$

The previous section went into the calculations for EOM. Will note changes...

For the Jacobian part

$$\begin{bmatrix} \ddot{h}_1 \\ \ddot{h}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{P} \end{bmatrix}, \quad \begin{bmatrix} \dot{h}_1 \\ \dot{h}_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{P} \end{bmatrix}, \quad \begin{bmatrix} h_1 \\ h_2 \\ \theta_1 \\ \theta_2 \\ P \end{bmatrix}$$

$\Uparrow$ M $\quad$ $\Uparrow$ C $\quad$ $\Uparrow$ K

Not a lot of changes...

The Equations of motion:

$$M=\begin{bmatrix} m\frac{cs}{5} & m\frac{cs}{6} & m\frac{cs}{8}(c-2x_f) & m\frac{cs}{10}(c-2x_f) & m\frac{cs}{3} \\ m\frac{cs}{6} & m\frac{cs}{7} & m\frac{cs}{10}(c-2x_f) & m\frac{cs}{12}(c-2x_f) & m\frac{cs}{4} \\ m\frac{cs}{8}(c-2x_f) & m\frac{cs}{10}(c-2x_f) & m\frac{cs}{9}(c^2-3cx_f+3x_f^2) & m\frac{cs}{12}(c^2-3cx_f+3x_f^2) & m\frac{cs}{4}(c-2x_f) \\ m\frac{cs}{10}(c-2x_f) & m\frac{cs}{12}(c-2x_f) & m\frac{cs}{12}(c^2-3cx_f+3x_f^2) & m\frac{cs}{15}(c^2-3cx_f+3x_f^2 & m\frac{cs}{6}(c-2x_f) \\ m\frac{cs}{3} & m\frac{cs}{4} & m\frac{cs}{4}(c-2x_f) & m\frac{cs}{6}(c-2x_f) & M+cms \end{bmatrix}$$

$$C=\begin{bmatrix} \frac{a_w c q s}{5V} & \frac{a_w c q s}{6V} & 0 & 0 & \frac{a_w c q s}{3V} \\ \frac{a_w c q s}{6V} & \frac{a_w c q s}{7V} & 0 & 0 & \frac{a_w c q s}{4V} \\ -\frac{a_w c^2 e q s}{4V} & -\frac{a_w c^2 e q s}{5V} & -\frac{M_\oplus c^3 q s}{12V} & -\frac{M_\oplus c^3 q s}{16V} & -\frac{a_w c^2 e q s}{2V} \\ -\frac{a_w c^2 e q s}{5V} & -\frac{a_w c^2 e q s}{6V} & -\frac{M_\oplus c^3 q s}{16V} & -\frac{M_\oplus c^3 q s}{20V} & -\frac{a_w c^2 e q s}{3V} \\ \frac{a_w c q s}{3V} & \frac{a_w c q s}{4V} & 0 & 0 & \frac{a_w c q s}{V} \end{bmatrix}$$

→ Code also prints out these symbolic matrices

→ $M\ddot{x}+C\dot{x}+Kx=0$

$$K=\begin{bmatrix} \frac{4EI}{s^3} & \frac{6EI}{s^3} & \frac{a_w c q s}{4} & \frac{a_w c q s}{5} & 0 \\ \frac{6EI}{s^3} & \frac{12EI}{s^3} & \frac{a_w c q s}{5} & \frac{a_w c q s}{6} & 0 \\ 0 & 0 & \frac{GJ}{s}-\frac{a_w c^2 e q s}{3} & \frac{GJ}{s}-\frac{a_w c^2 e q s}{4} & 0 \\ 0 & 0 & \frac{GJ}{s}-\frac{a_w c^2 e q s}{4} & \frac{4GJ}{3s}-\frac{a_w c^2 e q s}{5} & 0 \\ 0 & 0 & \frac{a_w c q s}{2} & \frac{a_w c q s}{3} & 0 \end{bmatrix}$$

The next part is where we use the two different definitions of M

Solve for

a) Frequency vs Flow
b) Damping vs Flow
c) Flutter Speed

Refer to Matlab handout for graphs.

$M = msc$

$\boxed{V_F = 79.38 \text{ m/s}}$

$\Delta V = 0.1 \text{ m/s}$

$M = 10msc$

$\boxed{V_F = 81.25 \text{ m/s}}$

c) As expected, for both parts, the flutter speed increased as the fuselage mass increased.

For $M = msc$, the flutter speed increased as we added modes. This was the behavior I was expecting for both cases. However for $M = 10msc$, with more modes added, the speed went down by ~0.4m/s. This surprised me, but what it might suggest that as the mass of the fuselage increases, the sensitivity to added modes might matter less. For $M = msc$, there was a ~2 m/s change between the two. It also might suggest that when a fuselage is added, that it more important to have more modes as the fuselage mass increases. The fact the flutter speed was less is alarming because you want the more accurate answer to be faster... not slower.

Final Exam - Aeroelasticity - ME597/AAE556

Victoria Nagorski - 12/12/22

Contents

Problem No. 1 Part a

```
clear;clc;close all;
syms x a omega rho EI lambda omega1 omega2 Omega c1

% Solve for the integrals
A1 = int(sin(pi*x/a)^2,x,0,a);
B1 = int(sin(pi*x/a)*sin(2*pi*x/a),x,0,a);
C1 = int(sin(pi*x/a)*cos(pi*x/a),x,0,a);
D1 = int(sin(pi*x/a)*cos(2*pi*x/a),x,0,a);

A2 = int(sin(2*pi*x/a)*sin(pi*x/a),x,0,a);
B2 = int(sin(2*pi*x/a)^2,x,0,a);
C2 = int(sin(2*pi*x/a)*cos(pi*x/a),x,0,a);
D2 = int(sin(2*pi*x/a)*cos(2*pi*x/a),x,0,a);

% Create the matrix
Temp1 = -omega^2*rho/EI + omega1^2;
Temp2 = -omega^2*rho/EI + omega2^2;
Matrix = [(Temp1*A1 + lambda*(pi/a)*C1) (Temp2*B1 + lambda*(2*pi/a)*D1);
          (Temp1*A2 + lambda*(pi/a)*C2) (Temp2*B2 + lambda*(2*pi/a)*D2)]

A = [(a/2)*omega1^2 -(4/3)*lambda;
     (4/3)*lambda    (a/2)*omega2^2];
determ = det(A - eye(2)*Omega);
OMEGA = solve(determ == 0,Omega)
Small_Omega = OMEGA(1);

% Guess
c2 = 1;

% Ax = Omega*x (Solve for eigenvector)
left = A*[c1;c2];
right = Small_Omega*[c1;c2];

% Solve for c1
C1 = solve(left(1) == right(1), c1);

% Check bottom
Left = A*[C1;c2];
Right = Small_Omega*[C1;c2];

Bottom = double(simplify(Left(2)-Right(2)))
if Bottom == 0
    fprintf('c1 and c2 are correct\n')
else
    fprintf('Not correct\n')
end
```

```
Matrix =

[(a*(omega1^2 - (omega^2*rho)/EI))/2,                      -(4*lambda)/3]
[                      (4*lambda)/3, (a*(omega2^2 - (omega^2*rho)/EI))/2]


OMEGA =

(a*omega1^2)/4 - (-((- 3*a*omega1^2 + 3*a*omega2^2 + 16*lambda)*(3*a*omega1^2 - 3*a*omega2^2 + 16*lambda))/36)^(1/2)/2 + (a*omega2^2)/4
(-((- 3*a*omega1^2 + 3*a*omega2^2 + 16*lambda)*(3*a*omega1^2 - 3*a*omega2^2 + 16*lambda))/36)^(1/2)/2 + (a*omega1^2)/4 + (a*omega2^2)/4


Bottom =

     0

c1 and c2 are correct
```

Problem No. 1 Part d

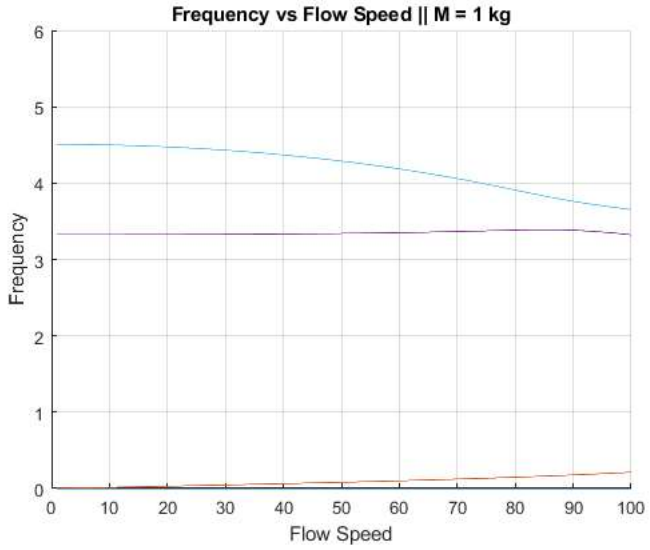
```
clear;clc;close all;
syms x a omega EI lambda omega1 omega2 omega3 rho N Nx OMEGA Omega real
% Solve for the matrix
n = 3;

% Define equation
start = (-omega^2*rho/EI + (pi*N/a)^4 - Nx/EI*(pi*N/a)^2)*sin(pi*N*x/a) + ...
    (pi*N/a)*lambda*cos(pi*N*x/a);

% Loop over the equation
eqs = sym(zeros(n,n));                  % Initialize the equation matrix
for int_vert = 1:n
    temp = sin(int_vert*pi*x/a);        % Define new sine for new line
    for int_hor = 1:n
        temp2 = subs(start,N,int_hor);
        temp3 = int(temp*temp2,x,0,a);  % Define each c_N part
        eqs(int_vert,int_hor) = temp3;
    end
end

% Clean up A matrix
A = eqs;
A(1,1) =  (a/2)*omega1^2;
A(2,2) =  (a/2)*omega2^2;
A(3,3) =  (a/2)*omega3^2;

% Solve for derminant
determ = det(A-(eye(n).*Omega))
warning('off','all')            % Keep publish clean

% Solve for eigenvalues
OMEGA = solve(determ == 0,Omega,"MaxDegree",3);

% Set two eigenvalues equal (where they collide)
LAMBDA = solve(OMEGA(2)==OMEGA(3),lambda,"MaxDegree",3);
warning('on','all')

% Pull out eigenvalues without imaginary numbers in them
LAMBDA3 = LAMBDA(3);
LAMBDA4 = LAMBDA(4);

% Make sure they don't become imaginary when square-rooted
test1 = sqrt(LAMBDA3);
test2 = sqrt(LAMBDA4);

string(test1);
strfind(ans,'i');

string(test2);
temp = strfind(ans,'i');

% Make sure no imaginary numbers
if isempty(temp) == 1
    fprintf('Velocity is not imaginary\n')
else
    fprintf('Velocity is imaginary\n')
end

fprintf('The lambda term is:\n')
lambda = string((-(75*((11428703*a^3*omega1^6 - 1191016*a^3*omega2^6 + ...
    2703959*a^3*omega3^6 + 4617996*a^3*omega1^2*omega2^4 - ...
    17559642*a^3*omega1^4*omega2^2 + 3784821*a^3*omega1^2*omega3^4 - ...
    16726467*a^3*omega1^4*omega3^2 - 11896698*a^3*omega2^2*omega3^4 - ...
    1044948*a^3*omega2^4*omega3^2 + 28620*3^(1/2)*abs(omega1^2 - ...
    omega3^2)*abs(a)^3*(50463*omega1^8 - 150845*omega1^6*omega2^2 - ...
    51007*omega1^6*omega3^2 + 144273*omega1^4*omega2^4 + ...
    163989*omega1^4*omega2^2*omega3^2 - 5484*omega1^4*omega3^4 - ...
    52152*omega1^2*omega2^6 - 132090*omega1^2*omega2^4*omega3^2 - ...
    31899*omega1^2*omega2^2*omega3^4 + 14289*omega1^2*omega3^6 + ...
    11236*omega2^8 + 7208*omega2^6*omega3^2 + 55233*omega2^4*omega3^4 - ...
    26189*omega2^2*omega3^6 + 2975*omega3^8)^(1/2) + ...
    25883292*a^3*omega1^2*omega2^2*omega3^2)^(4/3) - ...
    2779189261*a^4*omega1^8 - 126247696*a^4*omega2^8 + ...
    168568979*a^4*omega3^8 + 11363*a^2*omega1^4*(11428703*a^3*omega1^6 - ...
    1191016*a^3*omega2^6 + 2703959*a^3*omega3^6 + 4617996*a^3*omega1^2*omega2^4 -...
    17559642*a^3*omega1^4*omega2^2 + 3784821*a^3*omega1^2*omega3^4 - ...
    16726467*a^3*omega1^4*omega3^2 - 11896698*a^3*omega2^2*omega3^4 - ...
    1044948*a^3*omega2^4*omega3^2 + 28620*3^(1/2)*abs(omega1^2 - omega3^2)*abs(a)^3*(50463*omega1^8 - 150845*omega1^6*omega2^2 - 51007*omega1^6*omega3^2 + 144273*om
```

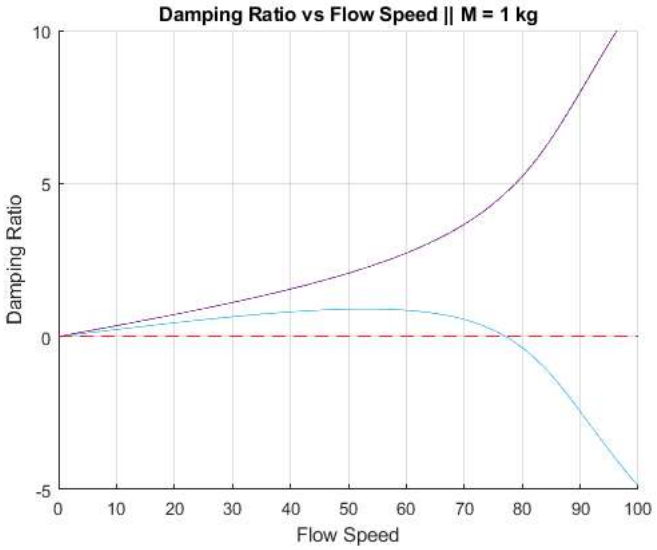
```
determ =

- Omega^3 + (Omega^2*a*omega1^2)/2 + (Omega^2*a*omega2^2)/2 + (Omega^2*a*omega3^2)/2 - (Omega*a^2*omega1^2*omega2^2)/4 - (Omega*a^2*omega1^2*omega3^2)/4 - (Omega*a^

Velocity is not imaginary
The lambda term is:

lambda =
```

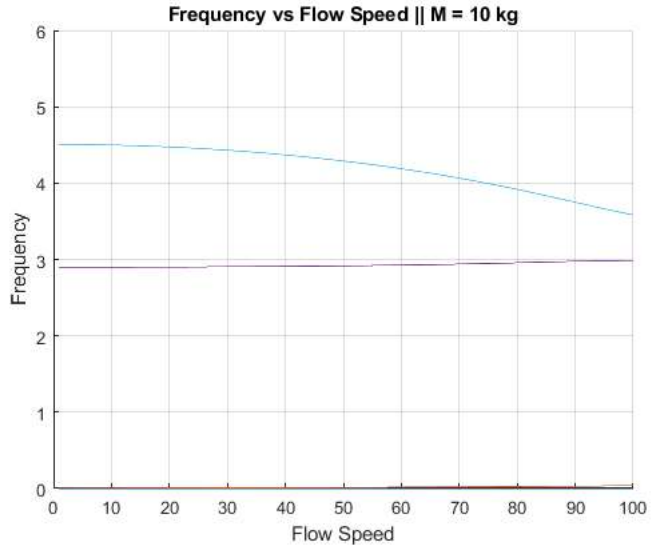
"(-(75*(11428703*a^3*omega1^6 - 1191016*a^3*omega2^6 + 2703959*a^3*omega3^6 + 4617996*a^3*omega1^2*omega2^4 - 17559642*a^3*omega1^4*omega2^2 + 3784821*a^3*omega

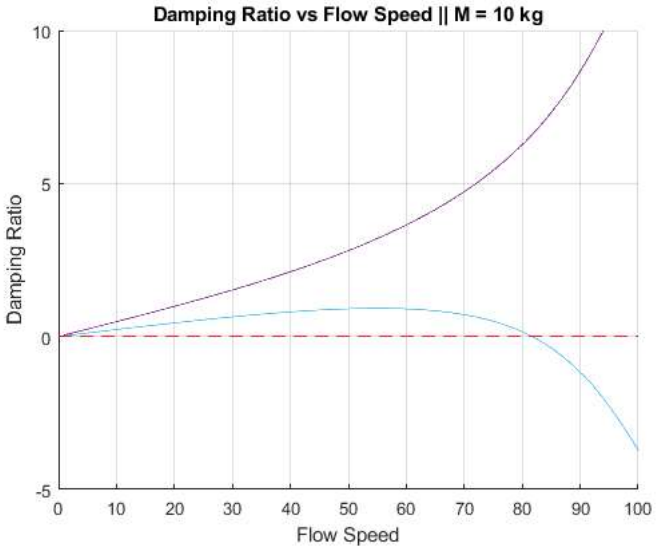
Problem No. 2 Part a

```
clear;clc;close all;
syms y s h(t) x xf theta(t) t EI GJ aw rho e M_thetad c m dh dtheta p(t) dp M

% Define the variables... The function explains variables better
mult = 1;
fuse_mass = mult*200*7.5*2;        % M = mult*m_wing
vars = [s;c;xf;m;EI;GJ;aw;rho;e;M_thetad;M];
nums = [7.5;2;0.48*2;200;2e7;2e6;2*pi;1.225;0.23;-1.2;fuse_mass];
bend_mode = (y/s)^2*h;
dB = (y/s)^2*dh;
tor_mode = (x-xf)*(y/s)*theta;
dT = (y/s)*dtheta;
plunge_mode = p;
modes = [h theta p];
dP = dp;
d = [dh dtheta dp];
damping = true;                    % Not set C = 0 with analysis?

[sys1,~] = FlexWing_Lagrange(vars,nums,bend_mode,tor_mode,damping,modes,dB,dT,d,plunge_mode,dP,mult);

% For publishing purposes
M1 = simplify(sys1.M);
K1 = simplify(sys1.K);
C1 = simplify(sys1.C);

display(M1);
display(K1);
display(C1);

% For 10*m_wing
% Define the variables
mult = 10;          % M = mult*m_wing
fuse_mass = mult*200*7.5*2;
nums = [7.5;2;0.48*2;200;2e7;2e6;2*pi;1.225;0.23;-1.2;fuse_mass];
[~,~] = FlexWing_Lagrange(vars,nums,bend_mode,tor_mode,damping,modes,dB,dT,d,plunge_mode,dP,mult);
```


Frequency vs Flow Speed || M = 1 kg
Frequency
Flow Speed
0
1
2
3
4
5
6
0
10
20
30
40
50
60
70
80
90
100


Damping Ratio vs Flow Speed || M = 1 kg
Damping Ratio
Flow Speed
-5
0
5
10
0
10
20
30
40
50
60
70
80
90
100





Problem No. 2 Part b

```
clear;clc;close all;
syms y s h1(t) h2(t) x xf theta1(t) theta2(t) t EI GJ aw rho e...
    M_thetad c m dh1 dh2 dtheta1 dtheta2 p(t) dp M

% Define the variables... The function explains variables better
mult = 1;           % M = mult*m_wing
fuse_mass = mult*200*7.5*2;
vars = [s;c;xf;m;EI;GJ;aw;rho;e;M_thetad;M];
nums = [7.5;2;0.48*2;200;2e7;2e6;2*pi;1.225;0.23;-1.2;fuse_mass];
bend_mode = (y/s)^2*h1 + (y/s)^3*h2;
dB = (y/s)^2*dh1 + (y/s)^3*dh2;
tor_mode = (x-xf)*(y/s)*theta1 + (x-xf)*(y/s)^2*theta2;
dT = (y/s)*dtheta1 + (y/s)^2*dtheta2;
plunge_mode = p;
modes = [h1 h2 theta1 theta2 p];
dP = dp;
d = [dh1 dh2 dtheta1 dtheta2 dp];
damping = true;                     % Not set C = 0 with analysis?

[sys2,~] = FlexWing_Lagrange(vars,nums,bend_mode,tor_mode,damping,modes,dB,dT,d,plunge_mode,dP,mult);

% For publishing purposes
M2 = simplify(sys2.M);
K2 = simplify(sys2.K);
C2 = simplify(sys2.C);

display(M2);
display(K2);
display(C2);

% For 10*m_wing
% Define the variables
mult = 10;          % M = mult*m_wing
fuse_mass = mult*200*7.5*2;
```

```
nums = [7.5;2;0.48*2;200;2e7;2e6;2*pi;1.225;0.23;-1.2;fuse_mass];

[~,~] = FlexWing_Lagrange(vars,nums,bend_mode,tor_mode,damping,modes,dB,dT,d,plunge_mode,dP,mult);
```

M2 =

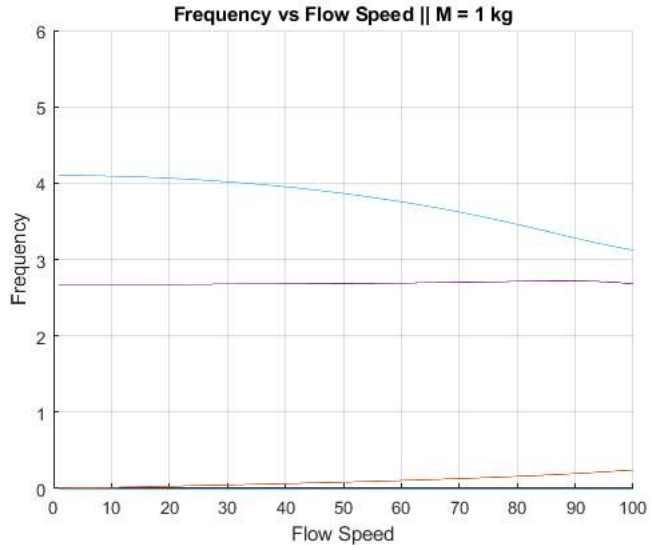
$$\begin{bmatrix} \frac{c m s}{5} & \frac{c m s}{6} & \frac{c m s (c - 2 xf)}{8} & \frac{c m s (c - 2 xf)}{10} & \frac{c m s}{3} \\ \frac{c m s}{6} & \frac{c m s}{7} & \frac{c m s (c - 2 xf)}{10} & \frac{c m s (c - 2 xf)}{12} & \frac{c m s}{4} \\ \frac{c m s (c - 2 xf)}{8} & \frac{c m s (c - 2 xf)}{10} & \frac{c m s (c^2 - 3 c xf + 3 xf^2)}{9} & \frac{c m s (c^2 - 3 c xf + 3 xf^2)}{12} & \frac{c m s (c - 2 xf)}{4} \\ \frac{c m s (c - 2 xf)}{10} & \frac{c m s (c - 2 xf)}{12} & \frac{c m s (c^2 - 3 c xf + 3 xf^2)}{12} & \frac{c m s (c^2 - 3 c xf + 3 xf^2)}{15} & \frac{c m s (c - 2 xf)}{6} \\ \frac{c m s}{3} & \frac{c m s}{4} & \frac{c m s (c - 2 xf)}{4} & \frac{c m s (c - 2 xf)}{6} & M + c m s \end{bmatrix}$$

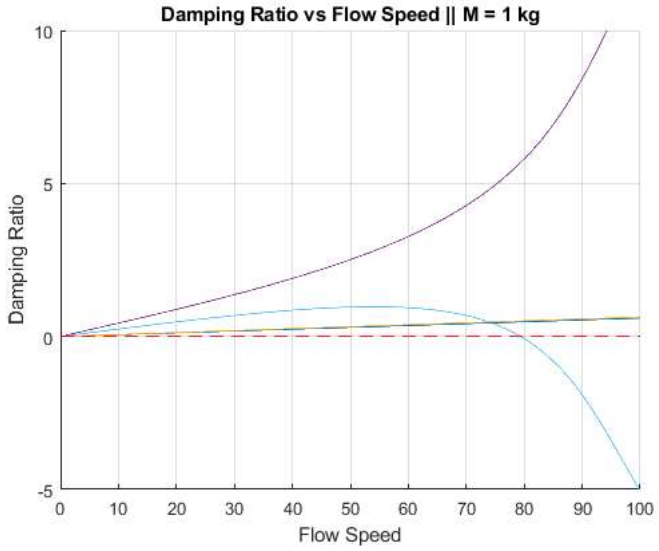
K2 =

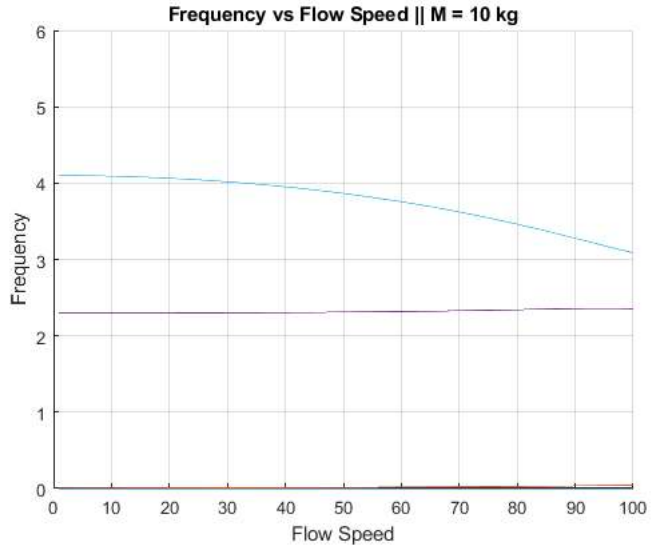
$$\begin{bmatrix} \frac{4 EI}{s^3} & \frac{6 EI}{s^3} & \frac{aw c q s}{4} & \frac{aw c q s}{5} & 0 \\ \frac{6 EI}{s^3} & \frac{12 EI}{s^3} & \frac{aw c q s}{5} & \frac{aw c q s}{6} & 0 \\ 0 & 0 & \frac{GJ}{s} - \frac{aw c^2 e q s}{3} & \frac{GJ}{s} - \frac{aw c^2 e q s}{4} & 0 \\ 0 & 0 & \frac{GJ}{s} - \frac{aw c^2 e q s}{4} & \frac{4 GJ}{3 s} - \frac{aw c^2 e q s}{5} & 0 \\ 0 & 0 & \frac{aw c q s}{2} & \frac{aw c q s}{3} & 0 \end{bmatrix}$$

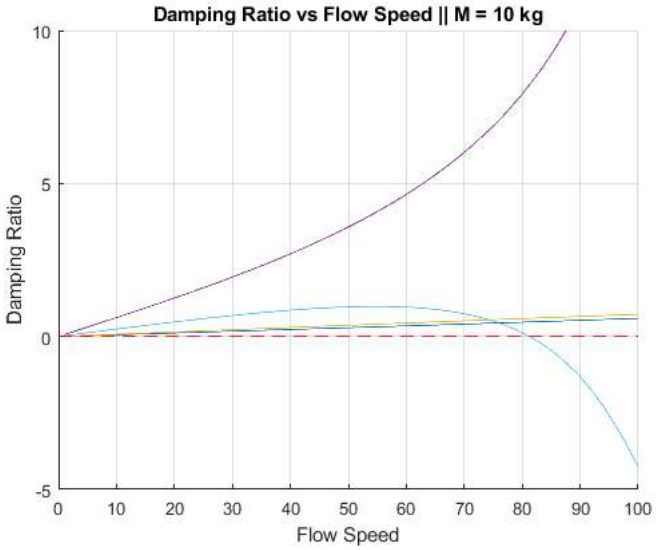
C2 =

$$\begin{bmatrix} \frac{aw c q s}{5 V} & \frac{aw c q s}{6 V} & 0 & 0 & \frac{aw c q s}{3 V} \\ \frac{aw c q s}{6 V} & \frac{aw c q s}{7 V} & 0 & 0 & \frac{aw c q s}{4 V} \\ -\frac{aw c^2 e q s}{4 V} & -\frac{aw c^2 e q s}{5 V} & -\frac{M_{thetad} c^3 q s}{12 V} & -\frac{M_{thetad} c^3 q s}{16 V} & -\frac{aw c^2 e q s}{2 V} \\ -\frac{aw c^2 e q s}{5 V} & -\frac{aw c^2 e q s}{6 V} & -\frac{M_{thetad} c^3 q s}{16 V} & -\frac{M_{thetad} c^3 q s}{20 V} & -\frac{aw c^2 e q s}{3 V} \\ \frac{aw c q s}{3 V} & \frac{aw c q s}{4 V} & 0 & 0 & \frac{aw c q s}{V} \end{bmatrix}$$









Dummy section because of local function

```
clear;clc;close all;
```

====================== Local Function ============================

```
function [sys,other] = FlexWing_Lagrange(var,nums,bend,tor,damping,mode,dB,dT,d,plunge,dP,mult)
% This function is for a flexible wing with damping
% Outputs:
%   - sys: Mass [M], Damper [C], Spring [K], and Plant [A]
%   - other: Damping and Frequencies
% Inputs:
%   - var: variables
%   - nums: values for the variables
%   - bend: bending modes definition
%   - tor: torsional modes definition
%   - damping: True or False whether aerodynamic damping is to be
%              considered
%   - mode: Modes to consider
%   - dB: delta bend
%   - dT: delta theta
%   - d: delta modes
%   - plunge: plunge mode defintion
%   - dP: delta plunge
%
% Start code below::
    syms t m x c y s EI GJ xf V rho q aw M_thetad e M
    % ============== Solve for Lagrange's equations ==============
    % Solve for kinetic energy
    part1 = diff(bend,t);     % Derivative w.r.t time of bending modes
    part2 = diff(tor,t);      % Derivative w.r.t time of torisional modes
    part3 = diff(plunge,t);   % Derivative w.r.t time of plunging modes
    together = (part1+part2+part3)^2;
    FuselageT = (1/2)*M*part3^2;
```

```
    T = simplify(m/2*int(int(together,x,0,c),y,0,s) + FuselageT);

    % Solve for potential energy
    U_pt1 = (1/2)*int(diff(bend,y,2)^2,y,0,s)*EI;
    U_pt2 = (1/2)*int(diff(tor/(x-xf),y)^2,y,0,s)*GJ;
    U = simplify(U_pt1 + U_pt2);

    % Solve for delta work
    wPart1 = q*c*aw * (tor/(x-xf) + diff(bend,t)/V + diff(plunge,t)/V)*(dP + dB);
    wPart2 = q*c^2*(e*aw*(tor/(x-xf) + diff(bend,t)/V + diff(plunge,t)/V) + M_thetad*(diff(tor/(x-xf),t)*c/(4*V)))*dT;
    dW = -int(wPart1,y,0,s) + int(wPart2,y,0,s);

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

    % Create the matrices
    sys.M = simplify(jacobian(eq,diff(mode,t,2)));      % Mass
    sys.K = simplify(jacobian(eq,mode));                % Spring
    if damping == false
        sys.C = zeros(length(mode),length(mode));
    else
        sys.C = simplify(jacobian(eq,diff(mode,t)));    % Damper
    end
    sys.A = [zeros(length(mode),length(mode))    eye(length(mode));
                    -inv(sys.M)*sys.K           -inv(sys.M)*sys.C];

    % ============= Solve for Frequency and Damping ==============
    A = simplify(subs(sys.A,q,(1/2)*rho*V^2));
    A = subs(A,var,nums);

    % Plug in velocities
    Velocities = 1:0.1:100;
    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,V,v));
        p = eig(Temp);
        wrad = abs(p);
        zeta = -real(p) ./ wrad;
        if imag(p) == 0
            wrad = 0;
        end
        whz = wrad ./ (2*pi);

        [whz, orderf] = sort(whz);

        other.frequencies(:,i) = whz;
        other.damping(:,i) = zeta(orderf);
    end

    % Plot everything
    figure
    hold on
    for i = 1:(length(mode)*2)
         plot(Velocities,other.frequencies(i,:))
    end
    title(['Frequency vs Flow Speed || M = ',num2str(mult),' kg'])
    xlabel('Flow Speed')
    ylabel('Frequency')
    xlim([0 100])
    ylim([0 6])
    grid('on')
    hold off

    figure
    hold on
    for i = 1:(length(mode)*2)
         plot(Velocities,other.damping(i,:).*100)
    end
    title(['Damping Ratio vs Flow Speed || M = ',num2str(mult),' kg'])
    xlabel('Flow Speed')
    ylabel('Damping Ratio')
    xlim([0 100])
    ylim([-5 10])
    line([0 100],[0 0],'Color','red','LineStyle','--')
    grid('on')
    hold off
end
```

```
M1 =
```

```
[          (c*m*s)/5,             (c*m*s*(c - 2*xf))/8,           (c*m*s)/3]
[(c*m*s*(c - 2*xf))/8, (c*m*s*(c^2 - 3*c*xf + 3*xf^2))/9, (c*m*s*(c - 2*xf))/4]
[          (c*m*s)/3,             (c*m*s*(c - 2*xf))/4,           M + c*m*s]


K1 =

[(4*EI)/s^3,           (aw*c*q*s)/4, 0]
[         0, GJ/s - (aw*c^2*e*q*s)/3, 0]
[         0,           (aw*c*q*s)/2, 0]


C1 =

[     (aw*c*q*s)/(5*V),                         0,      (aw*c*q*s)/(3*V)]
[-(aw*c^2*e*q*s)/(4*V), -(M_thetad*c^3*q*s)/(12*V), -(aw*c^2*e*q*s)/(2*V)]
[     (aw*c*q*s)/(3*V),                         0,          (aw*c*q*s)/V]
```

Published with MATLAB® R2022a