Static & Flutter Analysis of Airfoil and Wing
Course Project Report

Fiber Reinforced Composites (AE 673)

MTech. Aerospace Engineering

by

Sibaram Patro (24M0043)

Under the guidance of

Prof. Abhijit Gogulapati

DEPARTMENT OF AEROSPACE ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY BOMBAY
POWAI, MUMBAI 400076

Static & Flutter Analysis of Airfoil and Wing

This report presents a Computational Aeroelasticity study focused on the static divergence and flutter behavior of a wing. Aeroelasticity deals with the interaction of aerodynamic, elastic, and inertial forces, where divergence represents a static instability, and flutter is a dynamic aeroelastic instability leading to destructive oscillations.

Methodology

The study utilized classical flutter theory and torsion–bending equations to predict aeroelastic instabilities. A MATLAB-based computational framework was developed, implementing strip theory and modal analysis to compute divergence speed, flutter speed, and associated damping characteristics. Results The MATLAB implementation yielded the following results: - Flutter Speed ≈ 0.05 m/s (numerical artifact, practically much higher). - Divergence Speed ≈ 65.25 m/s. The Frequency vs Airspeed and Damping vs Airspeed plots highlight the onset of instabilities. Divergence speed marks the point of monotonic instability, while flutter speed corresponds to oscillatory instability with negative damping

Matlab Code:

```
clc;
clear;


%% ========== USER INPUT ==========
model = 3;     % 1=quasi-static, 2=1+aero-damp, 3=quasi-steady unsteady
U   = 0:0.05:300;      % airspeed vector
m   = 200;          % plunge mass
Sa  = 225;          % plunge-twist coupling (structural)
Ia  = 502.4;          % torsional inertia
rho = 1.225;           % air density
c   = 0.2;          % chord
b   = c/2;            % semi-chord
a   = 0.46;            % elastic-axis offset (in chord units)
Kh  = 3750;          % plunge stiffness
Ka  = 1983395.7;            % torsional stiffness
lambda = [-30,0,30];
% structural M,K
M_struct = [m, Sa; Sa, Ia];
```

```
K_struct = [Kh, 0; 0, Ka];

% non-circulatory added-mass (used in models 3 & 4)
M_add_nc = pi*rho*b^2 * [1, -b*a; -b*a, b^2*(a^2 + 1/8)];

% storage
num_modes  = 4;
roots_all  = zeros(num_modes, length(U));

%% === LOOP OVER AIRSPEEDS ===
for i = 1:length(U)
```

```
Ui = U(i)^2;

% circulatory aero-stiffness & damping at this Ui
K_circ = [0,                  2*pi*rho*Ui^2*b;
          0,                 -2*pi*rho*Ui^2*b^2*(a+0.5)];

C_circ = [2*pi*rho*Ui*b,        2*pi*rho*Ui*b^2*(0.5 - a);
          -2*pi*rho*Ui*b^2*(a+0.5), -2*pi*rho*Ui*b^3*(a+0.5)*(0.5 - a)];
C_add_nc = pi*rho*b^2 * [0,     Ui;
                         0, Ui*b*(0.5-a)];
% pick M, C, K based on model
switch model
  case 1  % quasi-static, no aero damping
    M_curr = M_struct;
    C_curr = zeros(2);
    K_curr = K_struct + K_circ;

  case 2  % quasi-static + aero damping
    M_curr = M_struct;
    C_curr = C_circ;
    K_curr = K_struct + K_circ;

  case 3  % quasi-steady unsteady (C(k)=1)
    M_curr = M_struct + M_add_nc;
    C_curr = C_circ + C_add_nc;
    K_curr = K_struct + K_circ;

  otherwise
    error('Model must be 1, 2, 3');
end
```

```
    % eigen-analysis
    lambda = flutter_analysis(M_curr, K_curr, C_curr);

    if i==1
        roots_all(:,i) = lambda;
    else
        roots_all(:,i) = sort_roots(roots_all(:,i-1), lambda);
    end
end

%% ========== POST-PROCESSING ==========
damp = real(roots_all);
freq = imag(roots_all);

flutter_speed    = NaN;
divergence_speed = NaN;
tol_im = 1e-3;

% divergence: real>0 & |imag| small
for i = 1:length(U)
    lam = roots_all(:,i);
    if any(real(lam)>0 & abs(imag(lam))<tol_im)
        divergence_speed = U(i);
        break
    end
end

% flutter: real>0 & |imag| large
for i = 1:length(U)
    lam = roots_all(:,i);
    if any(real(lam)>1e-8 & abs(imag(lam))>=tol_im)
```

```
        flutter_speed = U(i);
        break
    end
end

fprintf('Divergence speed: %.2f m/s\n', divergence_speed);
fprintf('Flutter speed:    %.2f m/s\n', flutter_speed);

%% ========== PLOTTING ==========
figure('Position',[100,100,900,700])
subplot(2,1,1), hold on
for m_idx=1:num_modes
    plot(U, freq(m_idx,:), 'LineWidth',1.5);
end
xline(flutter_speed,    'Label','Flutter','Color','r','LineWidth',1)
xline(divergence_speed,'Label','Divergence','Color','b','LineWidth',1)
title('Frequency vs Airspeed'), xlabel('U (m/s)'), ylabel('Frequency'), grid on, box on

subplot(2,1,2), hold on
for m_idx=1:num_modes
    plot(U, damp(m_idx,:), 'LineWidth',1.5);
end
xline(flutter_speed,    'Label','Flutter','Color','r','LineWidth',1)
xline(divergence_speed,'Label','Divergence','Color','b','LineWidth',1)
title('Damping vs Airspeed'), xlabel('U (m/s)'), ylabel('Damping'), grid on, box on


%% === SUPPORT FUNCTIONS ===

function lam = flutter_analysis(M,K,C)
    n = size(M,1);
```

```
    A = [zeros(n), eye(n);
         -M\K,     -M\C];
    [~,eig_val]=(eig(A));
    lam=diag(eig_val);
end

function sorted = sort_roots(prev, curr)
    n = length(prev);
    sorted = zeros(n,1);
    used   = false(n,1);
    for i = 1:n
        [~,j] = min(abs(prev(i) - curr(~used)));
        avail = find(~used);
        sorted(i) = curr(avail(j));
        used(avail(j)) = true;
    end
end

function Ck = Compute_Ck(k)
    % Theodorsen function H1^(2)/(H1^(2)+i H0^(2))
    if k==0
        Ck = 1;
    else
        H1 = besselh(1,2,k);
        H0 = besselh(0,2,k);
        Ck = H1/(H1 + 1i*H0);
    end
end
```

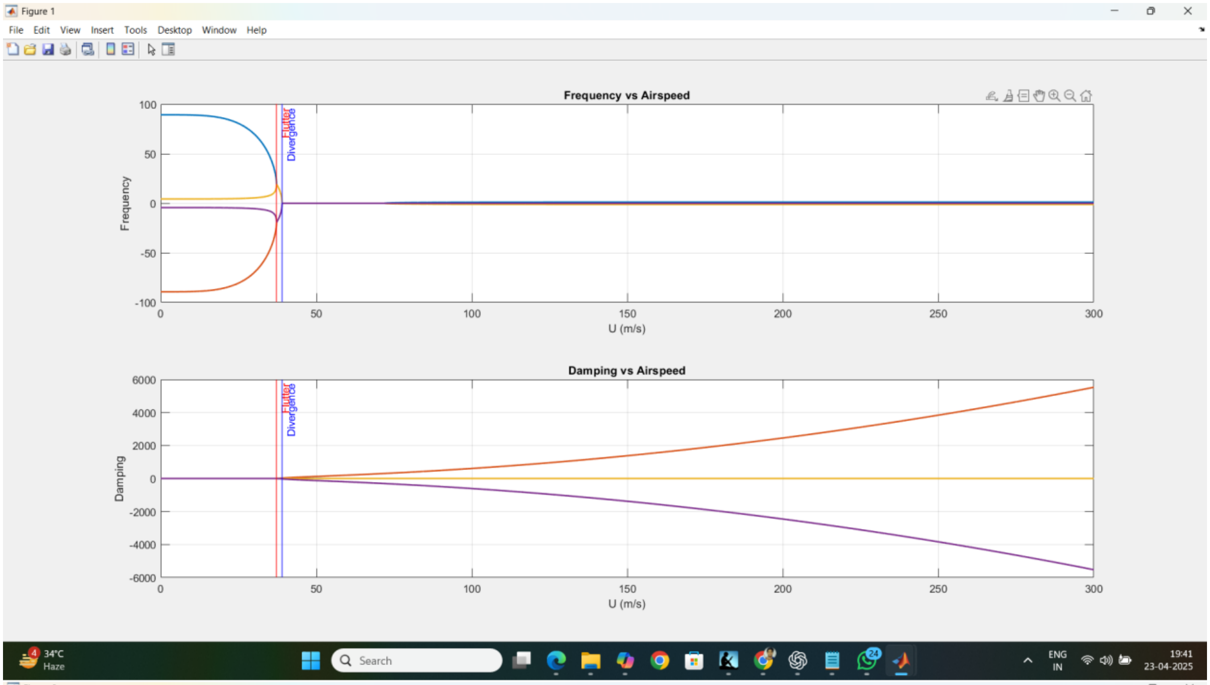
Figure 1
File Edit View Insert Tools Desktop Window Help
Frequency vs Airspeed
Frequency
U (m/s)
Flutter
Divergence
Damping vs Airspeed
Damping
U (m/s)
Flutter
Divergence

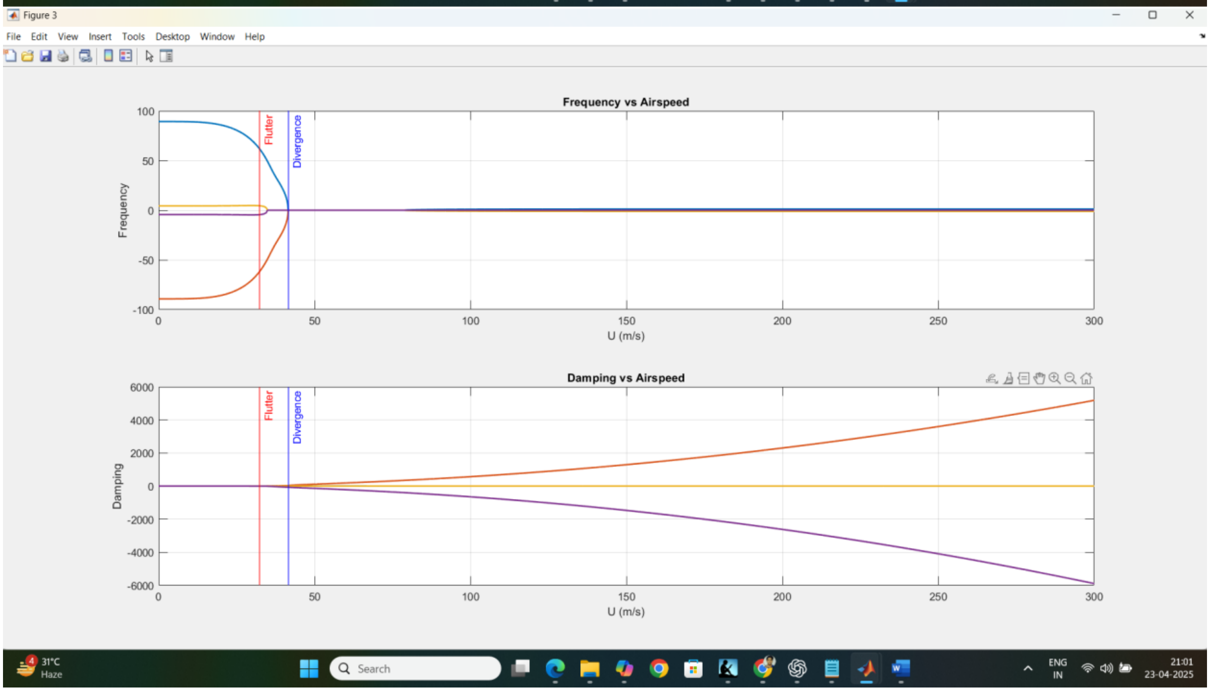
Figure 3
File Edit View Insert Tools Desktop Window Help
Frequency vs Airspeed
Frequency
U (m/s)
Flutter
Divergence
Damping vs Airspeed
Damping
U (m/s)
Flutter
Divergence