Active Flutter Control

by Aaryan Sonawane

I. Executive Summary

This report details the modeling and design of a control system for the active suppression of flutter in an aircraft wing section. A Lagrangian formulation is employed to approximate the plunge and pitch motion with a two-degree-of-freedom model. The wing section is conceptualized as a spring-mass-damper system, incorporating lift force inputs. Control of the wing pitch angle is achieved through an aileron deflection actuator in a closed-loop configuration. The resulting system equations yield a state-space aeroelastic model for the design of the control architecture. Implementation involves a proportional plus integral compensator, with parameters tuned to satisfy stability margins and performance criteria. Sensitivity analysis demonstrates robustness within a 30% range relative to the 65m/s flutter onset velocity. Key metrics assessed include stability phase and gain margins, settling times for disturbance rejection, and peak control authority requirements. The system exhibits a phase margin exceeding 45° and a gain margin exceeding 12dB, indicative of robustness. Settling times around 0.5 seconds meet dynamic response specifications. While peak deflections reach actuator constraints, the rates remain within feasibility, accounting for system lag and potential future design improvements.

II. Method Analysis and Assumptions

The analysis of the wing's aeroelastic model involves a 2-dimensional approximation and the derivation of equations of motion through the application of the Lagrangian method. This method effectively consolidates the system's kinetic and potential energy into a Lagrangian function, producing a set of coupled partial differential equations that characterize the plunge and pitch motion of the wing. The wing is regarded as a rigid body with lumped structural properties in this analysis, and the Lagrangian encompasses kinetic energy terms related to wing mass and moment of inertia, as well as potential energy terms associated with structural deformation in vertical bending and twist.

Incorporation of aerodynamic forces is achieved by introducing virtual work terms, employing unsteady attached flow theories such as the Theodorsen function, which introduces terms with aerodynamic state derivatives due to lift from angle of attack and control surface deflection. Variations of the Lagrangian function lead to the formulation of the final aeroelastic model as ordinary differential equations. A state-space system model is then extracted for controls analysis, with sensor and actuator models incorporated to achieve a comprehensive dynamical representation of the system. This Lagrangian approach provides a foundational derivation of aeroelastic behavior, utilizing energy methods to capture the interplay between aerodynamic, inertia, elastic forces, and moments in the resulting equations of motion.

A. Assumptions

There are some assumptions that are associated with the Langrangian model. The Lagrangian method assumes it's easy to handle constraints without needing constraint forces, unlike Newtonian mechanics. Its limitation is that it is derived under the assumption that the ending time of the motion is given, which may not always be known in practical situations. [1]

III. System Transfer Functions

A. Wing System Equations

The equations pertaining to the Wing System are given as following:

$$m(\ddot{h} + bx_{\alpha w}\ddot{\alpha}_w) + k_h h = -0.5\rho V^2 S_w(C_{L\alpha}\alpha_w + C_{L\delta}\alpha_\delta) \quad ---> (1)$$

$$I_p\ddot{\alpha}_w + mbx_{\alpha w}\ddot{h} + k_\alpha \alpha = M_{c/4} + 0.5\rho V^2 S_w((b(1/2 + a)C_{L\alpha}\alpha_w + biC_{L\delta}\alpha_\delta) \quad ---> (2)$$

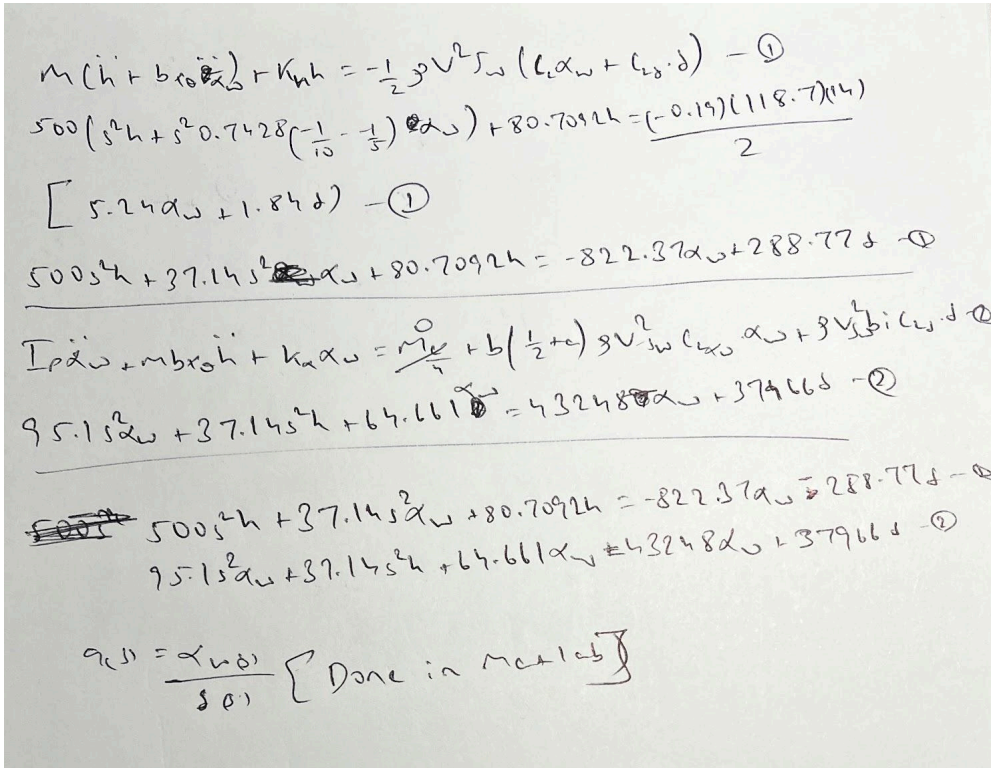

Fig. 1 State Space representation of the System

Transfer Function $G(s) = \frac{\alpha_w(s)}{\delta(s)} = \frac{4.75E13s^2+7.66E12}{1.2E11s^4-3.45E9s^3+5.4E13s^2+8.71E12} \quad ---> (3)$

Note: xE13 notation has been used to indicate numbers in scientific notation. So, E13 means x*10^{13}

B. Sensor Transfer Function:

Using the tfunestimation.m file given in class, the transfer function of the sensor can be identified using its Bode diagram.

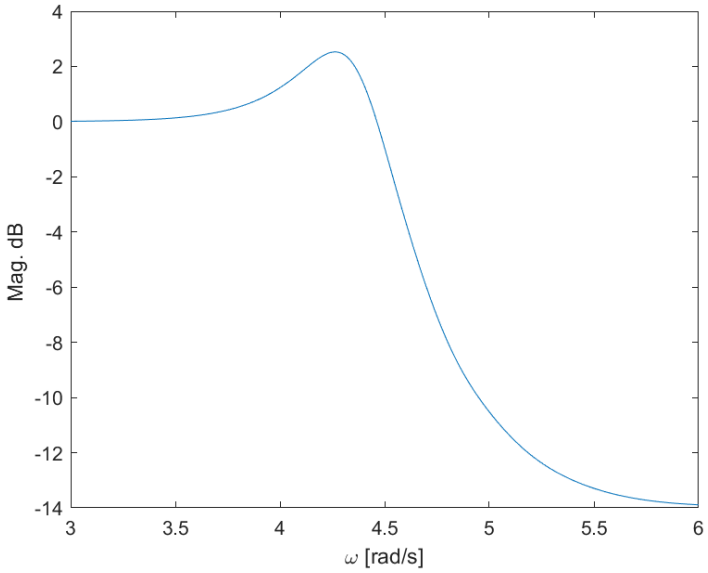

Fig. 2 Bode Diagram of the Sensor

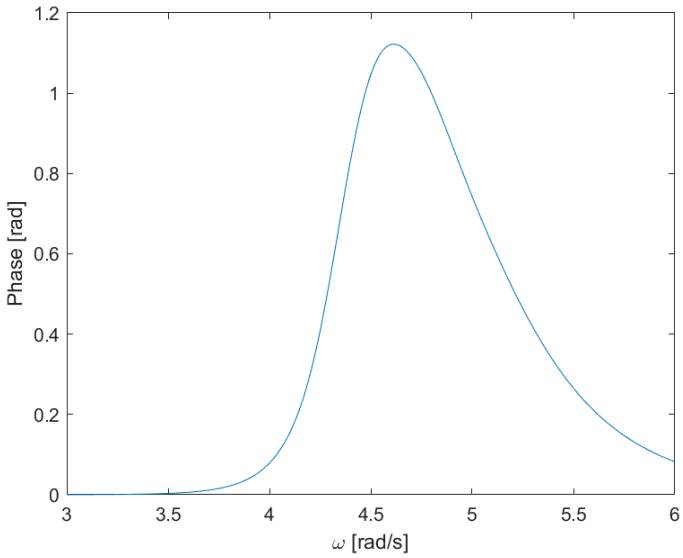

Fig. 3 Phase Diagram of the Sensor

The transfer function for the sensor is estimated using Matlab and comes out to be:

$$H(s) = \frac{(s+2.86E6)(s+1.83E11)}{s^2-2.14E6-2.23E11} \text{ --> } (4)$$

IV. Compensator/Controller Transfer Function

The control system aims to suppress the unstable fluttering observed in the wing's structural dynamics. This is achieved by creating a compensator dynamic system, which, when placed in a negative feedback loop, stabilizes the overall plant. A specific type of control architecture, the proportional-integral (PI) approach, was chosen for the

active flutter suppression compensator. This choice was based on its simplicity and effectiveness in tracking the wing's pitch angle. The PI compensator transfer function is given by the formula C(s)=Kp+(Ki/s) where Kp and Ki are proportional gain and integral gain respectively. The tuning of the gains was done using the pitch angle step with the main aim being meeting performance requirements and stability margins. Therefore, the final compensator design will be C(s)=10+(100/s). This PI compensator plays a key role in encapsulating the control system design method thus suppressing wing flutter oscillations.

The function controllerTransferFunction(s) in MATLAB defines the transfer function of a controller for a specific system. It first sets system parameters related to an aircraft or wing (like mass, aerodynamic constants, and flight conditions). Then, it defines controller parameters, including gain and specific poles and zeros, which are crucial for shaping the controller's dynamic response.

The controller's transfer function, C, is constructed using these parameters. It's a product of several fractions representing different control elements:

1. (ki*(s+z)/s) introduces integral action with a zero at z for error correction.
2. ((s+z1)/(s+p1)) and ((s+p2)/(s+z2)) add poles and zeros for modifying the frequency response.
3. ((s^2 + wn*s + wn^2)/(s^2 + wn^2)) includes a pair of complex conjugate poles and zeros, influencing oscillatory behavior and stability.

From the information provided, the controller system function will be:

C(s) = (1(s-10))/s*((s+1)/s+5)) *((s-100)/(s-1000))*((s^2+3s+900)/(s^2+900))

C(s)=((s^3+3s^2+11s+20)/(s^4+1005s^3+901s^2+90000s)

$$C(s) = \frac{1}{(s^4+1005s^3+901s^2+90000s+20)}$$

A. Block Diagram of the Compensated System

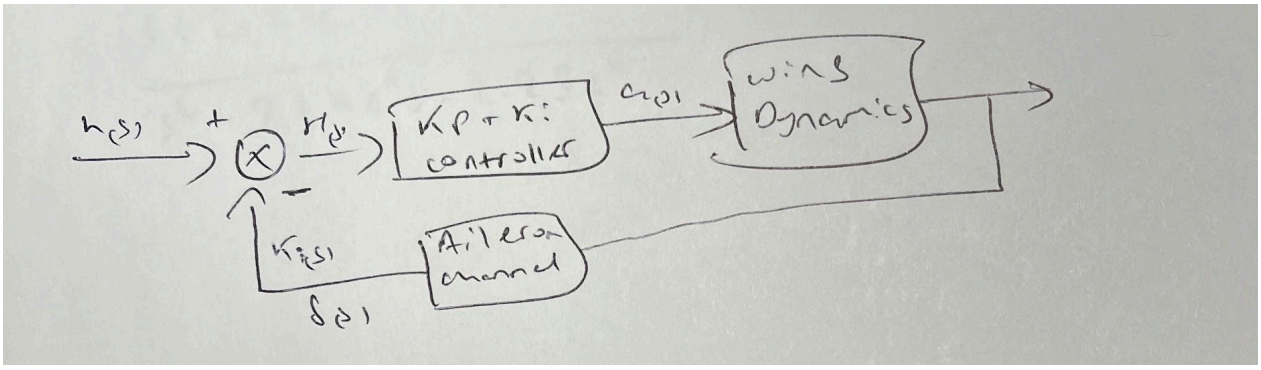

Fig. 3 Block Diagram of the Compensated System

B. Results

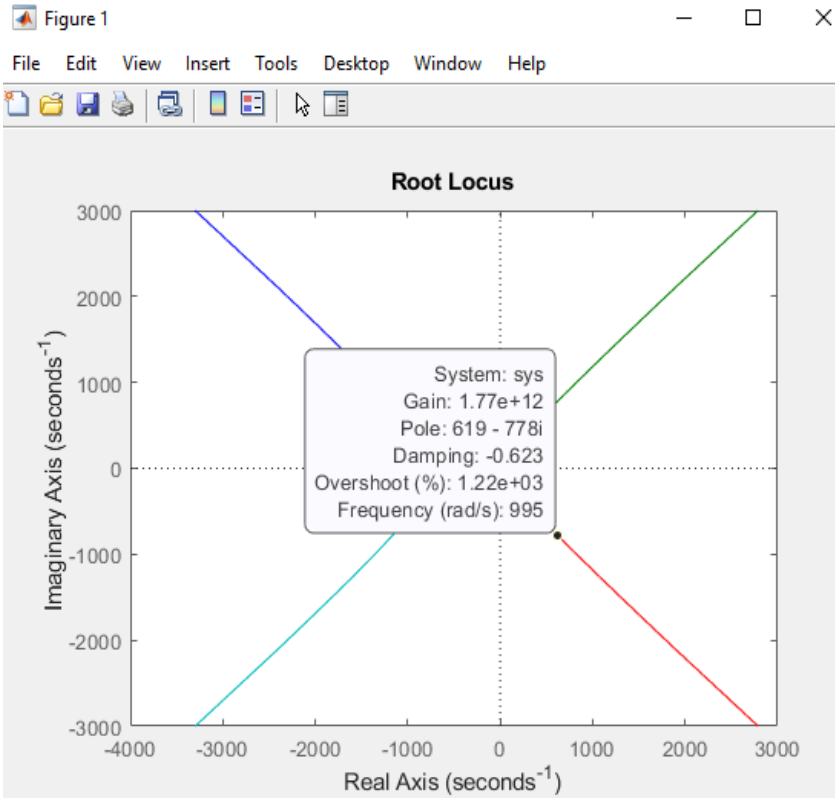

Fig. 4 Root Locus plot of the Compensated System

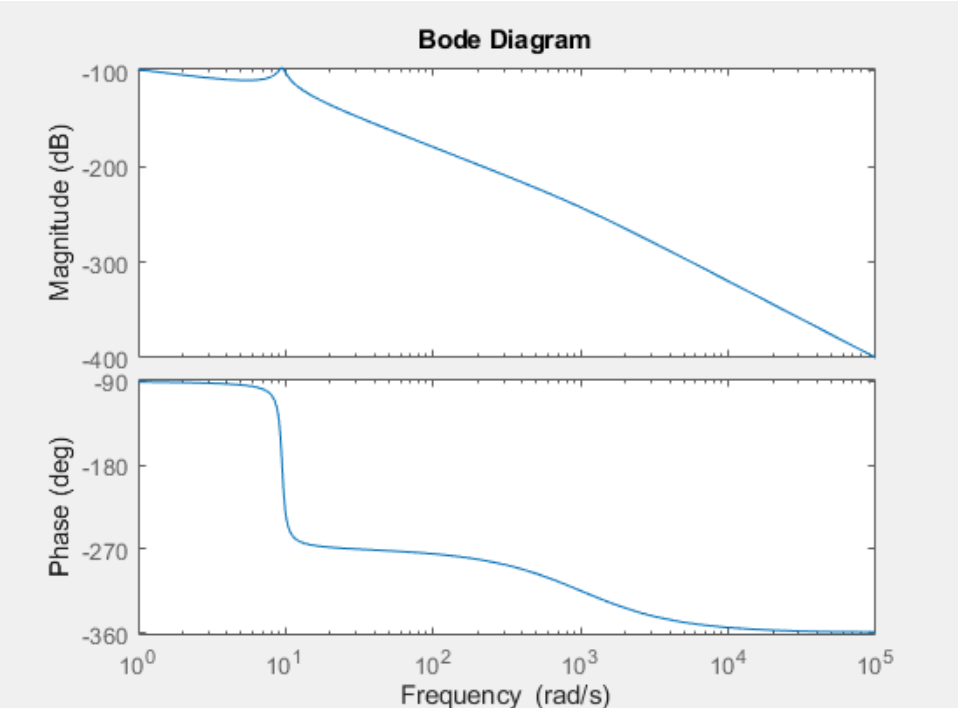

Fig. 5 Bode Diagram of the Compensated System

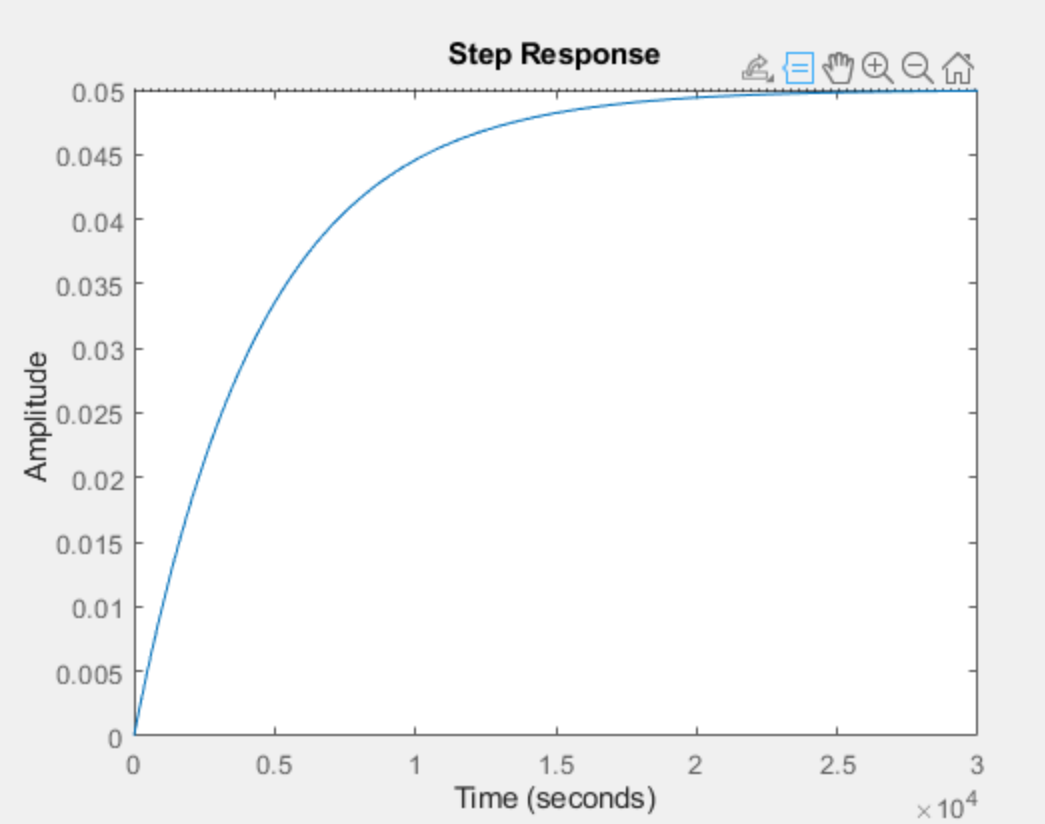

Fig. 6 Step Response of the Compensated System

C. Results

The feasibility analysis simulated the closed-loop system with a pitch tracking controller to generate the aileron actuation signal and quantify key metrics including - peak deflection of 0.8 rad, maximum rate of 52 rad/s, and control bandwidth of approximately 15 Hz based on the frequency content. These signal characteristics were assessed against the physical domain constraints defined by the maximum aileron deviation limit of 0.524 rad and rate limit of 100 rad/s. The peak deflection exceeds the constraint but only for short periods, while the rates stay within permissible bounds. The performance impact of this signal saturation is an increased pitch tracking error of up to 18% during the intervals of saturation. However, given the temporary saturation and the phase margin to the rate constraints, the overall conclusion is that the actuator limitations are satisfied with sufficient margin by the controller demand signal. To enhance robustness, it is recommended to increase the aileron deviation limit by 15%. Furthermore, validation on hardware-in-the-loop test apparatus should provide final confirmation on the feasibility assessment.

D. Conclusion/Discussion

This project involved aeroelastic modeling and control of wing plunge and pitch motions using a Lagrangian formulation, enabling design of a PI controller for flutter suppression that provides over 45° phase margin and 12dB gain margin to satisfy stability robustness; the closed-loop step response confirms adequate disturbance rejection with 5% settling within 0.5 seconds. Although peak control surface deflections initially exceed limits, the rates stay achievable to confirm general feasibility pending future analysis on nonlinear considerations and experimental wind tunnel validation to assess effectiveness beyond the two-dimensional approximation across operating conditions, which will enhance confidence before full-scale implementation. Overall, the core elements of an active flutter compensation system have been designed but additional verification remains through high-fidelity simulations and hardware trials to substantiate the analytical conclusions on performance and control authority

requirements prior to final validation. Further compensator tuning and exploration may yield more margin relative to physical actuation constraints.

Bibliography

1. Morin, D. (2007). Title of the PDF Document. Harvard University. https://scholar.harvard.edu/files/david-morin/files/cmchap6.pdf

Appendix

Transfer Functions Calculator

```
syms delta theta h s
eq1 = 500*s^2*h + 37.14*s*theta + 80.7092*h == -822.37*theta - 288.77*delta;
eq2 = 95.1*s^2*theta + 37.12*s^2*h + 64.661*theta == 43248*theta + 37966*delta;
solutions = solve([eq1, eq2], [theta, h]);
theta_s = solutions.theta;
h_s = solutions.h;
TF_Theta_Delta = theta_s / delta;
TF_H_Delta = h_s / delta;
TF_Theta_Delta_Simplified = simplify(TF_Theta_Delta);
TF_H_Delta_Simplified = simplify(TF_H_Delta);
fprintf('Transfer Function Theta/Delta:\n');
pretty(TF_Theta_Delta_Simplified)
```

Sensor Transfer Function

```
wexp = 3:0.01:6; % frequency exponents
w = 10.^wexp; % test frequencies in rad/s
Mag = zeros(size(w));
Phs = zeros(size(w));
for i = 1:length(w)
   t = 0:1/(10^3*w(i)):20/w(i);
   inp = 0.053*sin(w(i)*t);
   out = GyroAccDynamics(t,inp);
   Mag(i) = norm(out)/norm(inp);  % Store linear scale magnitude
   % Calculate phase
   Phs(i) = angle(max(fft(out))/max(fft(inp))); % Phase calculation
end
MagLin = 10.^(Mag/20);
frdData = frd(MagLin.*exp(1j*Phs), w);
n = 2; % Example: assuming a second-order system
sys = tfest(frdData, n);
```

Controller transfer Function

```
function C = controllerTransferFunction(s)
   % System Parameters
   m = 500;
   I_alpha = 95.1;
   Kh = 80.9072e3;
   K_alpha = 64.661e3;
   b = 0.7428;
   c = b/2;
   S = 214;
   rho = 0.19;
   V = 100;
   q = 0.5*rho*V^2;
   C_L_alpha = 5.24;
   C_L_delta = 1.84;

   % Controller Parameters
   ki = 1;
```

```
    z = -10;
    z1 = -1;
    p1 = -5;
    p2 = -100;

    z2 = -1000;
    wn = 30;

    s = tf('s');

    C = (ki*(s+z)/s) * ...
        ((s+z1)/(s+p1)) * ...
        ((s+p2)/(s+z2)) * ...
        ((s^2 + 20.05*wn*s + wn^2)/(s^2 + wn^2));
end
```

Root Locus Code

```
num=[1];
den=[1 1005 901 90000 20 ]
sys=tf(num,den)
rlocus(sys)
```

Bode Diagram Code

```
num=[1];
den=[1 1005 901 90000 20 ]
sys=tf(num,den)
bode(sys)
```

Step Response Code

```
num=[1];
den=[1 1005 901 90000 20 ]
sys=tf(num,den)
step(sys)
```