Warsaw University of Technology

FACULTY OF
POWER AND AERONAUTICAL ENGINEERING

Institute of Aeronautics and Applied Mechanics

Bachelor's diploma thesis

Aerospace Engineering
Vibrations and Aeroelasticity

Active Suppression of Flutter on a Plane Wing using LQR Control

Carmen Yanguas Rodríguez
K-4999

Thesis supervisor
Dr PhD, Franciszek Dul

Warsaw, 2019

1 Summary

ACTIVE SUPPRESSION OF FLUTTER ON A PLANE WING USING LQR CONTROL

This document describes the practical and theoretical implications of the research corresponding to the design of an optimal LQR control (Linear Quadratic Regulator) on the aeroelastic phenomenon of flutter in the wing of the A320 aircraft (previously modelled).

For the development of this project, Matlab software has been used, where the wing of the A320 aircraft with its corresponding elastic, aerodynamic and inertial characteristics has been modelled using the Finite Differential Method (FDM). The aeroelastic model is used in order to find the critical speed at which the flutter occurs.

Subsequently, and through this same program, a LQR control system has been developed that is capable of absorbing the vibrations originated on the wing.

Finally, the model is used to study more deeply the phenomenon of flutter, through the variation of some relevant parameters in its formation and the subsequent study of the consequences of these changes.

KEY WORDS: Aircraft; Aeroelasticity; Aerodynamics; Flutter; LQR control; Finite Differential Method.

Podsumowanie

Aktywne tłumienie trzepotania na skrzydle samolotu przy użyciu sterowania LQR

Niniejszy dokument opisuje praktyczne i teoretyczne implikacje badań odpowiadających projektowi optymalnej kontroli LQR (Linear Quadratic Regulator) na aeroelastyczne zjawisko trzepotania w skrzydle samolotu A320 (wcześniej modelowane).

Do opracowania tego projektu wykorzystano oprogramowanie Matlab, w którym skrzydło samolotu A320 z odpowiadającymi mu właściwościami sprężystymi, aerodynamicznymi i bezwładnościowymi modelowano za pomocą metody skończonej różnicy. Model aeroelastyczny jest wykorzystywany w celu znalezienia prędkości krytycznej, przy której występuje trzepotanie.

Następnie, dzięki temu samemu programowi, opracowano system kontroli LQR, który jest w stanie pochłaniać wibracje powstające na skrzydle.

Wreszcie, model jest wykorzystywany do głębszego badania zjawiska trzepotania, poprzez zmianę niektórych istotnych parametrów w jego tworzeniu i późniejsze badanie konsekwencji tych zmian.

SŁOWA KLUCZOWE: samoloty; aeroelastyczność; trzepotanie; kontrola LQR; metoda różnic skończonych.

Contents

Figures

Tables

Symbols and formulas

l	Typical length of a body
α	Typical angle between a body and a flow (angle of attack)
ρ	Density
U_∞	Speed of flow
μ	Viscosity
ω	Frequency
C	Speed of sound propagation
$C = \sqrt{\frac{\gamma p}{\rho}}$	Speed of sound propagation in a gas
$q = \frac{1}{2}\rho U^2$	Dynamic pressure
p	Static pressure
γ	Ratio of the specific heat at constant pressure to the specific heat at constant volume
S	Controllability matrix
V	Observability matrix
$C_L = \frac{lift}{qS}$	Lift coefficient
$C_D = \frac{drag}{qS}$	Drag coefficient
$C_M = \frac{pitching\ moment}{qSc}$	Pitching-moment coefficient
F_e	Elastic force
M_e	Elastic moment
ω_h	Bending eigenfrequency
ω_θ	Torsional eigenfrequency

L	Unsteady lift force
M_y	Unsteady aerodynamic moment
V_{crit}	Critical velocity of flutter
r_α	Radius of inertia of the wing
CG	Mass center of the wing
EC	Elastic center of the wing
$k_h[rigidity / unitspan]$	Elastic spring constant
k_θ	Linear spring constant
T	Kinetic energy
U	Potential energy
Q	Column vector of generalized forces
D	Rayleigh dissipated potential
M	Mass matrix
m	Mass of the wing/airfoil
I_E	Airfoil inertia
S_E	Airfoil static moment with respect to the elastic axis
K	Stiffness matrix
$\mathbb{C}$	Theodorsen function
E	Young modulus

Dimensionless numbers

$R = \frac{Ul\rho}{\mu} = \frac{Ul}{\vartheta}$	Reynolds number
$k = \frac{\omega l}{U}$	Reduced frequency or Strouhal number
$M = \frac{U}{c}$	Mach number

2 Introduction

A type of oscillation of airplane wings and control surfaces has been observed since the early days of flight. It was shown that in some situations, if a wing or wing-aileron was structurally restrained to a certain position of equilibrium, instabilities might appear under some particular conditions.

This phenomenon, that afterwards was called flutter, caused many problems in the beginnings of aviation due to its sudden nature. The vibrations developed very quickly, and destruction of the wing appeared just after a couple oscillation cycles, usually in a fraction of the second. For this reason the study of aeroelasticity and control of vibrations became necessary.

But the global vision that is required for its analysis makes it a complex subject, since in aeroelasticity different topics are united. Also, the non-stationary aerodynamic forces are very difficult to model. That is why, during the history, experimental results and trials have been necessary in real cases.

3 Objectives

During the development of this project we will try to describe the most representative phenomenon of aeroelasticity: the flutter. The main objective is to put in common the different forces that get inside the game in a vibratory problem: elastic forces, inertial forces and aerodynamic forces. To facilitate the study of this phenomenon, we will use the model of a slender wing, modelled by the finite differential method in Matlab, of the wing of the A320 aircraft, which will help us understand the numerical resolution and subsequent physical interpretation of the event.

As it will be seen, the flutter problem is a problem of dynamic instability and therefore its solution is based on the analysis of eigenvalues of a certain homogeneous problem.

The study of these eigenvalues will be used during the project to get an approach to the physical phenomenon of the flutter: its creation, characteristic speeds, parameters on which it depends on and, finally, the control of its damping.

With this last objective an active Lineal Quadratic Regulator control will be developed.

4 Overview

4.1 The concept of aeroelasticity

4.1.3 Introduction to aeroelasticity

When a fluid flow contacts an elastic body, it appears an interreaction between the flexible behaviour of the body and aerodynamics. Aeroelasticity is the branch, usually applied to aeronautics, of physics and engineering that studies these interactions between inertial forces, elastic forces, aerodynamic forces and control system dynamics.

Typically, the study of aeroelasticity is classified into two fields, depending on the nature of such interactions. They can be static (which concerns steady responses) or dynamic (which implicates variations with time and concerns vibrational responses).

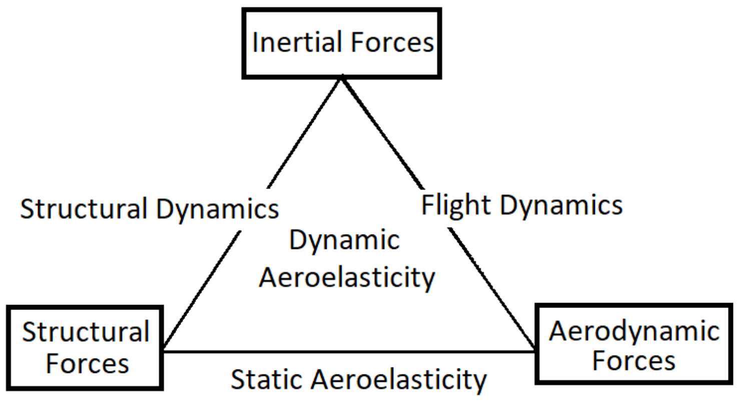


Figure 1: Aeroelasticity triangle (Author)

One of the problems studied in dynamic aeroelasticity that concerns specially the field of aeronautics is the stability of a structure exposed to wind. If the stiffness of a given elastic body is independent of the wind, the aerodynamic force can increase rapidly with the wind speed, until it reaches a critical wind speed that leads to instability of the structure. Such instability may be responsible of excessive deformations, and even the complete destruction of the structure.

Aerodynamic flutter is the dangerous phenomenon in flexible structures subjected to aerodynamic forces caused when the speed of the wind reaches a point at which structural damping is insufficient to damp out the instable motion, which increases due to the aerodynamic energy that is added to the structure. This is considered a major problem since structures such as airplanes or suspension bridges can be affected by nature small disturbances that, fortuitously, may produce oscillations of varying intensity.

The appearance of these phenomena is called a problem of *dynamic aeroelastic instability.* And the *static aeroelastic instability* would be the particular case of *dynamic aeroelastic instability* with zero frequency and neglectable inertia force.

Generally, elasticity studies the stress and deformation caused by known external forces or displacements on an elastic body (deformation is assumed small enough so that external forces remain unaffected and thus, known). Nevertheless, the situation changes when facing aeroeslasticity, because the attitude of the body relative to the flow affects strongly the aerodynamic forces. As the deformation influences the external loading, it cannot be known until the elastic problem itself is solved. That is to say, the *response* of an aeroelastic system to an externally applied load is to be found.

This response may be in the form of displacement, motion or stress state; and thus the response problems may be classified into static or dynamic problems, depending on whether the inertia forces can or cannot be neglected.

Therefore, it can be observed a close relationship between stability and response problems. Although most stability problems can be modelled using homogeneous equations, while response problems are described by nonhomogeneous systems, a response problem usually associates with a stability problem. When a finite disturbance generates a finite response, the structure will remain stable. If, on the contrary, a finite disturbance creates an indefinite response, the structure will be unstable. In unstable inertial (dynamic) structures *flutter phenomenon, buffeting* (if the external excitation is harmonic) *or gusts* will occur, while if the problem is static and unstable, the phenomena involved are called *divergence* (if the modification of the angle of attack can be stabilized or that can explode in function of velocity) or *control reversal* (if deflection of the wing increases the lift, but there is also an increase in the moment. This decreases the angle of attack, which in turn decreases the lift. The command loses effectiveness.). If the aircraft reaches the critical velocity (a specific sufficiently high speed), quick oscillations may be created on the airplane rapidly, and the wing structure will often be destroyed.

Collar triangle (*Figure 2*) of aeroelastic forces shows the relations between the different actions and the phenomena under study.

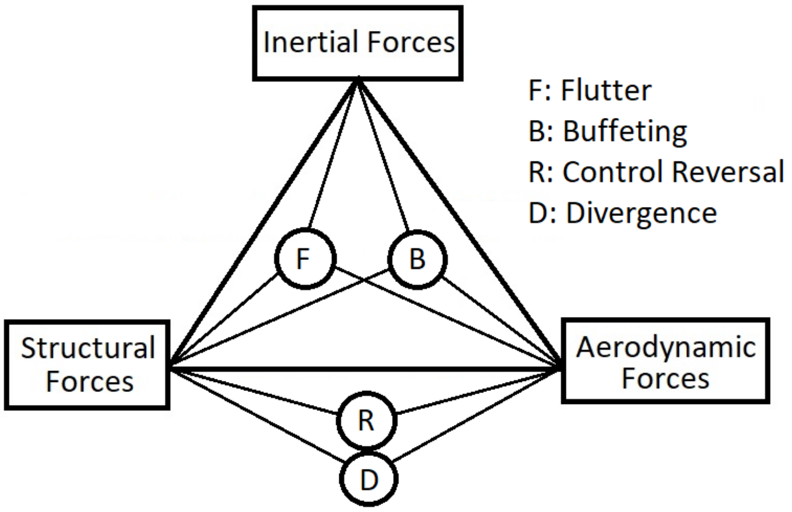


Figure 2: Collar Triangle (Author)

All aeroelastic phenomena are very famous, important and dangerous in aviation, but we will deepen on flutter motion, thus it is the motion that will be studied in the present analysis.

Flutter occurs when the profile starts to oscillate. If the aerodynamic forces draw energy from the system, the system is damped. If they deliver power to the system, then, if the system can dissipate it, it will be damped, but if it cannot dissipate the energy, the system will became instable. If during the oscillation the non-linear zone is reached, during part of the cycle stall will occur.

4.1.2 First approach to the modelling of aeroelastic phenomena

In all branches of engineering, models are made in order to be able to study and better understand the behaviour of physical systems. These models ideally represent their behaviour and allow expressing it mathematically. Aeroelasticity is not different in this sense. Since its birth, scientists have created models that allow us to study the interaction between the different forces that appear in the Collar triangle.

With that objective, the wing is the element that serves as a starting point for the aeroelastic analysis because it is the element that receives the largest component of the lift (vertical force), and also because it is the most deformable component. Besides, it has the advantage that all the forces are relatively simple to evaluate. This allows the scholars to obtain analytical results that permit to draw relevant conclusions from it.

But before starting with the study of the forces that the wing of an airplane suffers, it is important to establish the conditions (assumptions) that will be taken about the body and the environment in which it is located.

A real fluid is viscous and compressible. But if the speed of flow is clearly below Mach number, the relative motion between a body and the flow causes a variation of density small enough so that the fluid can be considered incompressible. In addition, in the case of fluids like water or air, viscosity causes an influence only in the boundary layer (next to the solid wall of the body); outside this layer the fluid can be considered nonviscous. A nonviscous and incompressible fluid is called a perfect fluid.

Many times in aeroelasticity it is possible to regard the fluid as a perfect fluid. Nevertheless, there are cases in which the viscosity of the flow affects evidently the problem under study, as it controls the boundary layer, and then the nonviscous hypothesis must be ignored.

Generally, when modelling an aeroelastic system, two main parts are taken into account when defining the equations: the structural part (elastic and inertial forces) and the flow (aerodynamic) part. In addition, at least two degrees of freedom are required in order to create a condition of instability, as it has been known that vibrations of a single degree of freedom are damped out simply by the air forces.

The *aerodynamic force* suffered by the body as a result of its interaction with the fluid depends on the relative velocity between body and fluid flow. The force consists of two components: the *pressure force* normal to the surface of the body and the *shearing force*, tangential to the surface.

The parameters on which depends the force that acts on a body in a flow are, among others, the geometry of the body and its attitude relative to the flow (these can be characterized by a typical length and angle), the density of the fluid, the viscosity of the fluid, the speed of flow, the compressibility of the fluid (which can be expressed as a function of the speed of propagation of sound in the fluid), and the non-stationary characteristics of the flow (in a periodic oscillation, it is characterized by the frequency).

Then it is shown that, for an oscillating flow of a compressible fluid, the force experienced by a body may be expressed as:

$$F = f\left(\alpha, \frac{Ul\rho}{\mu}, \frac{\omega l}{U}, \frac{U}{C}\right)\frac{1}{2}\rho U^2 l^2 \qquad [1]$$

In aeroelasticity there are two components of force and one component of moment that have a high impact on a body: *Lift* (force perpendicular to the direction of motion), *Drag* (force in the direction of motion with opposite orientation), and *Pitching moment* (moment about an axis perpendicular to the direction of motion and the lift vector, positive when it tends to raise the leading edge of the body). These three parameters lead to the three primary airplane coefficients: lift coefficient, drag coefficient and pitching-moment coefficient.

Finally, according to the theory of thin airfoils in a two-dimensional noncompressible fluid, the center of pressure of the additional lift due to change of α is located at $\frac{1}{4}$ or the chord. This point is called the *aerodynamic center.*

4.2 Modelling of flutter

We begin in this chapter the description of the flutter. The study will be supported in the Lagrange equations of movement.

We are going to develop different methods of flutter modelling, to gradually approach the model with which we have decided to carry out the approximation: the finite differencial method.

4.2.1 Semi-rigid model

In the semi-rigid approach to wing flutter and related problems a reference section of the wing is selected to represent the entire three-dimensional wing. This simplification works quite well for slender wings, that is, wings of high-aspect ratio.

In this first approach to the model of the mechanism of flutter, no particular type or shape of airfoil shall be of concern. The treatment of the phenomenon will be restricted to primary effects. The differential equations for the several degrees of freedom will be put down considering only small oscillations about the position of equilibrium.

To introduce the phenomenon we will use the problem of two d.o.f. shown in *Figure 3*, which will help us in its numerical resolution and physical interpretation. The degrees of freedom are the vertical displacement $h(t)$, positive downward, and the rotation of the airfoil around the elastic axis $\theta(t)$ with the positive sense as shown. $h(t)$ and $\theta(t)$ represent, then, the phenomena of flexion and elastic torsion in real wings.

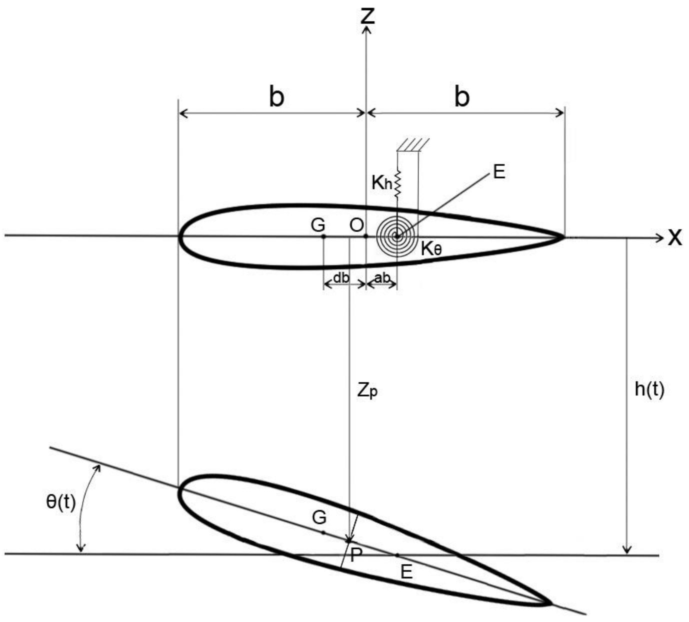


Figure 3: Semi-rigid Model (Author)

Where the parameters a and d are dimensionless parameters.

These degrees of freedom can be grouped in a dimensionless vector:

$$u(t) = \left[\frac{h}{b}, \theta \right]^T \quad [2]$$

The vector represents the amplitudes of oscillation of the system with respect to a certain position of equilibrium. Forces whose magnitudes do not depend on the degrees of freedom (for example, the weight of the profile, the aerodynamic moment due to the curvature of the airfoil or the stationary lift due to a geometric torsional angle) are not interesting because they will be placed as independent terms in the equation of motion. Thus, $h=0$ and $\theta=0$ represent a horizontal profile without aerodynamic forces acting.

It is considered that the flexor and torsion stiffness are concentrated in a point of the profile called the elastic axis E and located in the coordinate $xE = ab$. The stiffness are represented by elastic and linear springs k_h and k_θ.

The general case of a profile will be considered, whose centre of gravity is located at $xG = db$, and moment of inertia IG is located around the centre of gravity.

The equations of motion of the problem will be Lagrange's equations, based in an energetic treatment of the problem. From the equation it appears a differential equation system in time.

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{u}}\right)+\frac{\partial D}{\partial \dot{u}}+\frac{\partial U}{\partial u}=Q(t) \qquad [3]$$

The matrices that take place in the analysis of free vibrations in a mechanic system are mass matrix, stiffness matrix and damping matrix. Mass and stiffness matrices are extracted from the kinetic and potential energy associated with the deformation.

From kinetic energy it can be obtained the mass matrix, which has the form:

$$M=\begin{bmatrix} b^2 m & bS_E \\ bS_E & I_E \end{bmatrix} \qquad [4]$$

Where:

$$m=\int_{-b}^{b} dm$$
$$I_E=\int_{-b}^{b}\left(x-x_E\right)^2 dm$$
$$S_E=\int_{-b}^{b}\left(x-x_E\right) dm \qquad [5]$$

From the deformation energy of the system it is deduced the stiffness matrix.

$$K=\begin{bmatrix} k_h b^2 & 0 \\ 0 & k_\theta \end{bmatrix} \qquad [6]$$

Assuming that there are not aerodynamic forces, the equation of movement without damping takes the form [7].

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{u}}\right)+\frac{\partial D}{\partial \dot{u}}+\frac{\partial U}{\partial u}=M\ddot{u}+Ku=0 \qquad [7]$$

Now, we introduce the parameters $\omega_h = \sqrt{K_h / m}$, $\omega_\theta = \sqrt{\frac{K_\theta}{I_E}}$, $\eta = \frac{\omega_h}{\omega_\theta}$, $i_\theta = \sqrt{\frac{I_E}{mb^2}}$ and $r_\theta = \frac{S_E}{mb} = \frac{x_G - x_E}{b}$, $r = r_\theta / i_\theta$. Physically, i_θ represents the adimensional radius of the section, r_θ is the distance between the elastic axis and the centre of gravity, ω_h would be the frequency of the system if it cannot turn, and ω_θ would be the frequency in the case that the airfoil has a fixed joint in $xE = ab$.

From the equations of movement, and getting the harmonic solutions, we obtain the natural frequencies, which are the roots of the equation.

$$\lambda_1^2 = \left(\frac{\omega_1}{\omega_\theta}\right)^2 = \frac{1+\eta^2+\sqrt{\eta^4 - 2\eta^2 + 4\eta^2 r^2 + 1}}{2(1-r^2)}$$
$$\lambda_2^2 = \left(\frac{\omega_2}{\omega_\theta}\right)^2 = \frac{1+\eta^2-\sqrt{\eta^4 - 2\eta^2 + 4\eta^2 r^2 + 1}}{2(1-r^2)}$$

[8]

These frequencies are the modes of vibration associated with the problem. Both of them are composed by displacement and torsion, that is to say, there is a coupling bending-torsion. The parameter r leads the degree of coupling, while η represents the relation between the frequencies of bending and torsion that are not coupled.

It is possible to calculate the components of the vector of generalized forces $Q(t)$ from the virtual work of the resultant (lift) and the moment. The two generalized forces associated to our problem are $Q_{h/b} = -Lb$ and $Q_\theta = M_a$. And we can obtain an analytic solution of the lift and moment on one point.

$$L = 2\pi\rho_\infty U_\infty b\left[\dot{h} + U_\infty\theta + b\left(\frac{1}{2} - a\right)\dot{\theta}\right]\cdot\mathbb{C} + \pi\rho_\infty b^2\left(\ddot{h} + U_\infty\dot{\theta} - ab\ddot{\theta}\right)$$
$$M_a = 2\pi\rho_\infty U_\infty b^2\left(\frac{1}{2} + a\right)\left[\dot{h} + U_\infty\theta + b\left(\frac{1}{2} - a\right)\dot{\theta}\right]\cdot\mathbb{C} + \pi\rho_\infty b^2\left[ab\ddot{h} - U_\infty b\left(\frac{1}{2} - a\right)\dot{\theta} - b^2\left(\frac{1}{8} + a^2\right)\ddot{\alpha}\right]$$

[9]

In order to investigate flutter, it is necessary to analyse the stability of the solutions to the equations of flutter's model.

Assuming the harmonic form of bending and torsional vibrations:

$$h(t) = \bar{h}_0 e^{i\omega t} \qquad \omega_I = \mathrm{Im}(\omega) \qquad h(t) = \bar{h}_0 e^{-\omega_I t}\sin(\omega_R t) \tag{10}$$

$$\theta(t) = \bar{\theta}_0 e^{i\omega t} \qquad \omega_I = \mathrm{Im}(\omega) \qquad \theta(t) = \bar{\theta}_0 e^{-\omega_I t}\sin(\omega_R t + \Phi) \tag{11}$$

These mathematical expressions can be interpreted and used, then, for the understanding of the physical phenomenon. This is achieved with the analysis of the imaginary parts of the roots:

If $\omega_I > 0$ the motion is stable.

If $\omega_I = 0$ the motion is neutrally stable.

If $\omega_I < 0$ the motion is unstable. That is to say, flutter occurs.

Since ω_I depends on the velocity of the flow ($\omega_I = \omega_I(U_\infty)$), the condition $\omega_I = 0$ can be used in order to determine the critical velocity of flutter: V_{crit}.

The matrix form of the flutter equations is:

$$A(\omega,U)\bar{q} = 0$$

$$\bar{q} = [\frac{\bar{h}_0}{b}, \bar{\theta}_0] \tag{12}$$

And the condition that must be met for the flutter to occur is [13].

$$\det A(\omega,U) = 0 \tag{13}$$

From the previous expression it is derived one of the *Pines rules.*

If the mass center G is placed in front of the elastic center E, flutter will never occur.

If the ratio of bending and torsional frequencies of vibrations is small, $\frac{\omega_h}{\omega_\theta} \ll 1$, the critical velocity can be approximated to:

$$
\begin{aligned}
& M\ddot{q} + Kq = f(q) \\
& f(q) = M_A\ddot{q} + C_A\dot{q} + K_A q \\
& (M - M_A)\ddot{q} - C_A\dot{q} + (K - K_A)q = 0 \\
& q(t) = \bar{q}e^{i\omega t} \\
& \det\left(-\omega^2(M - M_A) - i\omega C_A + (K - K_A)\right) = 0
\end{aligned}
$$ [14]

From the previous expression we can infer that the critical velocity of flutter decreases when m decreases, r_θ decreases, ρ increases, E moves backward or G moves backward.

From here we can go to deepen in the different treatments that can be given to the flutter according to the particularities used in its modelling.

4.2.2 Beam model of bending and torsional vibrations

The beam model approaches the problem of modelling a wing assuming that due to its characteristics it can be treated as a beam.

The beam model includes the front and rear spar, as well as the upper and lower skin between the two spars. The engines are taken into account too. Ribs as well as all other components (e.g. leading/trailing edges) are neglected. For the beam model, the theory of bending according to Euler and Bernoulli and the theory of torsion according to St. Vénant for thin-walled closed section beams is used for the calculation of the displacements and stresses.

The Bernoulli assumptions for beams are as follows:

-It is a slender beam (the length is much bigger than all other dimensions).

-The beam cross section, which was rectangular to the beam axis before bending, remains rectangular to the beam axis after bending.

-The cross sections remains planar after bending.

The beam model of flutter enables one to determine the critical velocity of flutter for nonuniform wings with nonuniform aerodynamic loads. Usually, the accuracy of this model is higher than the semi-rigid model, but it is more complex computationally.

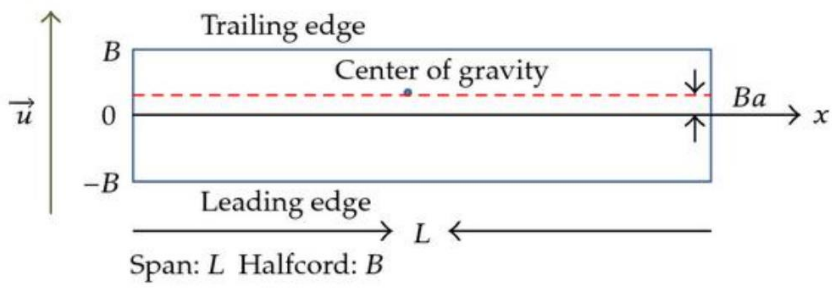


Figure 4: Wing Structure beam model (Shubov, Holt, & Wineberg, 2010)

The equations of motion of the wing are:

$$m\ddot{h}(y,t) - S_y\ddot{\theta}(y,t) - \frac{d^2}{dy^2}\left(EI(y)\frac{d^2h(y,t)}{dy^2}\right) = L(y,t)$$
$$S_y\ddot{h}(y,t) - I_y\ddot{\theta}(y,t) - \frac{d}{dy}\left(GJ(y)\frac{d\theta(y,t)}{dy}\right) = M_y(y,t) \qquad [15]$$

The aerodynamic loads derived from the strip model of the wing:

Lift force:

$$L(y,t) = \omega^2 L_h h + hL'_h\,\dot{h} + \omega^2 L_\theta \theta + \omega L'_\theta \qquad [16]$$

Aerodynamic moment:

$$M_y(y,t) = \omega^2 M_h h + \omega M'_h\,\dot{h} + \omega^2 M_\theta \theta + \omega M'_\theta\,\dot{\theta} \qquad [17]$$

The approximate displacement and twist of the wing are assumed in the form:

$$h(y,t) = \sum_{i=1}^{i=r} q_i(t) f_{\omega i}(y)$$
$$\theta(y,t) = \sum_{i=r+1}^{i=n} q_i(t) f_{ei}(y) \qquad [18]$$

Now the application of the Galerkin's method (*See APPENDICES: 9.1 Galerkin method*) to the equations of wing's motion leads to the equations of motion.

$$\int_0^l \left[m\ddot{h} - S_y\ddot{\theta} - \frac{\partial^2}{\partial y^2}\left(EI\frac{\partial^2 h}{\partial y^2}\right) - L\left(h,\dot{h},\theta,\dot{\theta}\right)\right] f_{h_i}(y)dy = 0 \qquad i = 1,\ldots,r \qquad [19]$$

$$\int_0^l \left[S_y\ddot{h} - I_y\ddot{\theta} - \frac{\partial}{\partial y}\left(GJ\frac{\partial \theta}{\partial y}\right) - M_y\left(h,\dot{h},\theta,\dot{\theta}\right)\right] f_{\theta_{ji}}(y)dy = 0 \qquad i = r+1,\ldots,n \qquad [20]$$

That can be expressed in the matrix form.

$$M\ddot{q} + Kq = f(q) \tag{21}$$

Where the aerodynamic loads have the matrix form [22].

$$f(q) = M_A\ddot{q} + C_A\dot{q} + K_A q \tag{22}$$

Equations of the wing flutter are then:

$$(M - M_A)\ddot{q} - C_A\dot{q} + (K - K_A)q = 0 \tag{23}$$

We assume again the wing's motion in the harmonic form [24].

$$q(t) = \bar{q}e^{i\omega t} \tag{24}$$

And finally we get the flutter determinant, that is:

$$\det\left(-\omega^2(M - M_A) - i\omega C_A + (K - K_A)\right) = 0 \tag{25}$$

4.2.3 Strip quasi-steady aerodynamic model

Usually, it is very complex to model a real structure (especially if we are talking about aircrafts). An aircraft is a complex structure in terms of geometry, mass distribution and stiffness distribution. In addition, in real designs small reparations are performed very often when hysteresis and backlashes occur, and therefore it is frequent to find elements riveted or glued.

Since the equations describing the motion of a vibrating wing are difficult to solve analytically, the dynamic properties of these structures are often described in terms of what are called modal parameters: natural frequency, damping factor, modal mass, and mode shape. This technique is called modal analysis, and can be implemented mathematically by attempting to uncouple the structural equations of motion so that the resulting equations can then be solved individually. From a model of an aircraft treated as a continuous system, these equations cannot be solved exactly. Consequently, a single equation of the theory of elasticity is impossible to derive, and numerical approximations are generally implemented.

That is why discretization is used. Discretization permits the model of a structure to be derived, by the use of the division of the structure onto elements, which can be easily modelled.

The strip quasi-steady aerodynamic model of a wing is a discrete model. In the strip quasi-steady aerodynamic model the wing is sub-divided into a set of small spanwise strips. The lift and pitching moment on each strip is modelled following 2D sectional lift and moment theories, in this case the quasi-steady one.

The basic principle of strip theory is that the portion of the airfoil, wing or craft submerged into the flow is divided into finite number of strips and then 2D hydrodynamic coefficients for added mass can be computed for each strip and after integrated over the length of the body to yield the 3D coefficients. The engines are not taken into account.

The equation of motion of the wing modelled as typical section are:

$$\begin{aligned} m\ddot{h} + S_\theta \ddot{\theta} + m\omega_h^2 h &= -\frac{1}{2}\rho U_\infty^2 S \frac{\partial C_z}{\partial \theta}\left(\theta + \frac{\dot{h}}{U_\infty}\right) \\ S_\theta \ddot{h} + I_\theta \ddot{\theta} + I_\theta \omega_\theta^2 \theta &= \frac{1}{2}\rho U_\infty^2 S\left(\frac{1}{2}+a\right) b \frac{\partial C_z}{\partial \theta}\left(\theta + \frac{\dot{h}}{U_\infty}\right) \end{aligned} \qquad [26]$$

And the flutter determinant for harmonic vibrations is:

$$\det\left[-\omega^2 A + i\omega C(U_\infty) + B(U_\infty)\right] = 0 \qquad [27]$$

$$C(U_\infty) = \begin{bmatrix} \frac{1}{2}\rho U_\infty^2 S \frac{\partial C_z}{\partial \theta} & 0 \\ -\frac{1}{2}\rho U_\infty^2 S\left(\frac{1}{2}+a\right) b \frac{\partial C_z}{\partial \theta} & 0 \end{bmatrix} \qquad [28]$$

In the quasi-steady model, the total force on the wing is summation of the forces on a set of chordwise strips, or blade elements.

The total number of strips used in the work is chosen. For each chord strip, the translational velocity, rotational velocity, and angle of attack are obtained from the reconstructed wing kinematics. The total force on each strip is composed of three components: the translational force, the rotational force, and the added-mass effect (or the acceleration effect).

There is no aerodynamic interaction between strips, so there is limited or no aerodynamic influence between elements.

4.2.4 FDM

The discretization of a structure can be implemented in different levels, depending on the grade of simplification used. As more detailed is the discretization, a larger number of degrees of freedom will be taken into account. The dynamic characteristics of discretized models are determined by approximate numerical methods.

Although the most commonly used method is the *Finite Element Method*, we will focus on a slightly different approximate numerical method: the *Finite Differential Method*. In engineering practice the finite difference methods are of very limited applicability, as only simplest geometries can be treated by this kind of discretisation.

FDM consists of breaking down the object into a large number of smaller, more manageable (finite) elements. Then the complex equations that describe the behaviour of a structure can often be reduced to a set of linear matrix equations which can be solved using standard matrix algebra techniques.

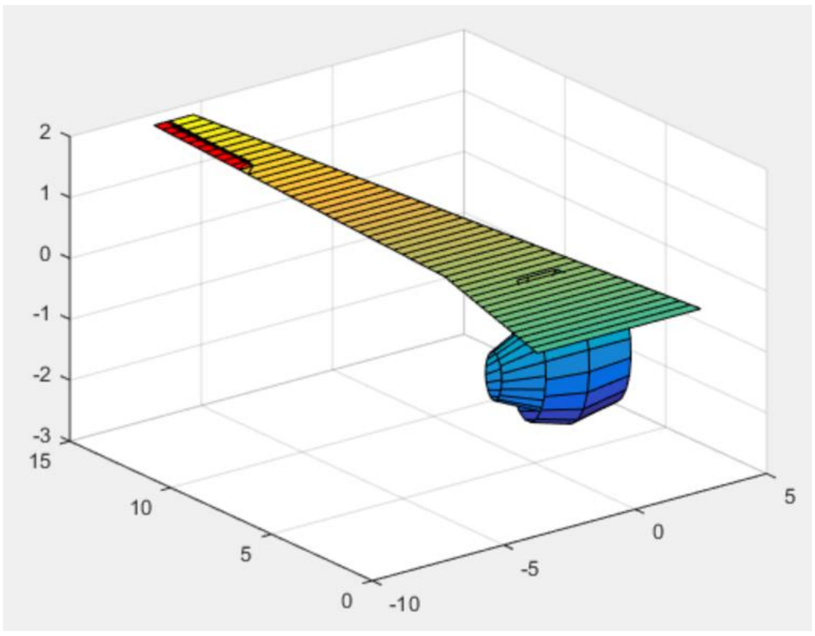


Figure 5: Wing devided with FDM (from program)

Let's suppose that it is necessary to apply Finite Differential Method to estimate the first derivative of function $u(x)$ at some point x_j, where the value of the function at the neighbouring points is known.

$$x_j = u(x_j), u_{j+1}, u_{j-1}$$

$$x_j = jh$$ [29]

According to the direct definition of the first derivative:

$$u_j' \equiv \left.\frac{du}{dx}\right|_{xj} = \lim_{h\to 0}\frac{u(x_j+h)-u(x_j)}{h}$$ [30]

The first derivative of the points can be approximated to the following algebraic formulas.

$u_j' \approx \frac{u_{j+1}-u_j}{h}$ Forward finite difference.

$u_j' \approx \frac{u_j-u_{j-1}}{h}$ Backward finite difference.

$u_j' \approx \frac{u_{j+1}-u_{j-1}}{2h}$ Central finite difference (the average of the first two).

4.3 Critical velocity

The critical velocity in an aeroelastic phenomenon refers to the least velocity of the flow at which the aeroelastic phenomenon may occur.

In this project, we will call critical velocity the velocity at which flutter phenomenon occurs.

In general, it is desired that the critical velocity is as large as possible, because the higher the critical velocity the more range of speeds are possible for the airplane without compromising its integrity and that of its passengers.

The critical velocity of flutter can be determined from the flutter determinant.

$$\det A(\omega,U)=0$$

Classically, such problems are solved with the algorithm of the V-g method. The algorithm is solves using the following steps.

1. Assume the value of the artificial damping coefficient g_s.
2. Define the new variable $z=\left(\frac{\omega_\theta}{\omega}\right)^2(1+ig)$.
3. Introduce the new variable to the flutter equation $z=\left(\frac{\omega_\theta}{\omega}\right)^2(1+ig_s)$.
4. Assume the value of the reduced frequency k.

5. Determine the aerodynamic loads $L(k)$ and $M(k)$.
6. Determine the damping g and the value of $\frac{U}{\omega b} = \frac{1}{k}$.
7. Repeat these calculations for various k and plot the function $g(\frac{U}{\omega b})$.
8. Determine the critical velocity from the intersection of the plot with line $g = g_s$.

4.5 LQR control

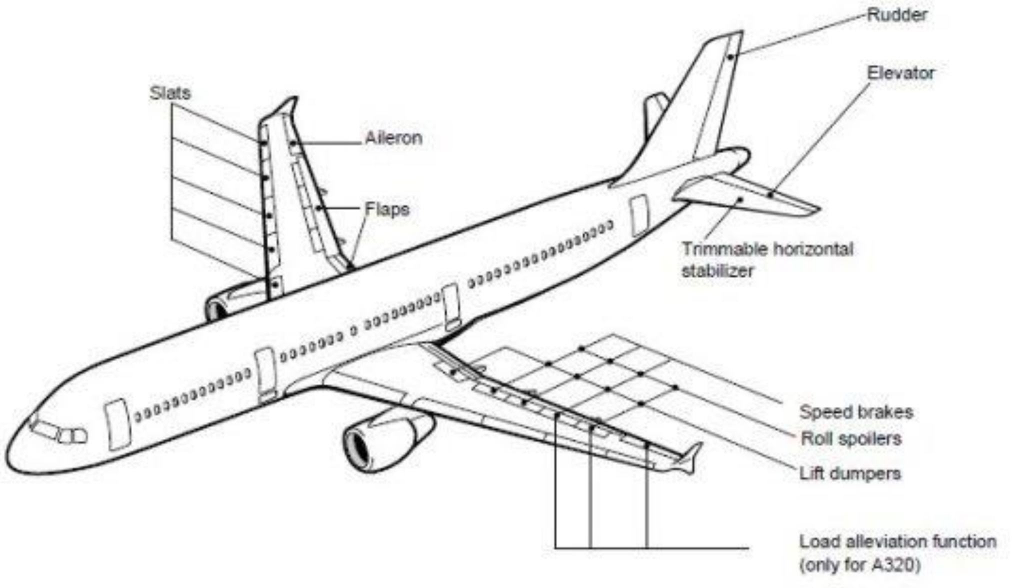

Rudder
Elevator
Slats
Aileron
Flaps
Trimmable horizontal stabilizer
Speed brakes
Roll spoilers
Lift dumpers
Load alleviation function (only for A320)

Figure 6: A320 Flight Control Surfaces (Modern Airliners)

The Linear Quadratic Regulator (LQR) is a well-known optimal control method that provides optimally controlled feedback gains. It is a modern multivariable control action that is characterized by its robustness both in discrete and continuous time to enable closed-loop stability and high performance designed systems.

It is called optimum control strategy because the proportional controller uses a mathematical algorithm that minimizes the cost function. The minimum cost is pursued when the dynamic system is operating.

For the derivation of the linear quadratic regulator it is necessary to consider a linear system state-space representation. The state equations of the system (of order n) that is going to be controlled are of the form [31].

$$\left[\dot{x}\right]_{nx1} = \left[A\right]_{nxn}\left[x\right]_{nx1} + \left[B\right]_{nxr}\left[u\right]_{rx1}$$

$$\left[\dot{y}\right]_{nx1} = \left[C\right]_{nxn}\left[x\right]_{nx1} + \left[D\right]_{nxp}\left[u\right]_{px1}$$

[31]

Where the feedback gain is a matrix [K] and the feedback control action is essentially a proportional control action, which takes the form:

$$u = -\left[K(t)\right]\left[x(t)\right]$$

$$\left[K(t)\right] = \left[R^{-1}\right]\left[B\right]^{T}\left[P(t)\right]$$

[32]

But before deepen in the definition and performance of the LQR control, three concepts must be introduced: *stability*, *controllability* and *observability.* The concepts of controllability and observability first presented by Kalman play an important role in the theoretical and practical aspects of modern control. The conditions on controllability and observability govern the existence of a solution to an optimal control problem.

The *stability* is the warranty that the system is stable (the output won't be infinite) in close-loop. This condition occurs when the uncontrollable states of the system are stable.

In a *controllable system* it is possible to build r control signals without any restriction that converge an initial state to any other finite state in a finite interval of time. Any control action that is applied to the system is capable of bringing the process states to a reference value in a determined time. As the states are affected by the inputs, the control is executed employing the system inputs.

In a controllable system every state must be affected by the inputs. It must exist an input or control function that transforms the states from a value to a different one in a finite time. If all the states are controllable, it is said that the system is completely controllable.

In order to ascertain the completely controllability of a system, it is necessary and sufficient that the expression [33] is fulfilled.

$$\left[S\right]_{nxrn} = \left[B \quad AB \quad A^{2}B \quad \ldots \quad A^{n-1}B\right]$$

[33]

$$range\left[S\right]_{nxnr} = n$$

Where S is called the controllability matrix.

Essentially, a system is fully observable if each state variable of the system affects an output at each instant of time, that is, there is a coupling between states and outputs. If any of the states cannot be observed from the measurements of the outputs, it is said that the state is not observable, and the system is not completely observable, or simply not observable.

Observability can be checked using the expression [34], that must be fulfilled.

$$\left[V\right]_{nxnp} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

$$Range\left[V\right]_{nxnp} = n \qquad [34]$$

Being V the observability matrix.

Now we can continue with the derivation of the LQR control. LQR control is calculated by minimizing the cost function J [35]. In this case, cost function J is expressed in the situation of a permanent (controllable and observable) regime.

$$J = \int_0^{\infty} ([x]^T [Q][x] + [u]^T [R][u])dt \qquad [35]$$

What is called algebraic Ricatti equation. From equation [32], $\left[P(t)\right]$is the Ricatti equation matrix, and it is found following equation [36].

$$\left[\dot{P}\right] = [A]^T [P] + [P][A] + [Q] - [P][B][R]^{-1} [B]^T [P] \qquad [36]$$

Ricatti equation can be solved only if the system is controllable and observable, therefore before applying the LQR control law, it must be verified that the process to control complies with these two conditions and thus the existence of the matrix $\left[K(t)\right]$for optimal control.

Once the matrices $[Q]$ (weighted error matrix) and $[R]$(weighted control matrix) have been defined, the cost function is determined and then the LQR technique minimizes J from the control variables.

4.5 Matlab

MATLAB (abbreviation of MATrix LABoratory) is a numerical computation system that offers an integrated development environment (IDE) with its own programming language (M language). It is available for Unix, Windows, Mac OS X and GNU/Linux platforms. It was developed by MathWorks.

Among its basic features are: the manipulation of matrices, the representation of data and functions, the implementation of algorithms, the creation of user interfaces (GUI) and communication with programs in other languages and with other hardware devices.

It is a software widely used in universities and research and development centers. In recent years the number of features has increased, such as programming directly digital signal processors or creating VHDL code.

5 Implementation

5.1 Matlab code

In order to model the control system and aeroelastic phenomena previously defined, it has been used the software Matlab. The program used for the model is called *wing_flutter*, and it is divided in five main parts: aeroelasticity (in which we can find scripts that model aerodynamics and elasticity), control, general information, mathematical developments and plotting.

Several scripts are dedicated to each of these parts of the program. None of them will be deeply explained here, but they will be named, and the general structure of the program will be shown. (For more detailed information about the program, go to APPENDICES).

The "skeleton" of the main program used is shown schematically in the image below. All the functions that make up the main program are shaded, divided according to their function in it.

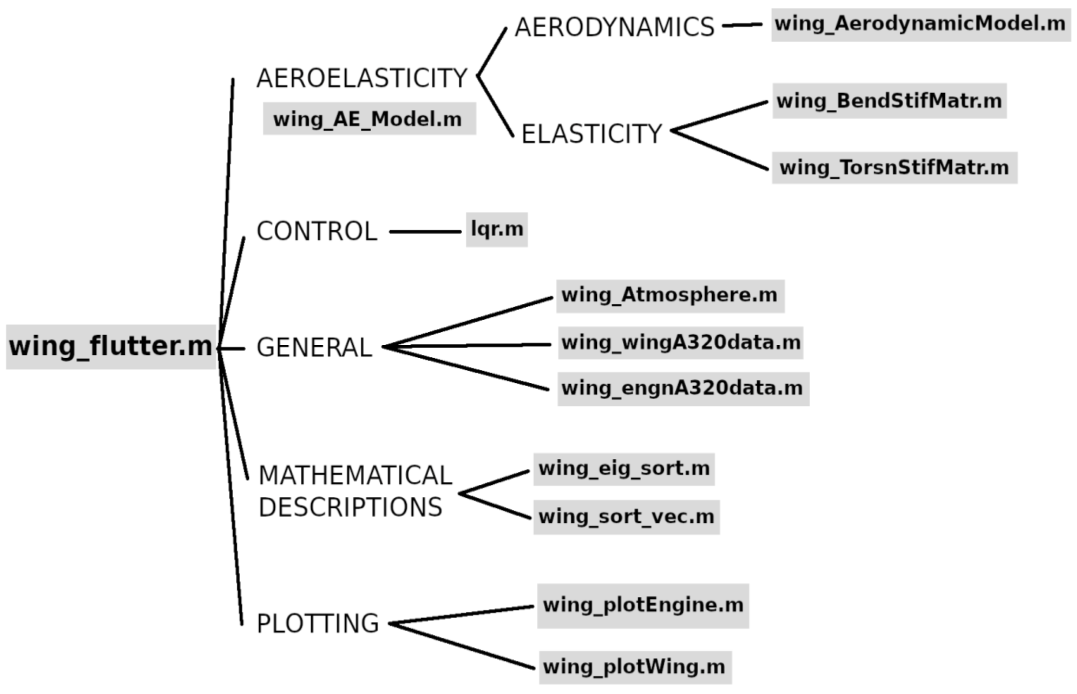


Figure 7: Scheme of wing_flutter.m (Author)

It is important to state that from now on the variable of the vertical displacement will be called w. (During the theoretical part it has been replaced by h so that it does not create confusion with the frequencies ω). Thus, the two degrees of freedom will be $w(y,t)$ and $\theta(y,t)$.

5.2 Airbus A320 Specifications

As has already been mentioned, for the modelling of the aeroelastic phenomena the specifications of a base plane wing will be used. For this purpose, it has been chosen to use the wing of the aircraft Airbus A320.

The basic characteristics of the base case are presented below, since some of the parameters established here will be used soon to discuss the consequences of the variation of certain parameters in the appearance of the flutter.

Wing span	35,8 m
Wing leading edge sweep angle	21,7º
Wing area	124 m^2
Wing aspect ratio	10,3
Payload	16,6 tons
Length	37,57 m
Height	11,76 m
Range	6100 Km
Engine position (**See** Figure 11)	x0_e = 0,0m; y0_e = 3,35m; z0_e = -1,0m;
Dihedral Angle	6,88º
Torsional Stiffness	9.5e+06 $\frac{Nm^2}{rad}$;
Bending Stiffness	7.0e+06 Nm^2;

Table 1: Specifications of Wing A320 (Author)

6 Results

Modelling a system as a set of concentrated masses joined by elastic elements produces a characteristic matrix. This type of matrix equation is well known in algebra, and contains a lot of information about the possible natural ways of vibrating a multi-mass system. A system with n masses (n number of wing sections in the FDM) has n natural vibration forms and frequencies. The vibration frequencies are the eigenvalues of the matrix and can be obtained by diagonalizing said matrix. For each form or mode of vibrating, the amplitudes have a characteristic distribution, which is given by the eigenvectors of the matrix. That is to say, each own frequency has an associated vector that informs us about the basic amplitudes of each mass in the mode or frequency.

In this section we run the simulation and analyse the results obtained. The results collected are in the form of vibration modes of the system and eigenvalues of the matrix $[A]_{nxn}$. First 50 eigenvalues correspond to vertical displacement of the elastic centre $w(y,t)$, while from 51 to 100 correspond to twist $\theta(y,t)$.

It is possible to find the critical speed of flutter through the real part of the eigenvalues, because at the moment when the value equals 0, the system becomes unstable and therefore the flutter is happening. In the case of the controlled system, the real part of the eigenvalues will never exceed the value -1, which means that the system will remain stable.

The vibration modes characterize the pattern or shape in which both degrees of freedom, the displacement and the twist, are affected by the perturbations of the system. They show the deformed wing.

In order to carry out a complete study of the flutter phenomenon, simulations of the base wing (wing of the A320 aircraft) will be carried out with and without activated control, in order to establish the critical speed in this base case.

Subsequently, some parameters will be varied, such as the position of the motors in the wing or its length, and the simulations will be repeated. In this way we can establish a relationship between the variation of the parameters and the critical speed at which the flutter appears.

6.1 Variation of control parameter

We start with the base case, using the wing model of the A320 with the control deactivated and activated.

Base Case

Engine position (See Figure 11**)**	x0_e = 0,0m;
	y0_e = 3,35m;
	z0_e = -1,0m;

Dihedral Angle	6,88º
Torsional Stiffness	9.5e+06 $\frac{Nm^2}{rad}$;
Bending Stiffness	7.0e+06 Nm^2;
Wing leading edge sweep angle	21,7º

Table 2: Specifications of Base Case (Author)

6.1.1 Uncontrolled wing

With the control disabled, the simulation is run. The results obtained are shown.

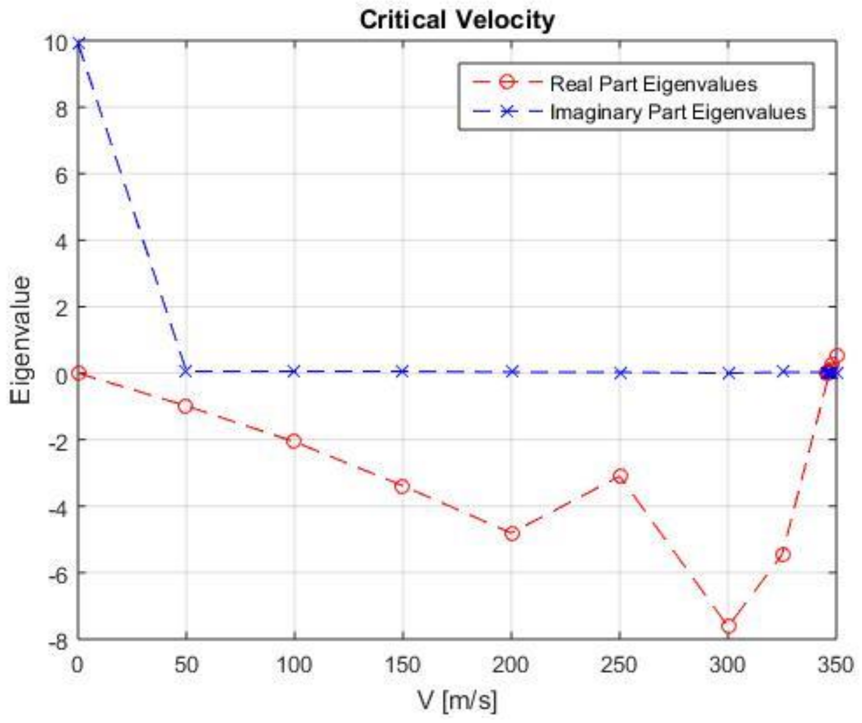


Figure 8: Eigenvalues Uncontrolled Wing (from program)

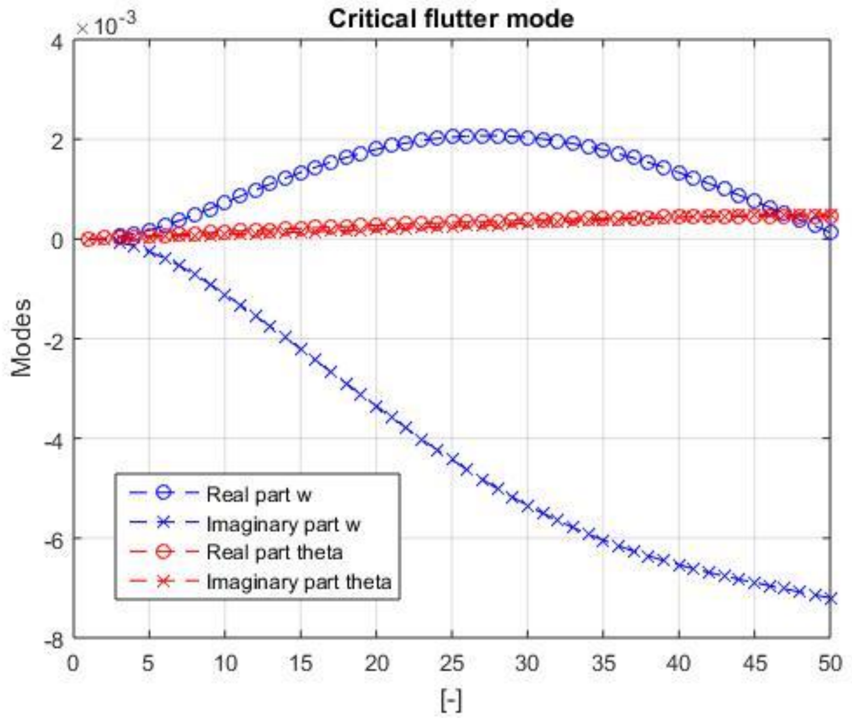


Figure 9: Modes Uncontrolled Wing (from program)

The flutter modes show how the vertical displacement generated by the flutter are more remarkable than the twist.

Base Case	*[m/s]*	*[km/h]*	*Mach*
Critical Flutter Velocity	346,789	1248,4	1,126

Table 3: Base Case. Critical velocity (Author)

6.1.2 Controlled wing

Now the control is activated. In the mathematical simulation of the control, it is modelled simply using an LQR control, but it is important to note that in the case of an actual airplane, the same vibration control could be done through different control surfaces. In our concrete case we will suppose that the control is done by aileron. Therefore, when modelling the wing, the model will consist of the wing plus the aileron.

It must be stated that the system was satisfactorily damped without need of control by structural damping, thus in the mathematical model the structural (bending and torsional) damping was set to 0. For later work, it is interesting to get some knowledge concerning the condition of absence of the internal friction, as this case constitutes a sort of lower limit, which it is not always desirable to exceed.

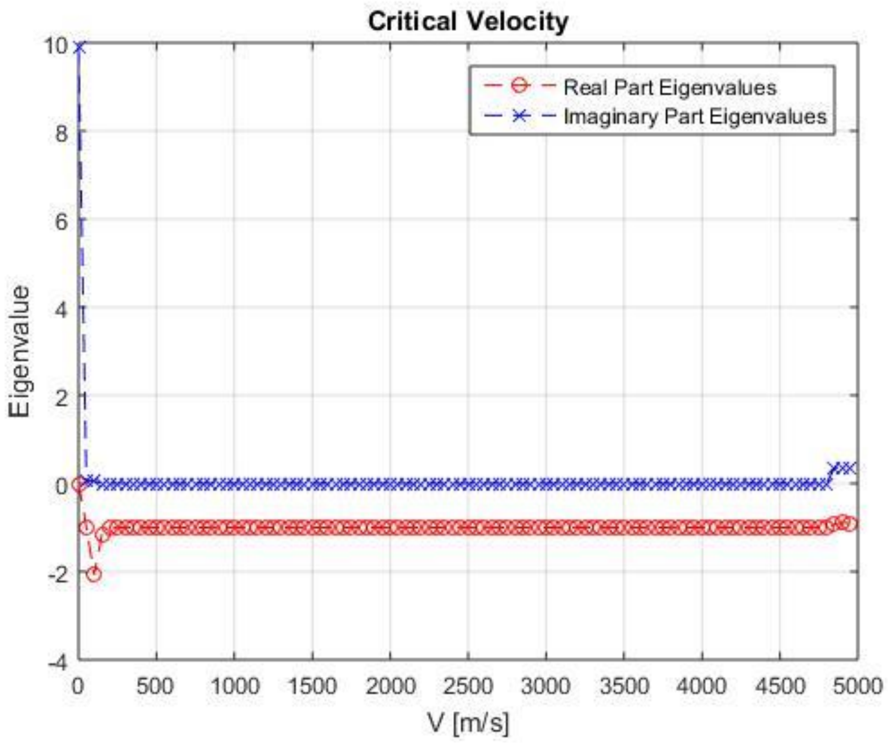


Figure 10: Eigenvalues Controlled Wing (from program)

It can be observed that controlled wing presents a maximum value of the real part of the eigenvalues smaller than zero, whereas the maximum real part of the eigenvalues of uncontrolled wing crosses zero at flutter speed.

6.2 Variation of other parameters

6.2.1 Variation of position of engines

The model used for the simulation of the flutter takes into account the position of the motors in the wing. In the model, the position of the motors is given by three coordinates: x0_e, y0_e, z0_e. These coordinates indicate the position of the engine with respect to the point where the wing joins the body of the airplane by the part of the nose. As indicated in the figure, the x-axis is parallel to the longitudinal axis of the airplane, the y-axis follows the direction of the span and the z-axis indicates the height.

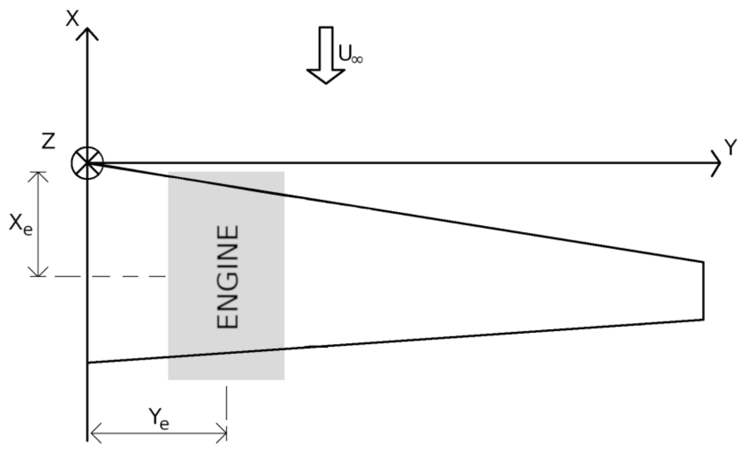


Figure 11: Position of Engines (Author)

The engines of the A320 aircraft are located following the following coordinates.

<u>Base case</u>

x0_e = 0,0m;

y0_e = 3,35m;

z0_e = -1,0m;

The first variation of the base case is carried out in the y axis in a positive direction and keeping the rest of the coordinates invariable.

Case 1

x0_e = 0,0m;

y0_e = 14,0m;

z0_e = -1,0m;

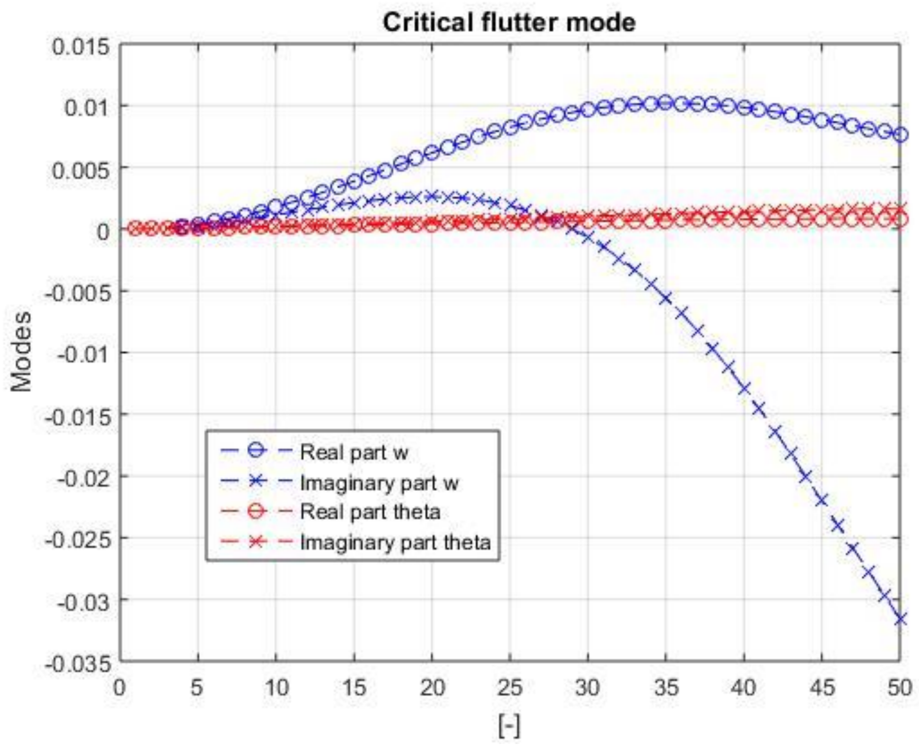


Figure 12: Variation of engine position. Modes Case 1 (from program)

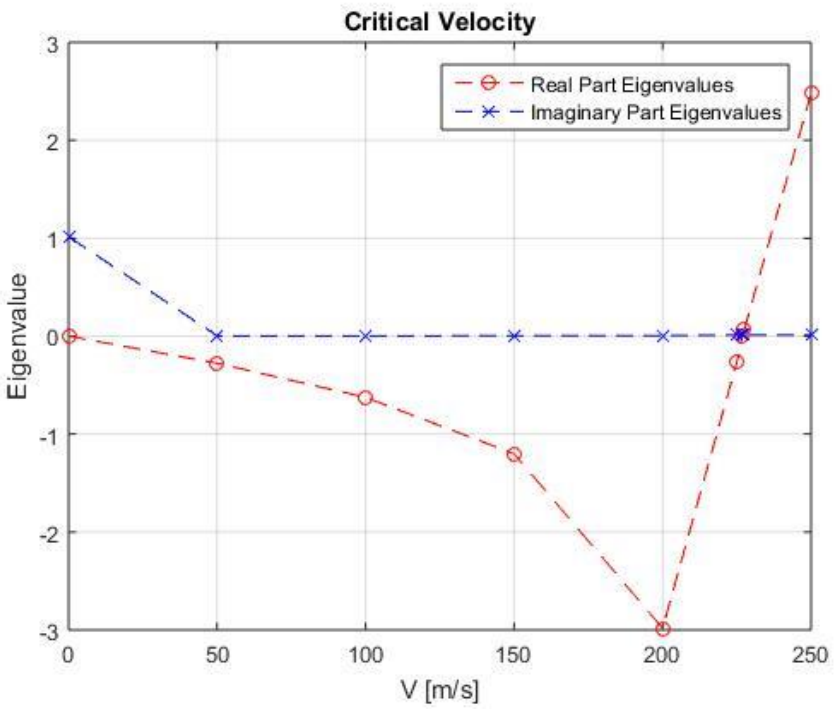


Figure 13: Variation of engine position. Eigenvalues Case 1 (from program)

U_crit=227,328 m/s.

The critical speed is lower in this case. That is, the flutter appears faster. Also, flutter modes show that vertical displacement of the wing behaves in a sharper way, thus the ranges of the upper and lower limit of the eigenvectors are higher.

For the second case, the variation is made again on the y-axis, but this case in the negative sense.

<u>Case 2</u>

x0_e = 0,0m;

y0_e = 0,5m;

z0_e = -1,0m;

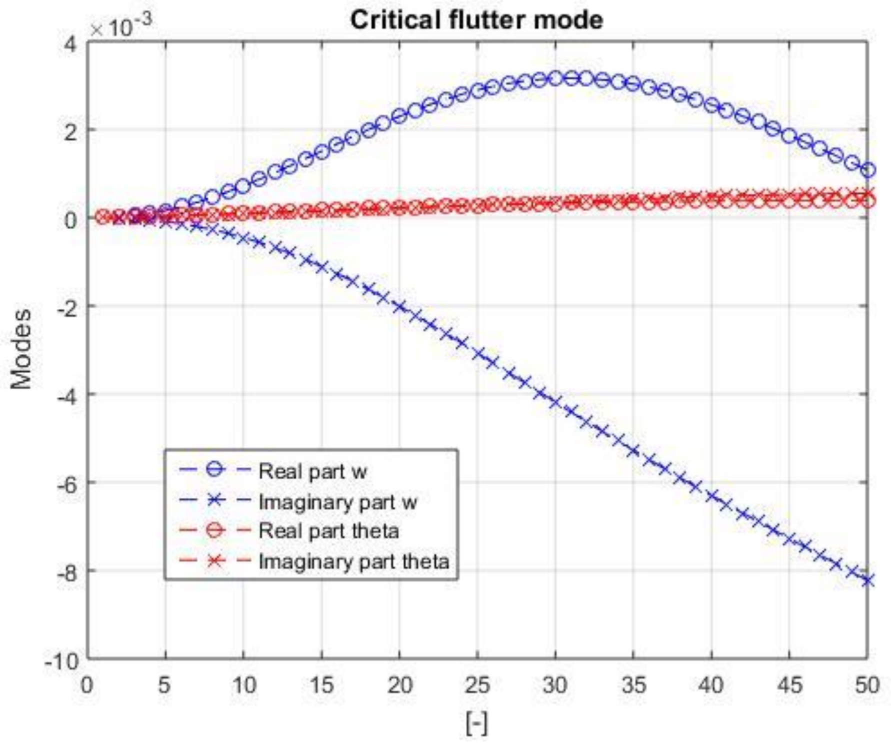


Figure 14: Variation of engine position. Modes Case 2 (from program)

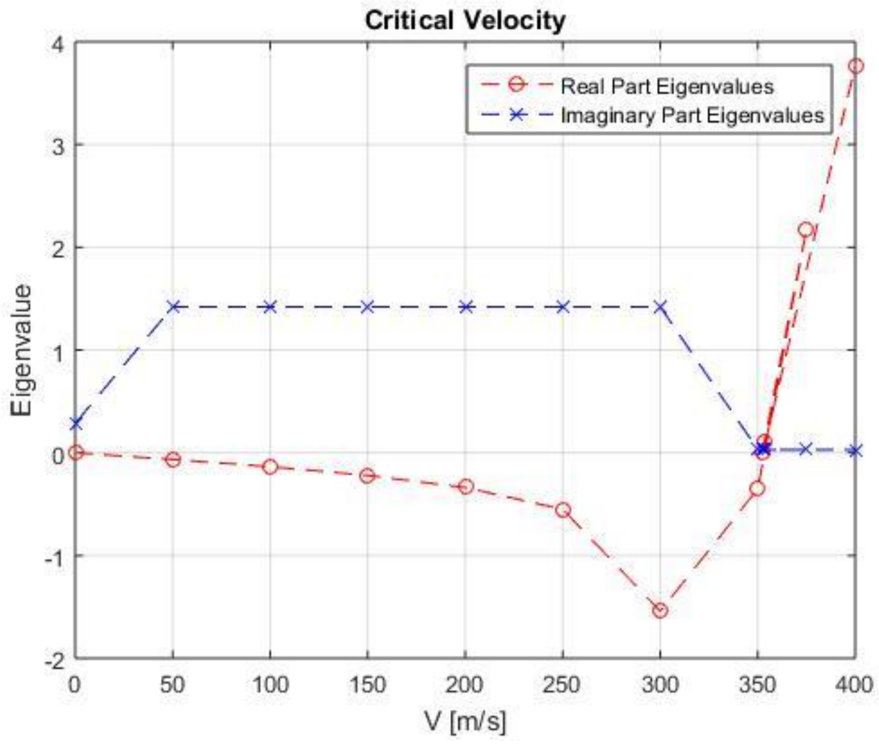


Figure 15: Variation of engine position. Eigenvalues Case 2 (from program)

U_crit=354,276 m/s.

The critical speed is in this case, greater than the speed of the base case. That is, the flutter will take longer to produce. Bending modes are smoother.

Now we change the values in the x-axis.

<u>Case 3</u>

x0_e = 2,0m;

y0_e = 3,35m;

z0_e = -1,0m;

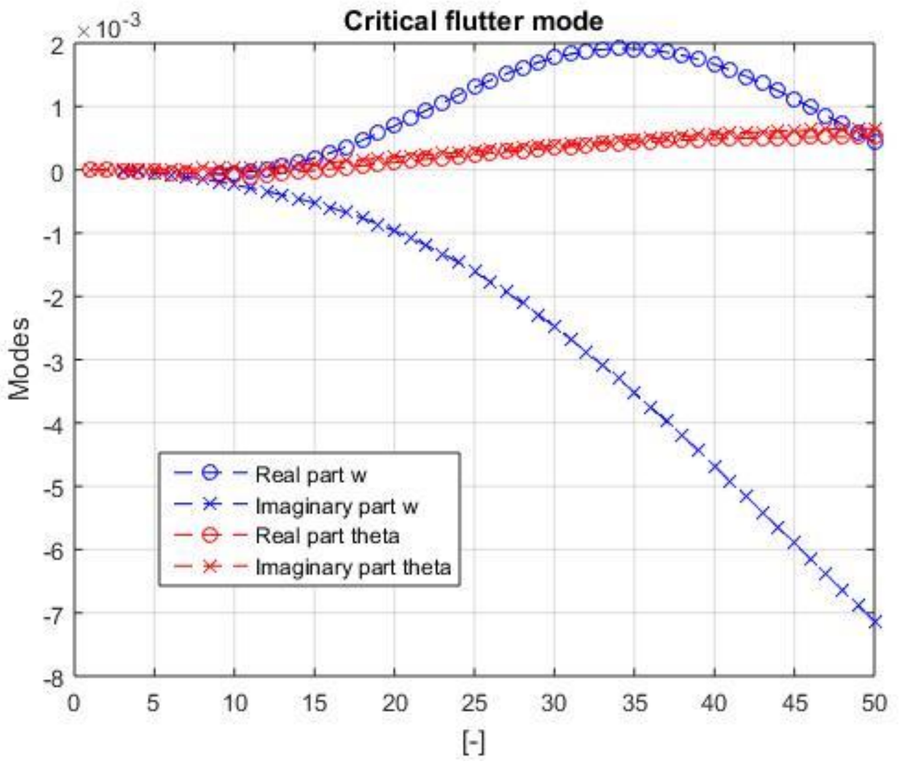


Figure 16: Variation of engine position. Modes Case 3 (from program)

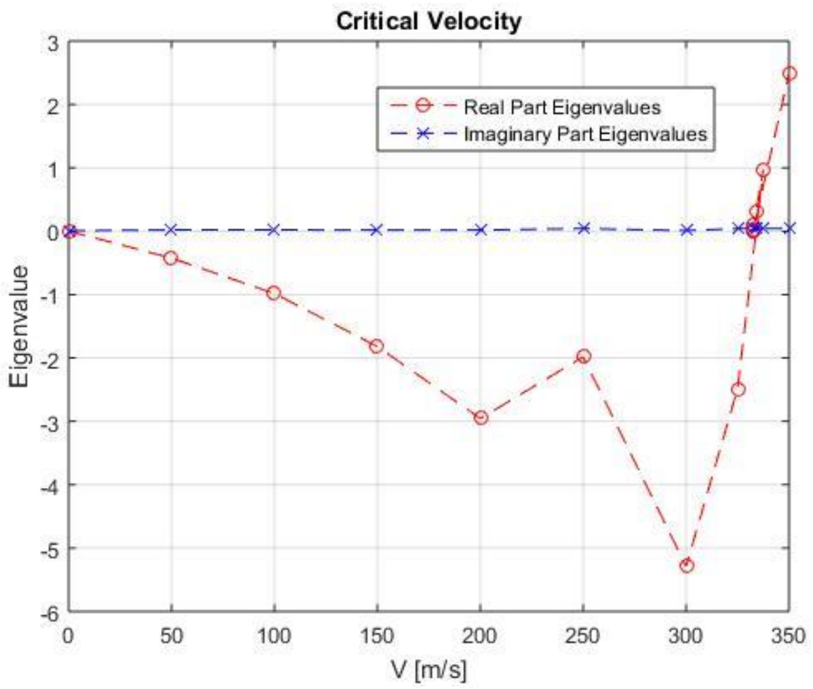


Figure 17: Variation of engine position. Eigenvalues Case 3 (from program)

U_crit=333,981m/s.

The speed is somewhat lower than in the base case.

If, on the other hand, the position of the motors is delayed:

<u>Case 4</u>

x0_e = -2,0m;

y0_e = 3,35m;

z0_e = -1,0m;

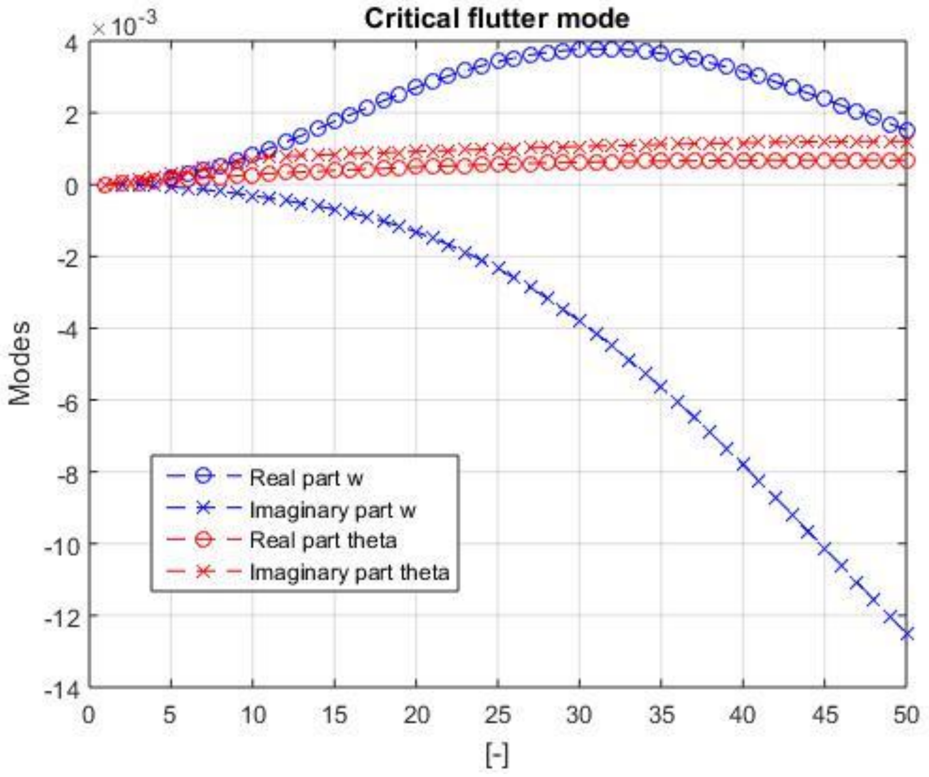


Figure 18: Variation of engine position. Modes Case 4 (from program)

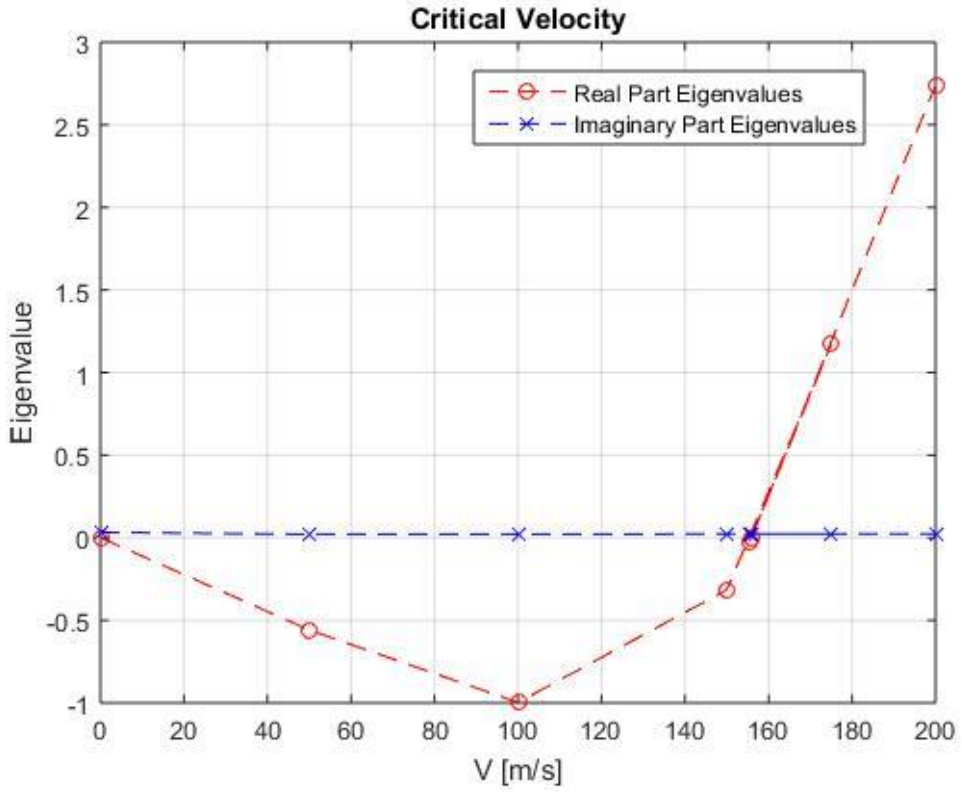


Figure 19: Variation of engine position. Eigenvalues Case 4 (from program)

U_crit=155,169m/s.

The critical speed is even lower.

And finally the position on the z axis is varied.

Case 5

x0_e = 0,0m;

y0_e = 3,35m;

z0_e = 2,0m;

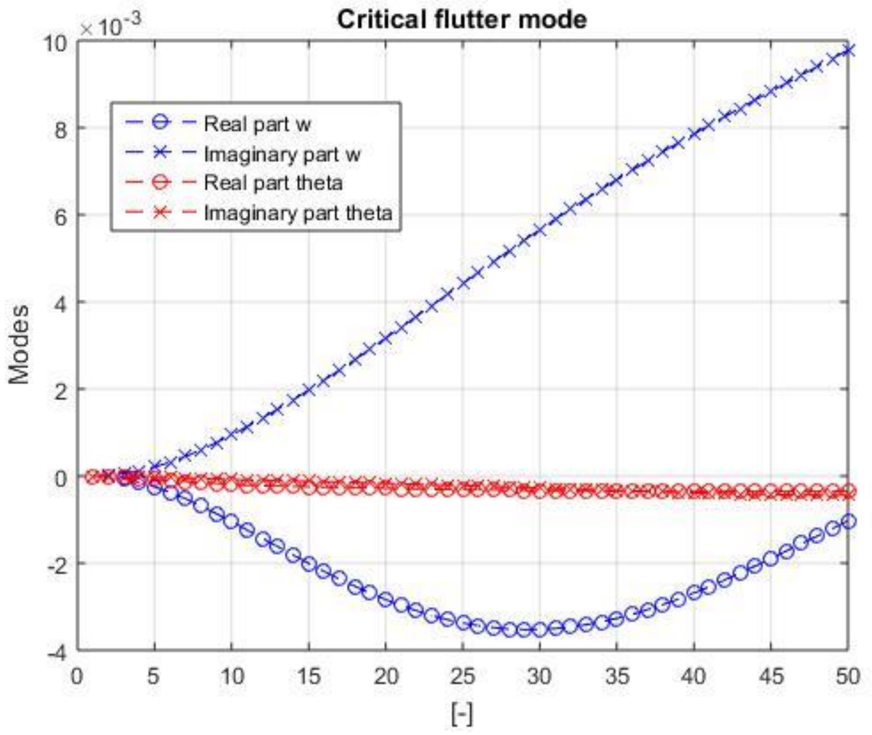


Figure 20: Variation of engine position. Modes Case 5 (from program)

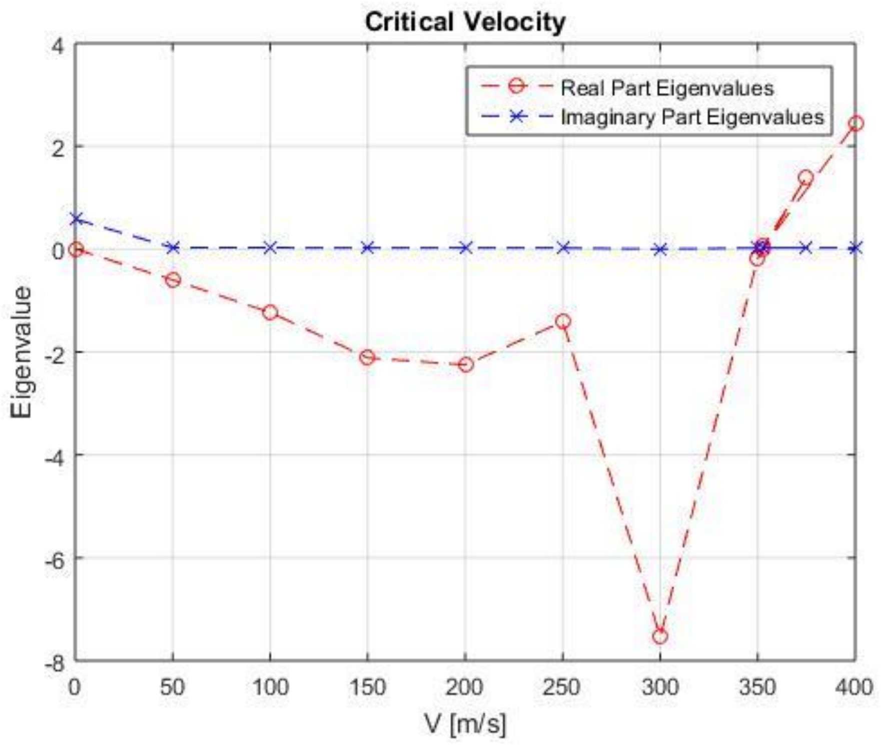


Figure 21: Variation of engine position. Eigenvalues Case 5 (from program)

U_crit=353,559 m/s.

By increasing the height of the motors, we also increase the critical flutter speed.

<u>Case 6</u>

x0_e = 0,0m;

y0_e = 3,35m;

z0_e = -3,0m;

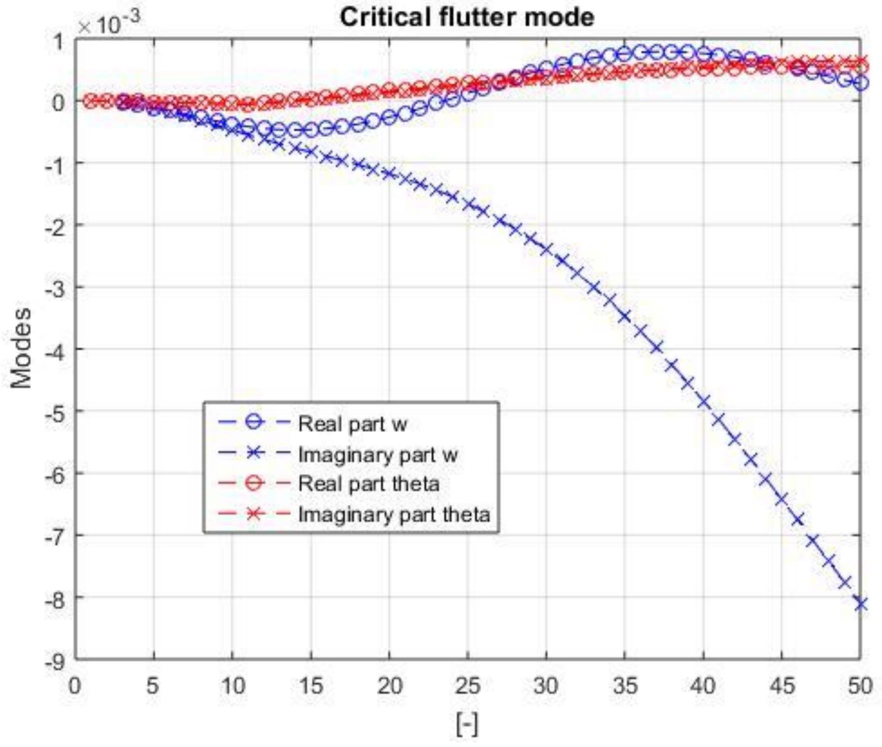


Figure 22: Variation of engine position. Modes Case 6 (from program)

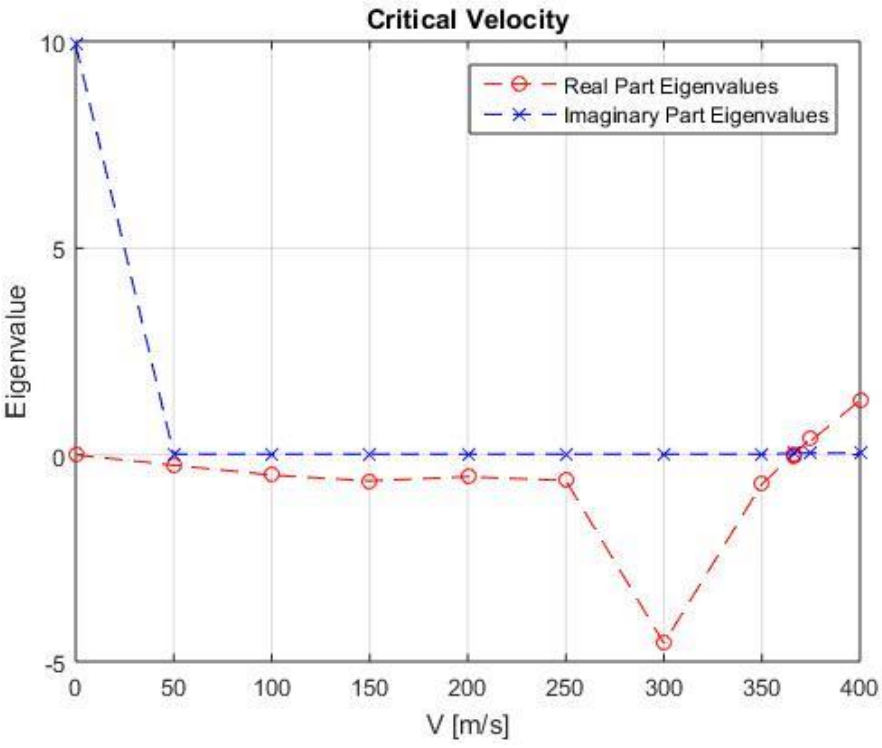


Figure 23: Variation of engine position. Eigenvalues Case 6 (from program)

U_crit=367,464m/s.

With a lower engine height, an increase in critical speed is again achieved.

6.2.2 Variation of dihedral angle

The dihedral angle of a wing is the upward angle of an aircraft's wing, from the wing root to the wing tip. The amount of dihedral determines the amount of inherent stability along the roll axis.

Now we will check how this wing parameterization angle affects the appearance of the flutter.

<u>Base case:</u>

We calculate the dihedral angle of the base case from the wing span and the vertical distance between the wing root and the wing tip:

dihedral = tan(1.82 / 15.07)=6,8862 degrees;

From here four changes in the dihedral are performed.

<u>Case 1</u>

dihedral = 0.0 degrees;

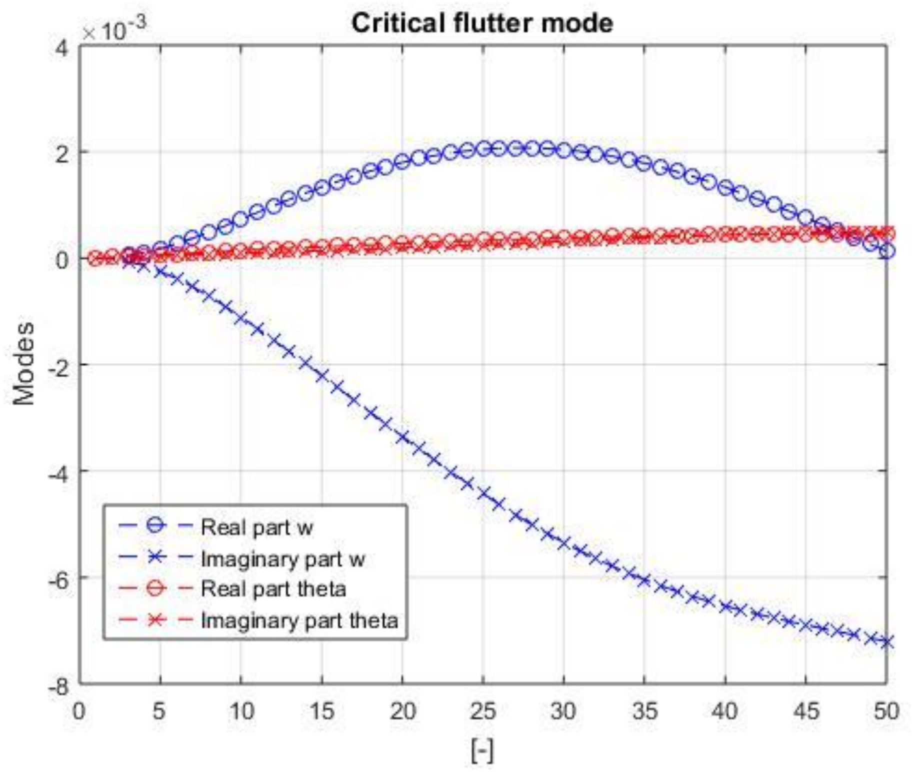


Figure 24: Variation of Dihedral angle. Modes Case 1 (from program)

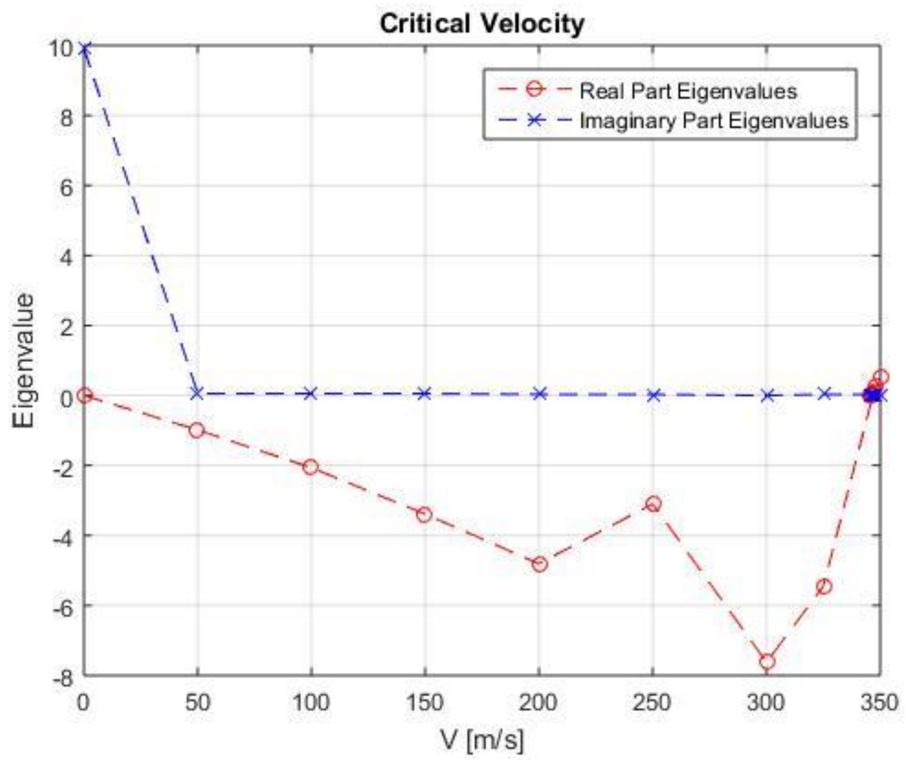


Figure 25: Variation of Dihedral angle. Eigenvalues Case 1 (from program)

U_crit= 346,789m/s.

If the wing is modelled completely horizontally, no change in the critical flutter speed occurs.

<u>Case 2</u>

Diedral=15 degrees;

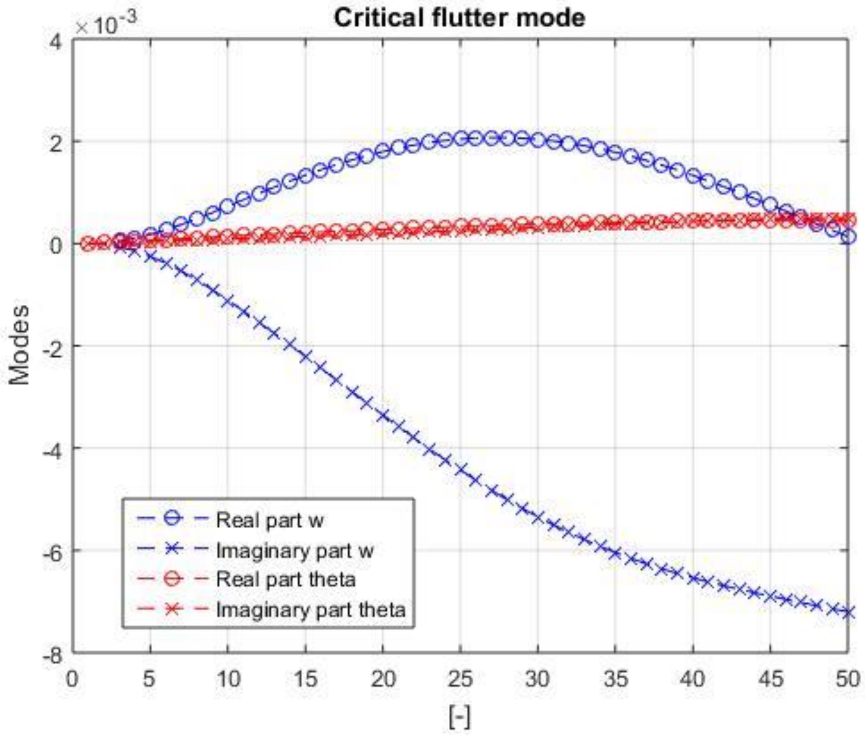


Figure 26: Variation of Dihedral angle. Modes Case 2 (from program)

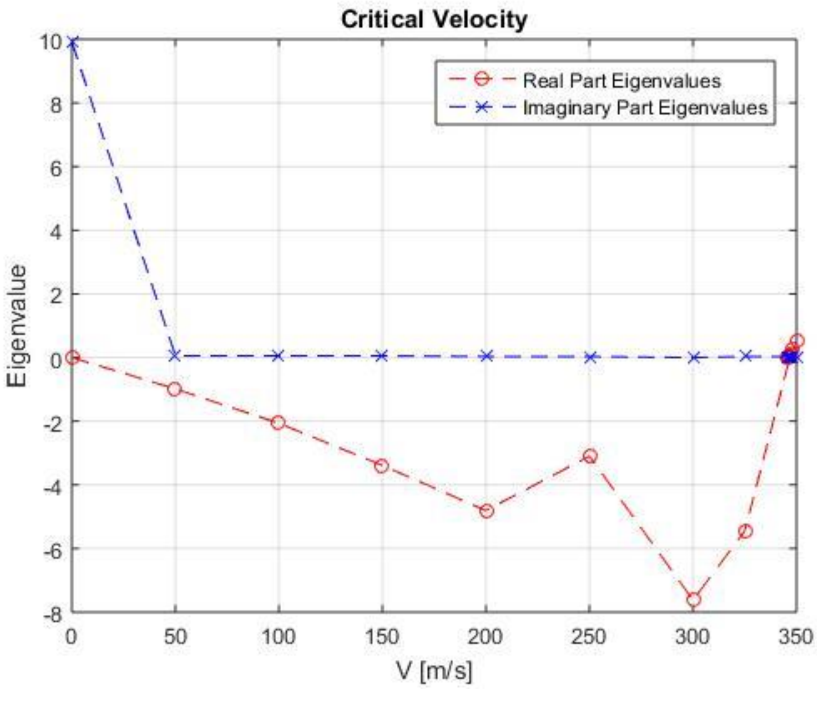


Figure 27: Variation of Dihedral angle. Eigenvalues Case 2 (from program)

U_crit=346,789m/s.

By increasing the dihedral angle by almost double, again no variations are observed in the flutter.

Case 3

Dihedral =60 degrees;

For this case, and given that previously no change has been made in the phenomenon under study, the dihedral angle is drastically increased.

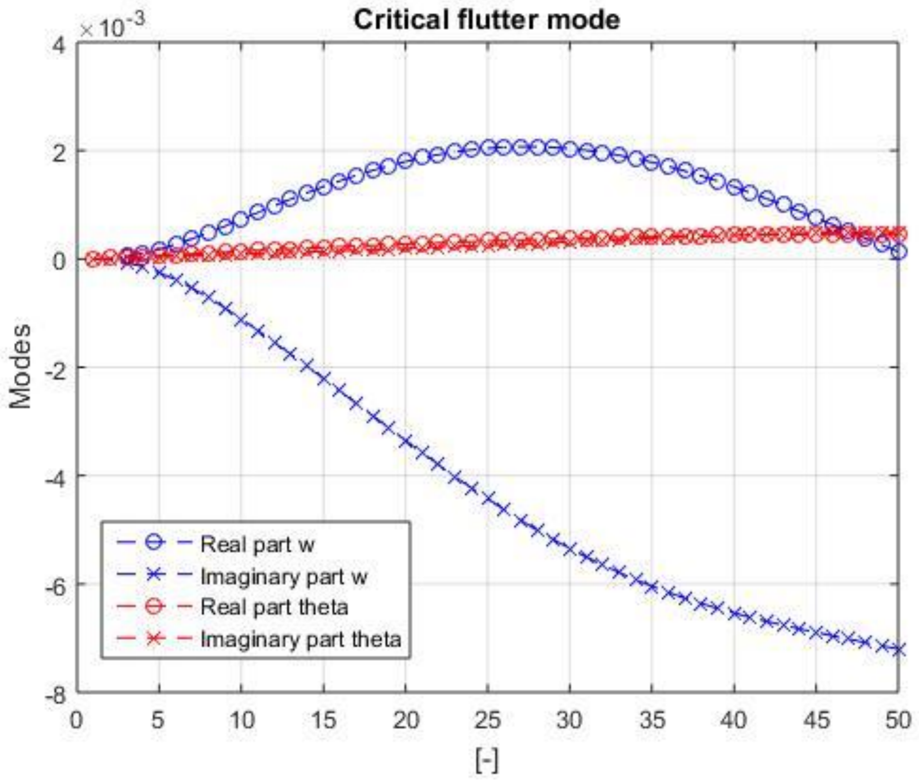


Figure 28: Variation of Dihedral angle. Modes Case 3 (from program)

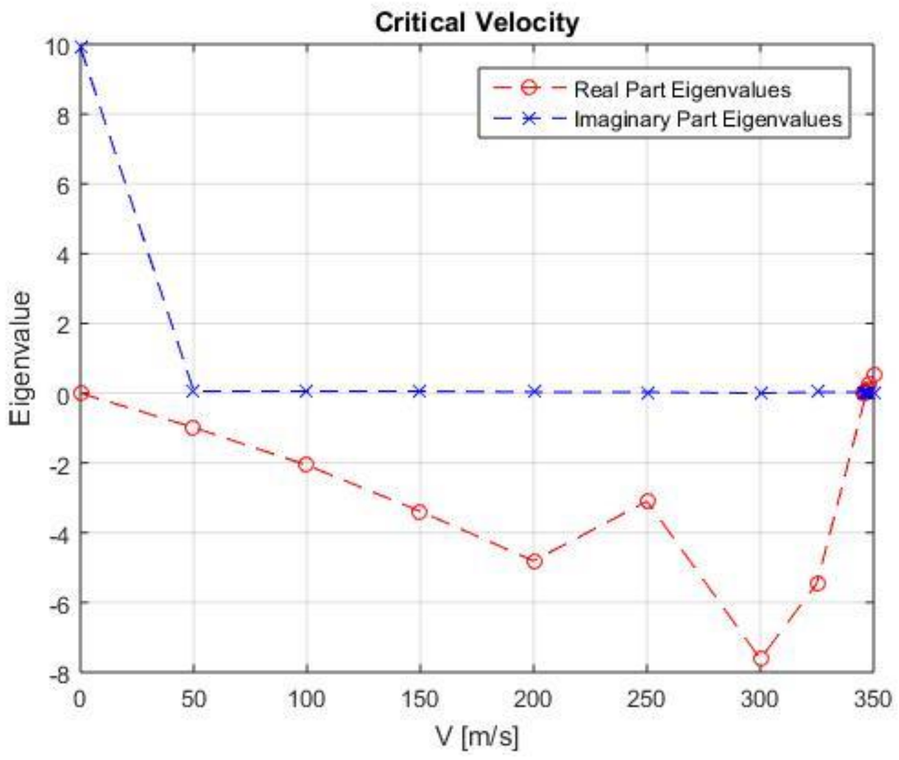


Figure 29: Variation of Dihedral angle. Eigenvalues Case 3 (from program)

U_crit=346,789m/s.

Again the same results are obtained.

To finish it is tested with the model of a wing with negative dihedral angle.

Case 4

Dihedral=-15 degrees;

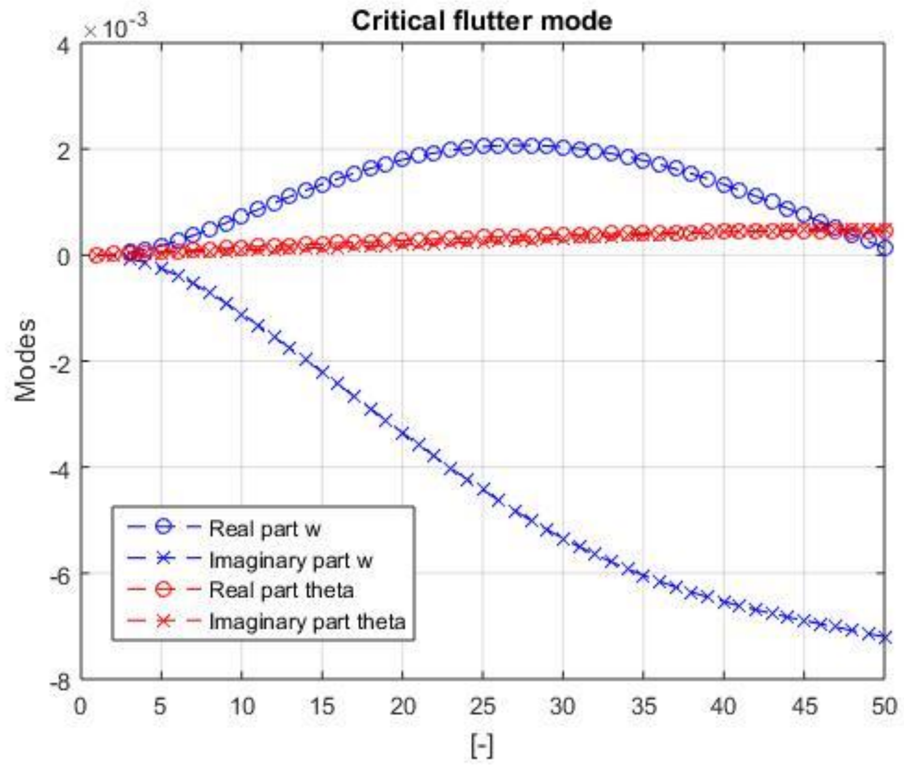


Figure 30: Variation of Dihedral angle. Modes Case 4 (from program)

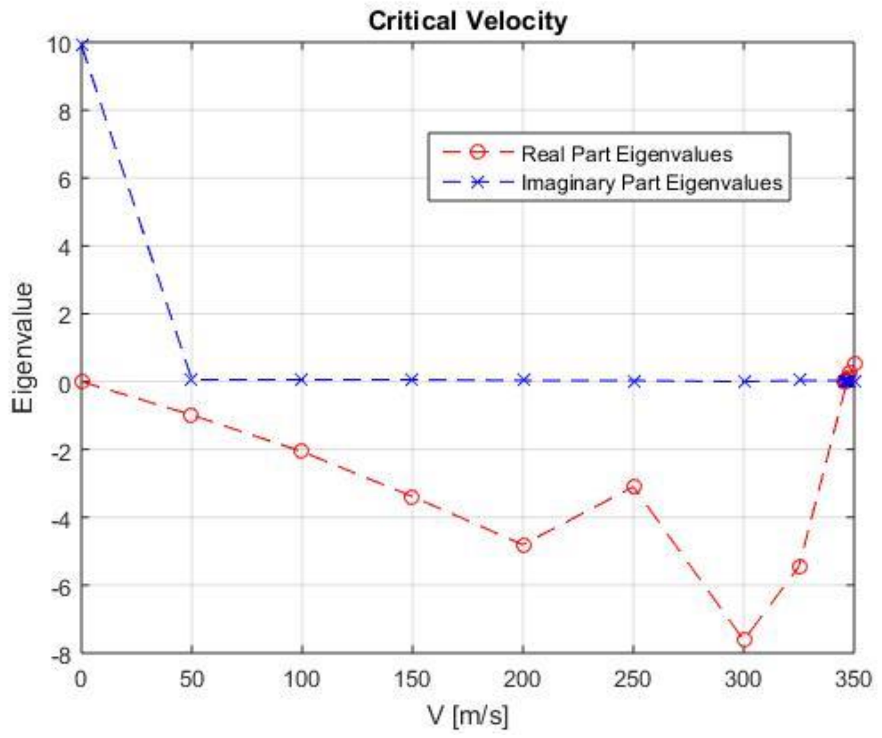


Figure 31: Variation of Dihedral angle. Eigenvalues Case 4 (from program)

U_crit=346,789m/s.

And as expected, we are again unchanged.

6.2.3 Variation of wing sweep angle

The sweep angle in a wing refers to the angle between a horizontal line in the span direction and the quarter chord line.

<u>Base Case</u>

We calculate the sweep angle of the base case from the wing span and the distance between the point $P_r = 0,25 \cdot chord_{root}$ and the point $P_t = 0,25 \cdot chord_{tip}$.

sweep = tan(6.0 / 15.07)=21,7096 degrees

<u>Case 1</u>

Sweep=0 degrees;

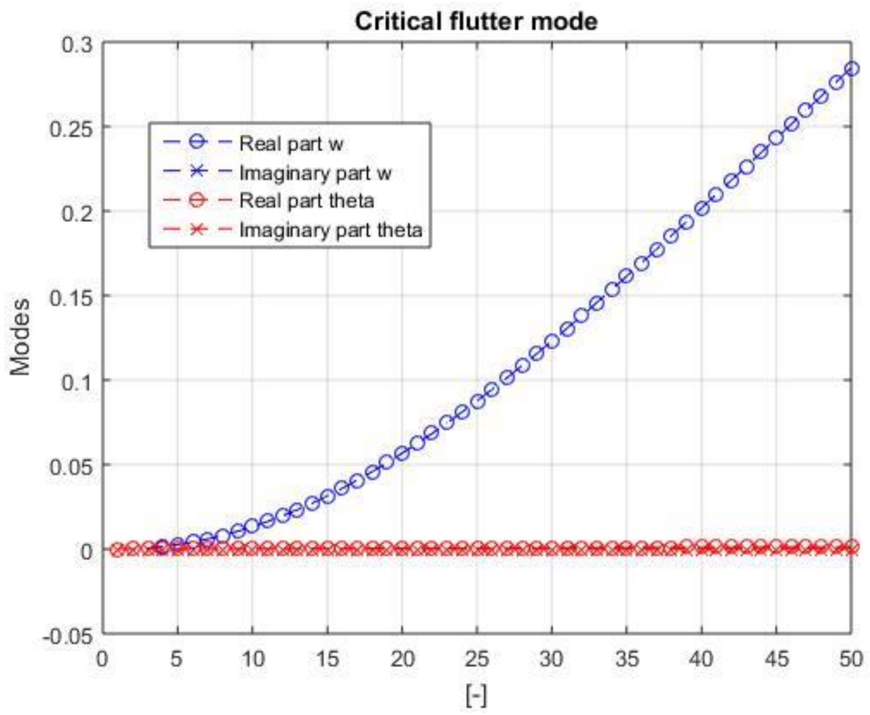


Figure 32: Variation of Sweep angle. Modes Case 1 (from program)

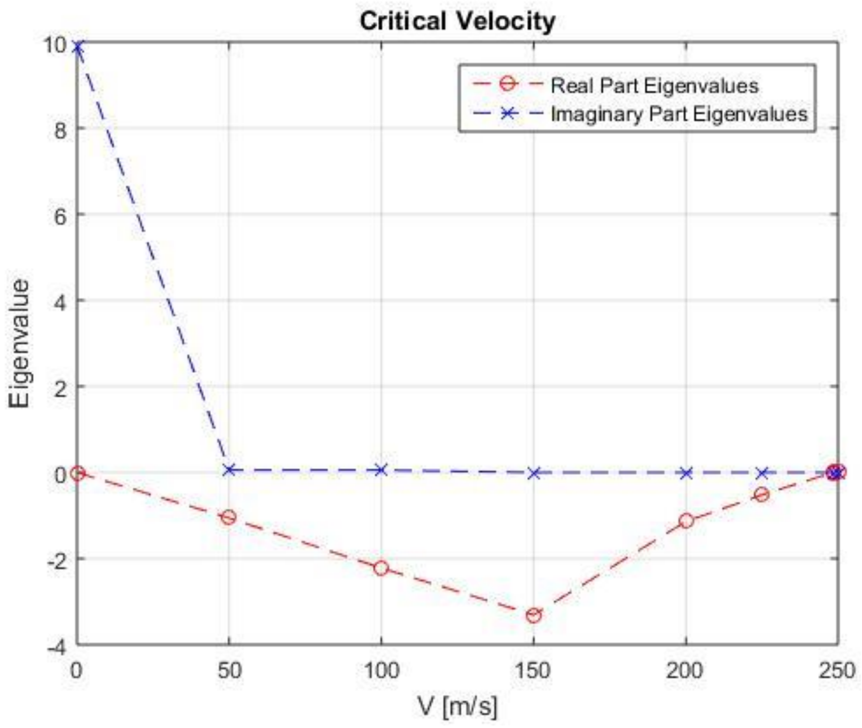


Figure 33: Variation of Sweep angle. Eigenvalues Case 1 (from program)

U_crit=248,530m/s.

The flutter occurs after the base case, but the value of the eigenvectors of the vibration modes is much higher than in the base case, that is, the deformed is much more pronounced.

<u>Case 2</u>

Sweep=6 degrees;

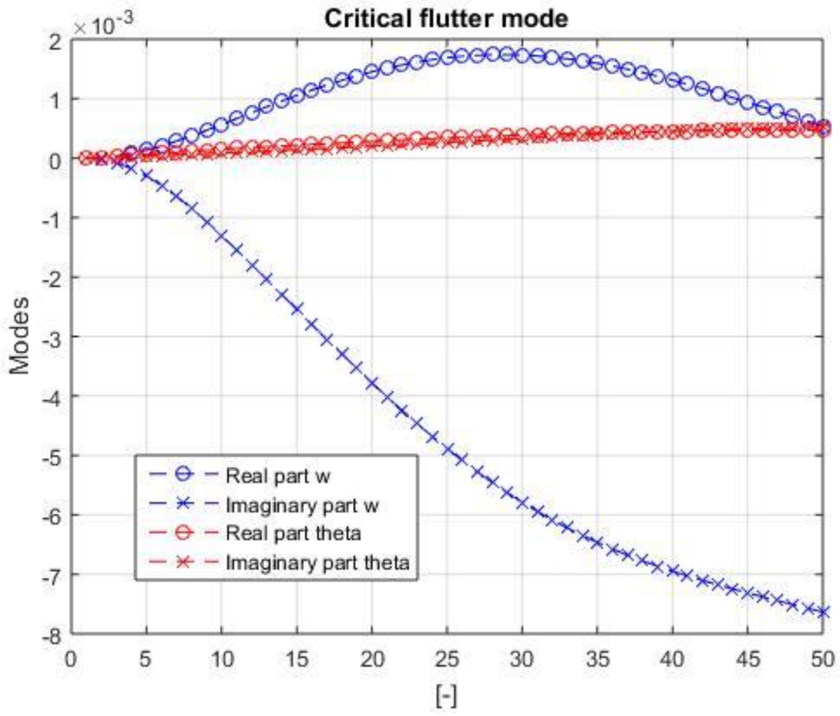


Figure 34: Variation of Sweep angle. Modes Case 2 (from program)

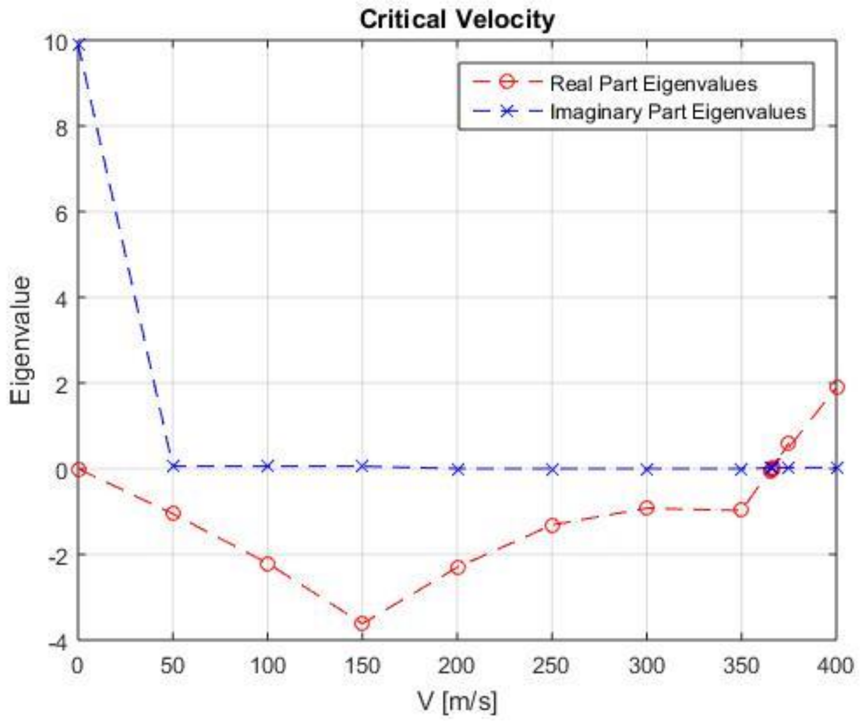


Figure 35: Variation of Sweep angle. Eigenvalues Case 2 (from program)

U_crit=367,059m/s.

In this case, the critical speed is even greater than that of the base case. We are, then, close to an optimum in terms of avoidance of flutter.

Case 3

Sweep=12 degrees;

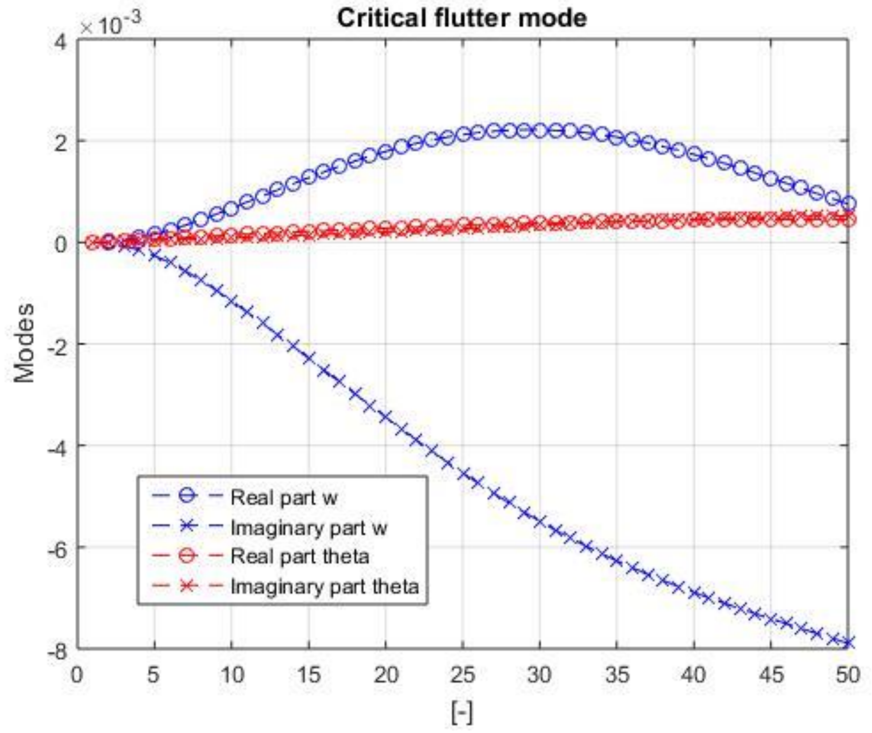


Figure 36: Variation of Sweep angle. Modes Case 3 (from program)

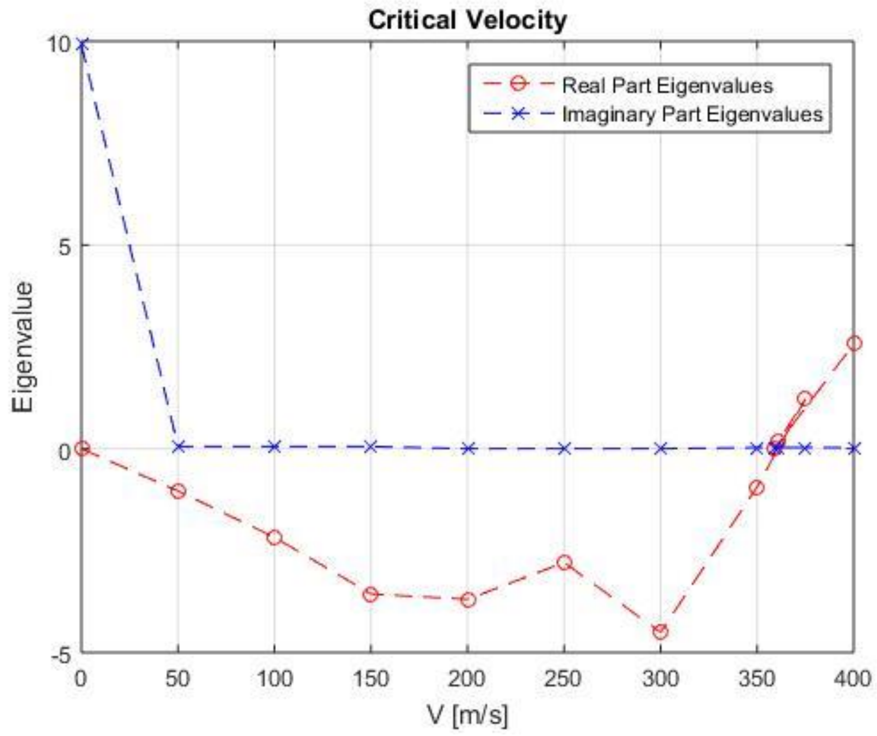


Figure 37: Variation of Sweep angle. Eigenvalues Case 3

U_crit=363,347 m/s.

Now the wing behaves very similar to the previous one, both in terms of critical speed and vibration modes.

Case 4

Sweep=30 degrees;

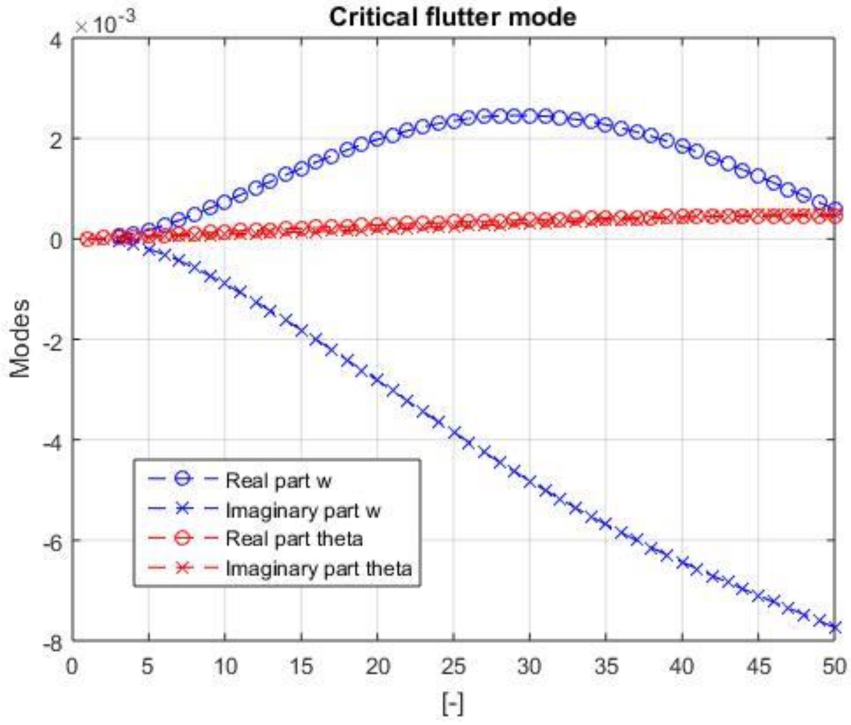


Figure 38: Variation of Sweep angle. Modes Case 4 (from program)

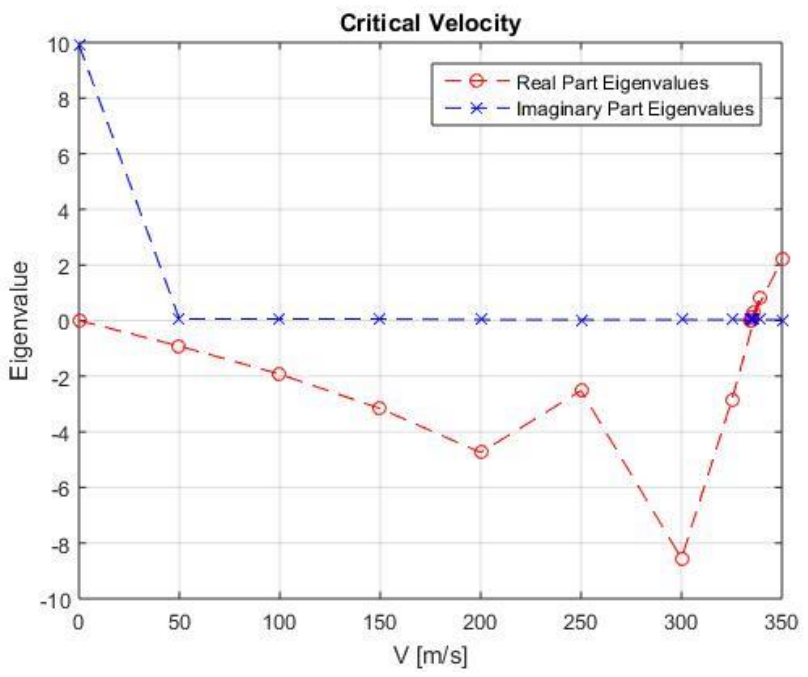


Figure 39: Variation of Sweep angle. Eigenvalues Case 4 (from program)

U_crit=339,822m/s.

In this last section we find that, from a certain point, to continue increasing the sweep angle worsens the vibrational conditions.

6.2.4 Variation of wing bending stiffness distribution

The bending stiffness (K) is the resistance of a member against bending deformation. It is a function of the Young's modulus (E), and the area moment of inertia (I).

In our model, the bending stiffness distribution of the wing is considered constant.

In the last two sections the flutter modes become more important, as we can see how the deformation varies in displacement and in torsion by varying the bending stiffness distribution first, and the torsional stiffness distribution afterwards.

Base case

EI(i) $= 7.0\text{e}{+}06\ Nm^2$;

Case 1

$$\text{EI(i)} \quad = 7.0\text{e}{+}07\ Nm^2;$$

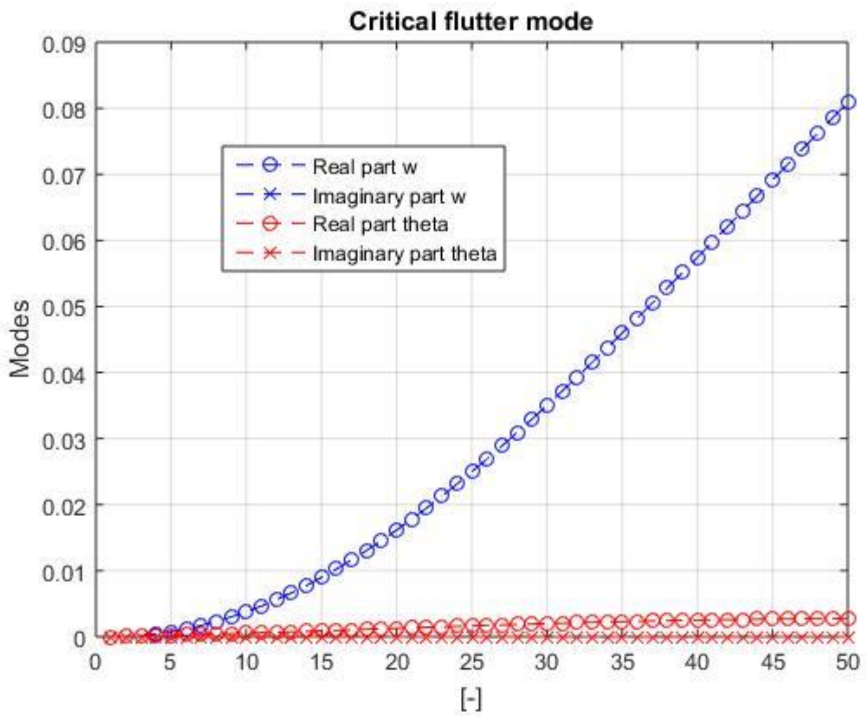


Figure 40: Variation of Bending stiffness. Modes Case 1 (from program)

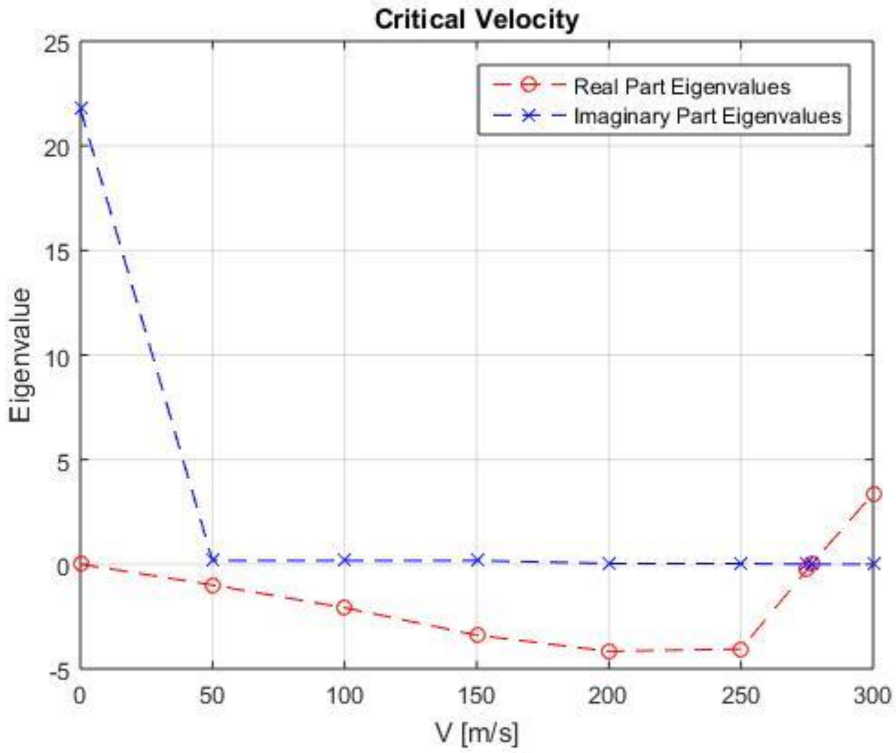


Figure 41: Variation of Bending stiffness. Eigenvalues Case 1 (from program)

U_crit=277,318m/s.

The critical flutter speed is lower than the base case. That is, flutter happens before. In addition, we see an increase in the modes of displacement.

Case 2

$$\text{EI(i)} \quad = 7.0\text{e}{+}05\, Nm^2;$$

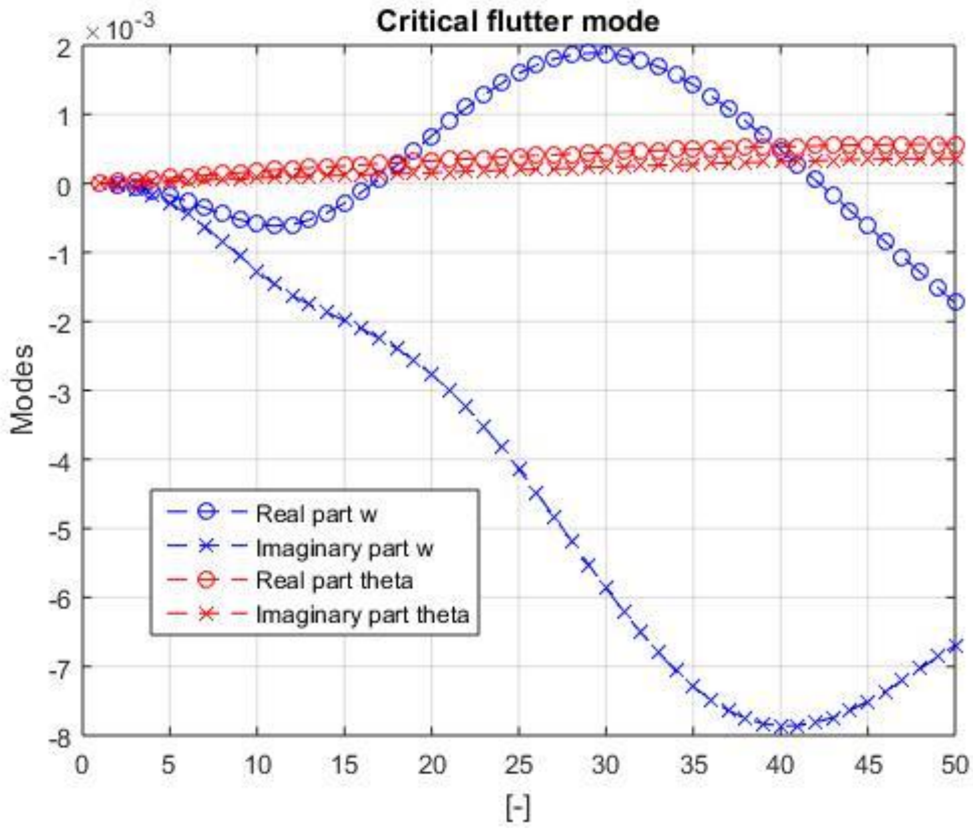
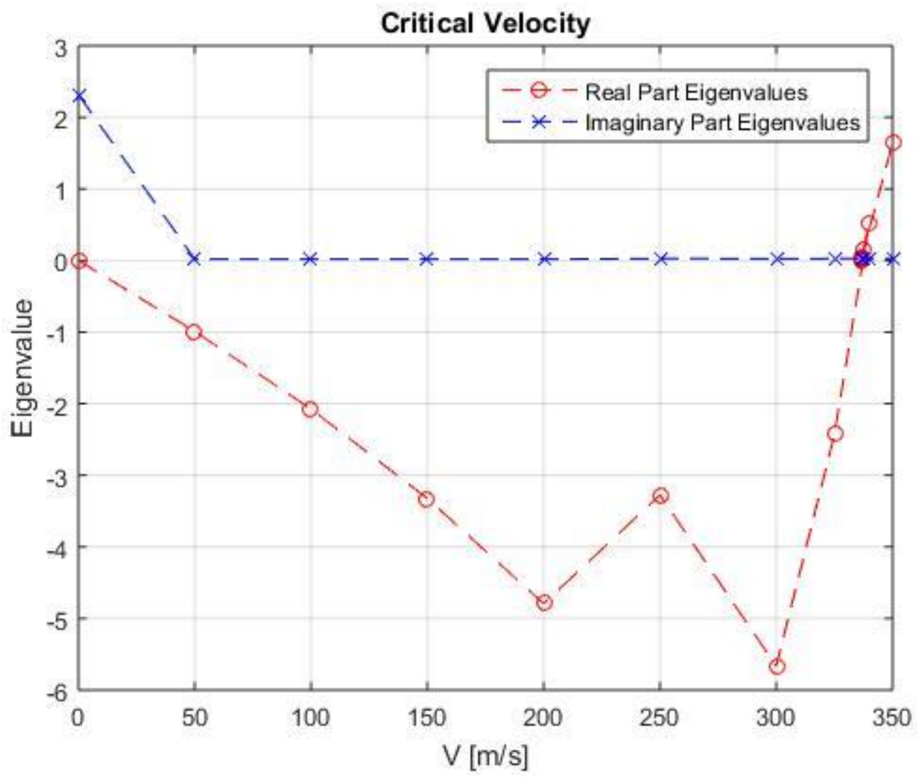


Figure 42: Variation of Bending stiffness. Modes Case 2 (from program)



e

Figure 43: Variation of Bending stiffness. Eigenvalues Case 2 (from program)

U_crit=338,735m/s.

By reducing the bending stiffness with respect to the previous case, the critical speed of flutter increases. And what is more interesting, the modes corresponding to the vertical displacement present, for the first time, more than one peak.

<u>Case 3</u>

$$\text{EI(i)} \quad = 7.0\text{e}{+}04\, Nm^2;$$

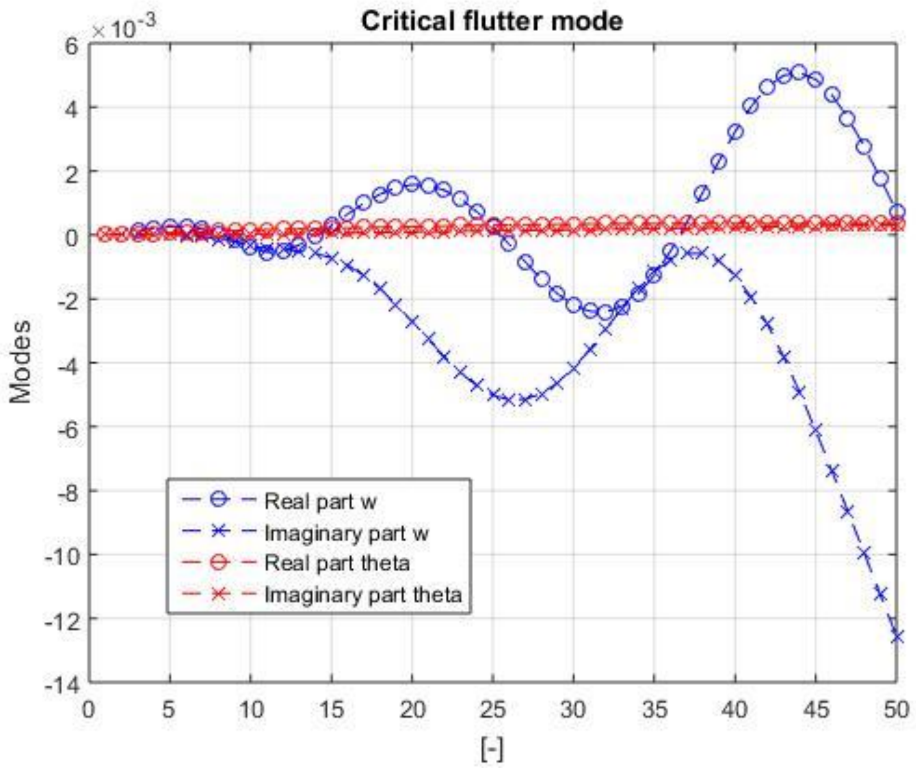


Figure 44: Variation of Bending stiffness. Modes Case 3 (from program)

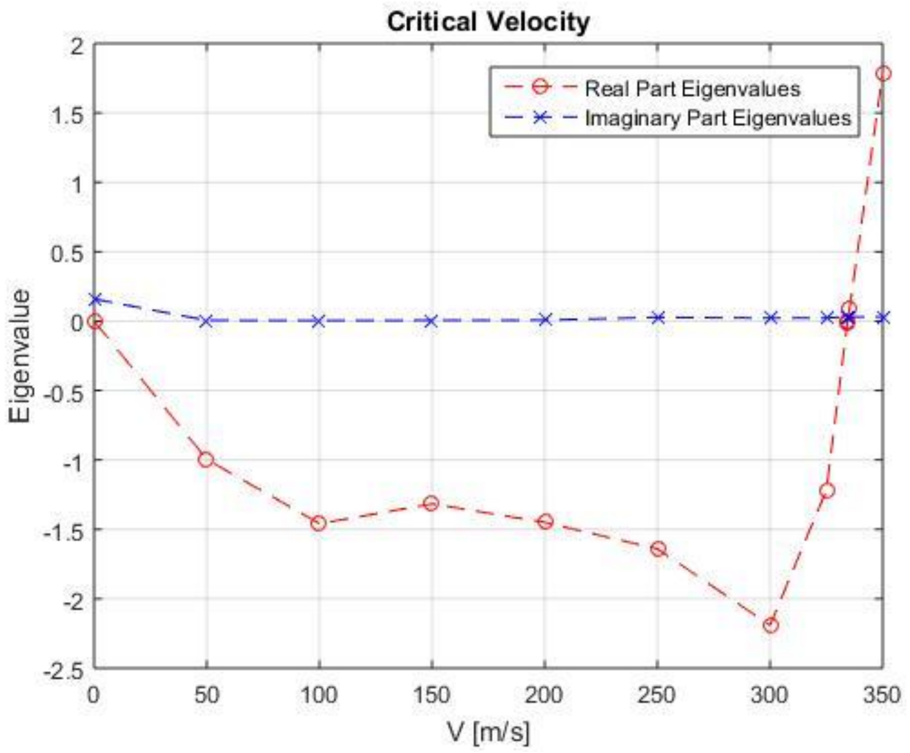


Figure 45: Variation of Bending stiffness. Eigenvalues Case 3 (from program)

U_crit=327,540m/s.

We keep reducing the rigidity, and the modes each time have more number of peaks.

Case 4

$$\text{EI(i)} \quad = 7.0\text{e+}03\, Nm^2;$$

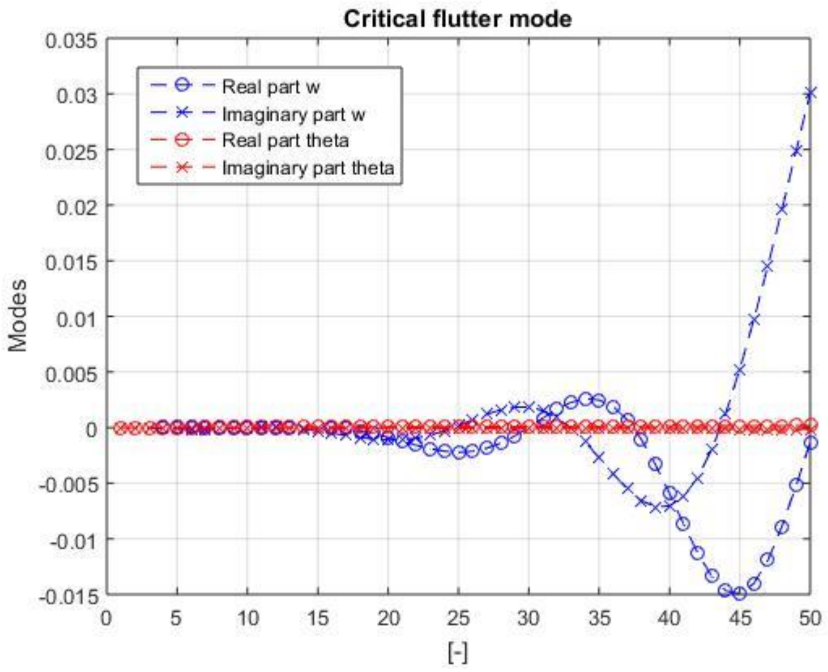


Figure 46: Variation of Bending stiffness. Modes Case 4 (from program)

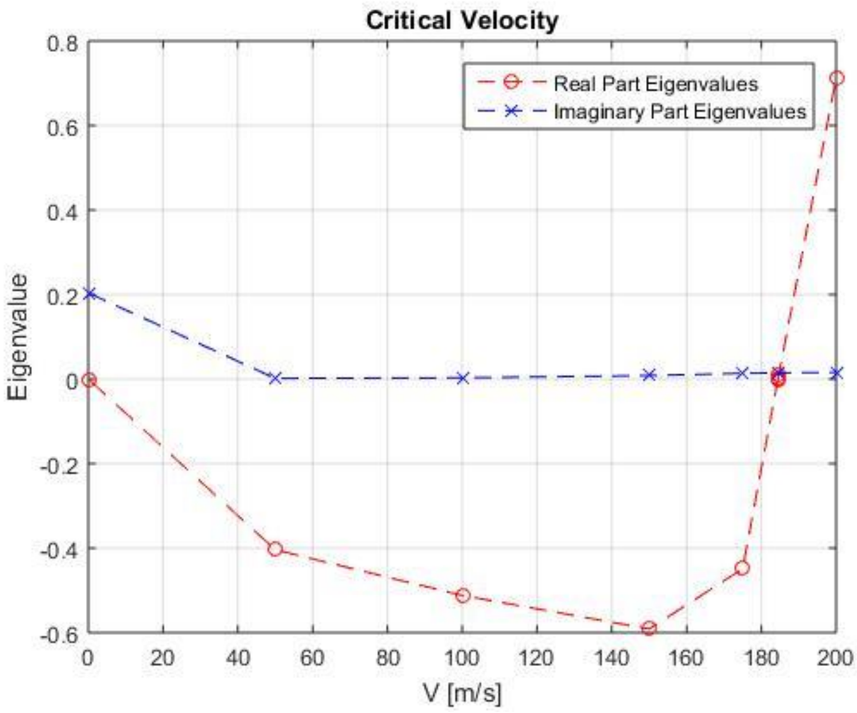


Figure 47: Variation of Bending stiffness. Eigenvalues Case 4 (from program)

U_crit=172,656m/s.

With even less stiffness the flutter occurs at a very low speed, and the peaks in the vibration modes are concentrated on the right side of the plot.

6.2.5 Variation of torsional stiffness

The torsional stiffness refers to the resistance of a member against torsion deformation. Again, in the model torsional stiffness has been considered constant over the wing.

<u>Base Case</u>

$$GJ(i) \quad = 9.5e+06\,\frac{Nm^2}{rad};$$

<u>Case 1</u>

$$GJ(i) \quad = 1e+07\,\frac{Nm^2}{rad};$$

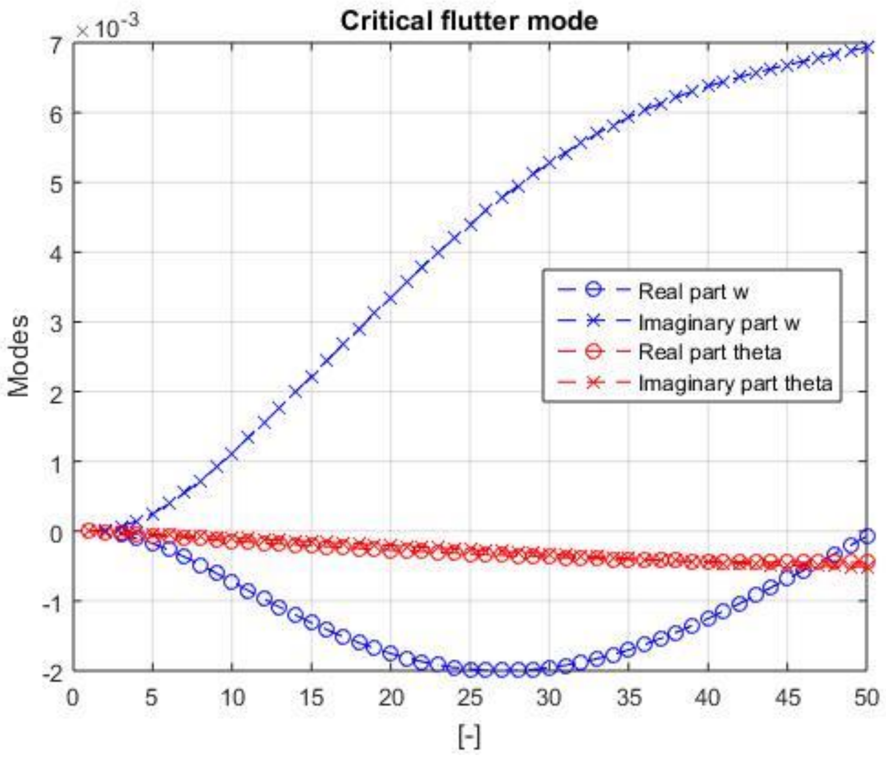


Figure 48: Variation of Torsional stiffness. Modes Case 1 (from program)

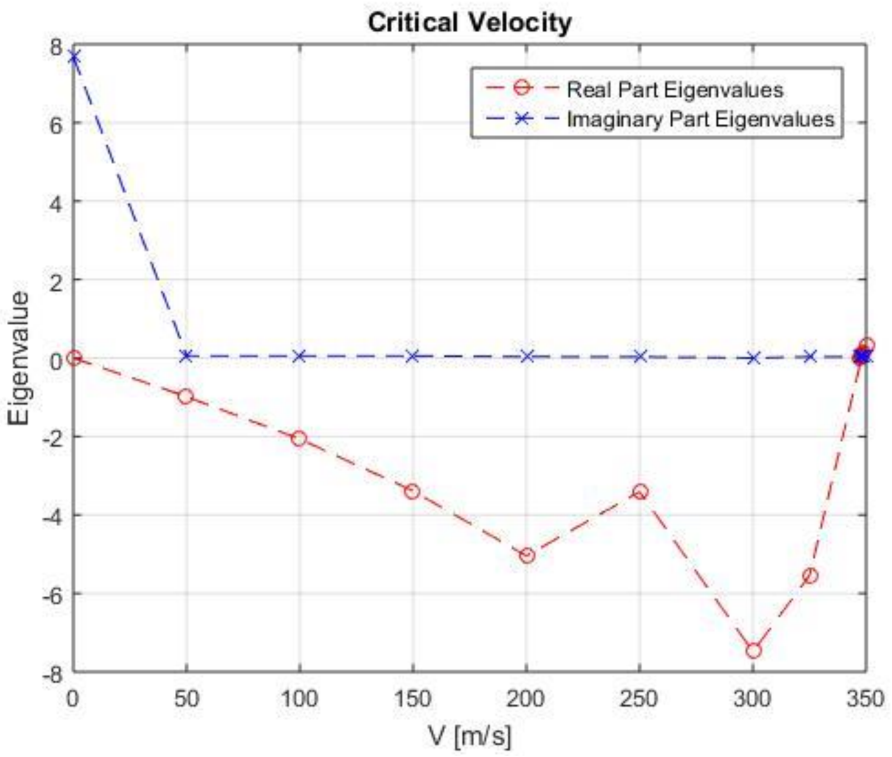


Figure 49: Variation of Torsional stiffness. Eigenvalues Case 1 (from program)

U_crit=347,936m/s.

Increasing torsional stiffness, critical velocity increases.

<u>Case 2</u>

$$GJ(i) \quad = 9.5e{+}07\ \frac{Nm^2}{rad};$$

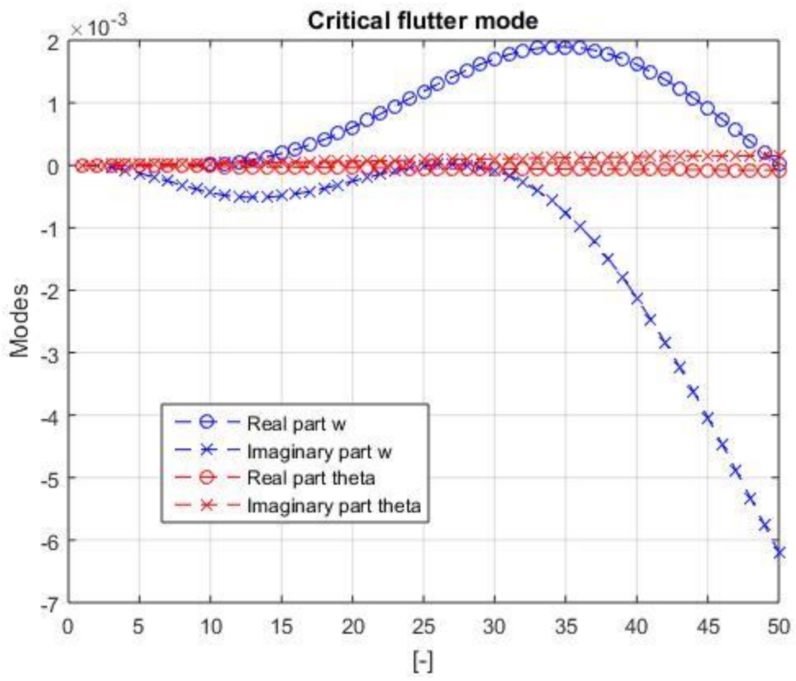


Figure 50: Variation of Torsional stiffness. Modes Case 2 (from program)

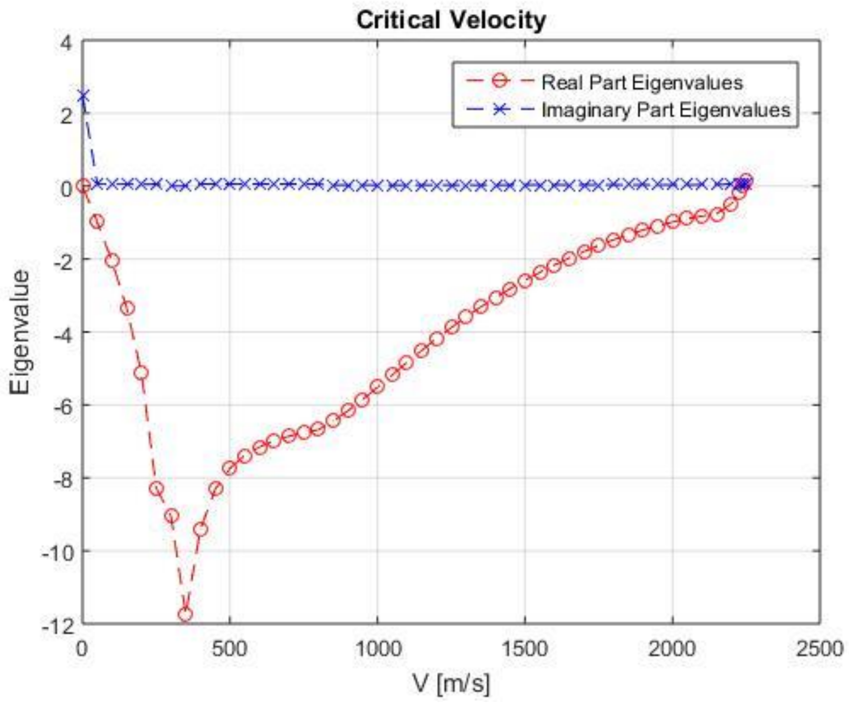


Figure 51: Variation of Torsional stiffness. Eigenvalues Case 2 (from program)

U_crit=2236,994m/s.

If we increase torsion stiffness in one order of magnitude, the critical velocity increases and reaches the biggest value until now.

Case 3

$$\text{GJ(i)} \quad = 9.5\text{e+}05\,\frac{Nm^2}{rad};$$

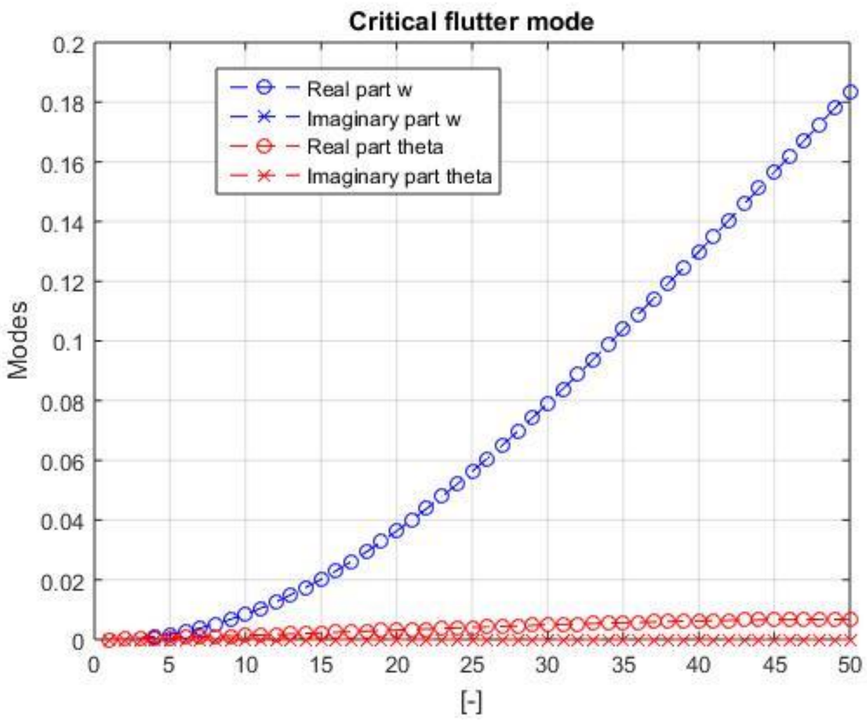


Figure 52: Variation of Torsional stiffness. Modes Case 3 (from program)

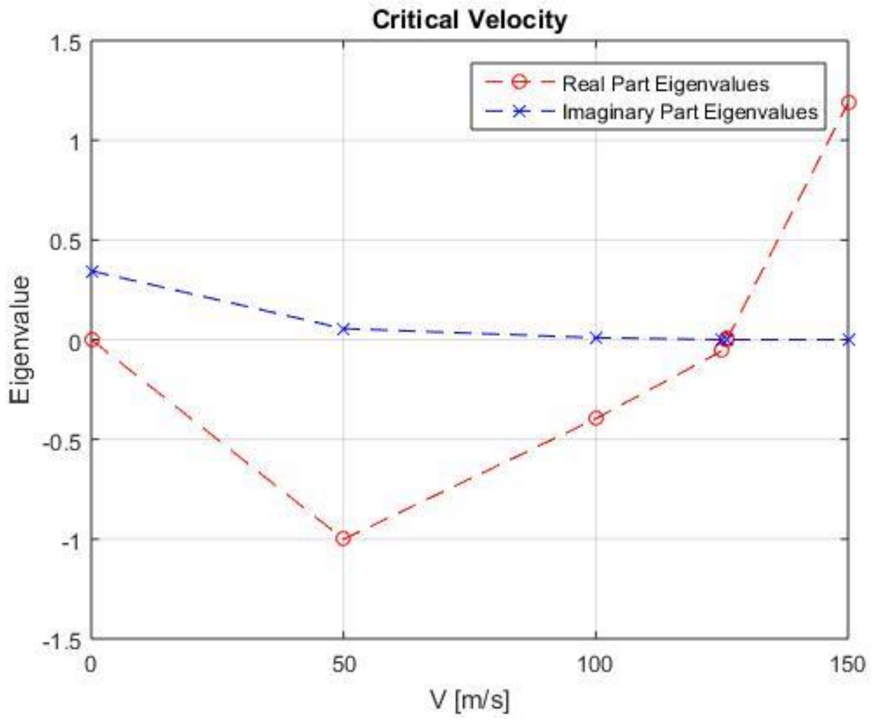


Figure 53: Variation of Torsional stiffness. Eigenvalues Case 3 (from program)

U_crit=112,505m/s.

On the other hand, decreasing torsional stiffness the conditions for the appearance of the flutter aggravate.

Case 4

$$GJ(i) \quad = 9.5e{+}04\ \frac{Nm^2}{rad};$$

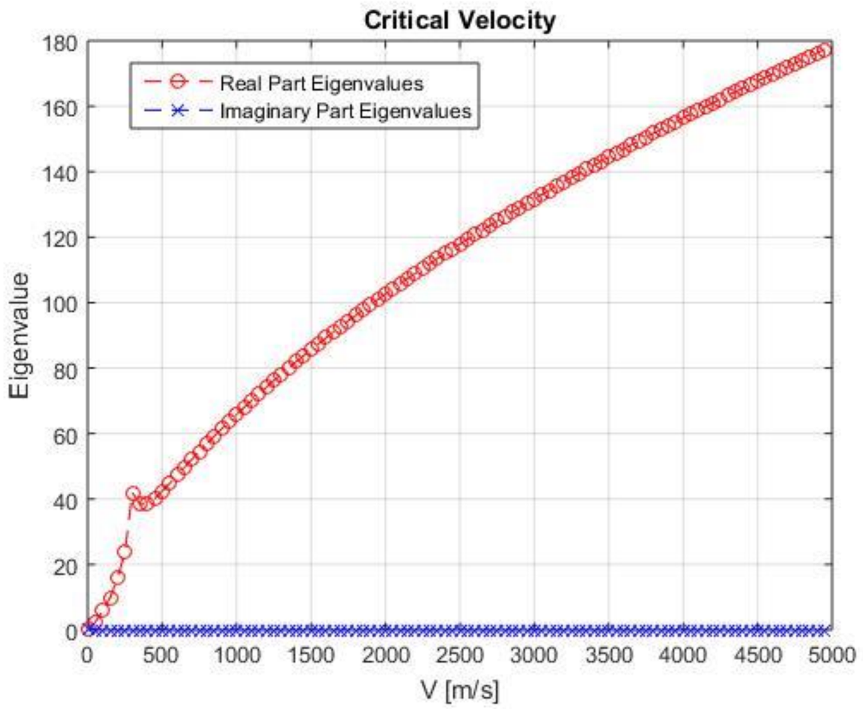


Figure 54: Variation of Torsional stiffness. Eigenvalues Case 4 (from program)

No u crit

Finally, with a lower torsional stiffness we reach a point in which the conditions for the simulation are not fulfilled, and we can't get a value of critical velocity.

7 Conclusions

Firstly, we must bear in mind that aircraft designers must reach a compromise between all the needs presented to them (stability, price, vibrations ...). Therefore, in some cases we verify that the parameters used in the design of the wing of the A320 are not optimal in terms of preventing the occurrence of flutter.

It should be noted too that analysis exhibits that the effect of vertical displacement modes stands out above the degree of torsional displacement in all cases. That is why, although the deformed shape in vertical displacement shows sometimes more than one peak, it never happens in the case of torsional displacement.

Here there are all the results obtained summarized in a table:

		Flutter Velocity		Flutter Modes		
		Critical Velocity [m/s]	↑↓ From base case	Vertical Displacement		Torsional Displacement
				Range Eigenvectors	Number Peaks/ Valleys	Range Eigenvectors
Base Case		346,789	-	$[0;10]\cdot10^{-3}$	1	$[0;0,5]\cdot10^{-3}$
Variation Engines [m]	y=14	227,328	↓	$[-32;10]\cdot10^{-3}$	1	$[0;1,5]\cdot10^{-3}$
	y=5	354,276	↑	$[-8;3]\cdot10^{-3}$	1	$[0;0,7]\cdot10^{-3}$
	x=2	333,981	↓	$[-7;2]\cdot10^{-3}$	1	$[0;0,8]\cdot10^{-3}$
	x=-2	155,169	↓	$[-12;4]\cdot10^{-3}$	1	$[0,7;1,7]\cdot10^{-3}$
	z=2	353,559	↑	$[-3,5;10]\cdot10^{-3}$	1	$[0;-0,2]\cdot10^{-3}$
	z=-3	367,464	↑	$[-8;1]\cdot10^{-3}$	2	$[0;0,7]\cdot10^{-3}$
Variation Dihedral Angle [º]	0	346,789	-	$[-7;2]\cdot10^{-3}$	1	$[0;0,5]\cdot10^{-3}$
	15	346,789	-	$[-7;2]\cdot10^{-3}$	1	$[0;0,5]\cdot10^{-3}$
	60	346,789	-	$[-7;2]\cdot10^{-3}$	1	$[0;0,5]\cdot10^{-3}$
	-15	346,789	-	$[-7;2]\cdot10^{-3}$	1	$[0;0,5]\cdot10^{-3}$
Variation Sweep Angle [º]	0	248,530	↑	$[0;280]\cdot10^{-3}$	1	$[0;0,7]\cdot10^{-3}$
	6	367,059	↑	$[-7,5;2]\cdot10^{-3}$	1	$[0;0,5]\cdot10^{-3}$
	12	363,347	↑	$[-8;2]\cdot10^{-3}$	1	$[0;0,5]\cdot10^{-3}$
	30	339,822	↓	$[-7,7;2,5]\cdot10^{-3}$	1	$[0;0,5]\cdot10^{-3}$
Variation Bending Stiffness $[Nm^2]$	$7\cdot10^7$	227,318	↓	$[0;80]\cdot10^{-3}$	1	$[0;5]\cdot10^{-3}$
	$7\cdot10^5$	338,735	↓	$[-8;2]\cdot10^{-3}$	3	$[0;700]\cdot10^{-3}$
	$7\cdot10^4$	327,540	↓	$[-13;5]\cdot10^{-3}$	5	$[0;0,5]\cdot10^{-3}$
	$7\cdot10^3$	172,656	↓	$[-15;30]\cdot10^{-3}$	4	$[0;0,1]\cdot10^{-3}$

Variation Torsional Stiffness $[\frac{Nm^2}{rad}]$	$9,5\cdot10^{7}$	2236,994	↑	$[-6;2]\cdot10^{-3}$	3	$[0;0,5]\cdot10^{-3}$
	$9,5\cdot10^{5}$	112,505	↓	$[0;180]\cdot10^{-3}$	1	$[0;10]\cdot10^{-3}$
	$9,5\cdot10^{4}$	-	-	-	-	-
	$1\cdot10^{7}$	347,936	↑	$[-2;7]\cdot10^{-3}$	1	$[0;0,5]\cdot10^{-3}$

Table 4: Final comparison (Author)

The final analysis of the variation of parameters has shown that:

- The most gravely vertical displacement deformed shape of the wing appears when the sweep angle is reduced to 0 and the when the value of the torsion stiffness is reduced in one order of magnitude. Curiously, the eigenvectors show a bigger range of vertical displacement when the torsional stiffness is reduced than when it is reduced the bending stiffness.
- The deformed shape in torsion shows the biggest range when decreasing in one order of magnitude the bending stiffness.
- That can be due to the coupling of the torsional and bending modes of vibrations. In fact, the energy inflow from the air flow, that creates the flutter, is controlled by the coupling.
- Nevertheless, the shape is more "deformed" in the sense that it shows more peaks when the bending stiffness is reduced. That makes sense, because if the bending stiffness is less, the resistance of the wing against bending is decreased, and therefore the whole wing will deform in more directions.
- There is an inverse relation between the number of peaks that appear in the mode shape and the value of the critical flutter velocity. But it does not exist such relation between the range of the modes and the velocity, or the range of the modes and the number of peaks.
- The variation of the dihedral angle does not derive any changes in the flutter conditions.
- Smaller sweep angles show better conditions for the flutter, because the critical velocity of flutter increases always that the sweep angle decreases, and decreases when the sweep angle is bigger than the base case.

- About the position of the engines, depending on the direction of the change of the position and its sense different results are shown. In general, moving the engines in the x axis always impair the flutter conditions. Moving engines in z direction from the base case improves the behaviour and in the y axis it depends on the sense of the displacement. If the engines are approached to the tip of the wing, the vibration will increase more quickly and the flutter will appear before. Otherwise, if the engines are approached to the body of the plane, the vibrations will take longer to appear.

8 Bibliography

Bisplinghoff, R. L., & Ashley, H. (2002). *Principles of Aeroelasticity.* Dover Publications.

Dowell, E. H. (1995). *A Modern Course in Aeroelasticity.* Dordrecht: Springer.

Fung, Y. C. (2002). *An Introduction to the Theory of Aeroelasticity.* Courier Corporation.

Hodges, D. H., & Pierce, G. A. (2002). *Introduction to Structural Dynamics and Aeroelasticity.* Cambridge: Cambridge University Press.

Modern Airliners. (n.d.). Retrieved May 14, 2019, from http://www.modernairliners.com/airbus-a320-introduction/airbus-a320-specs/.com

Murray, R. M. (2006). *Control and Dynamical Systems.*

Naidu, D. S. (2002). *Optimal Control Systems.* Pocatello: CRC Press.

Rodden, W. P. (2011). *Theoretical and Computational Aeroelasticity.* The Americas Group.

Theodorsen, T. (1979). *General Theory of Aerodynamic Instability and the Mechanism of Flutter.* Washington D.C.

Theodorsen, T., & Garrick, I. (1935). *Flutter Calculations in Three Degrees of Freedom.* Washington D.C.

Torres, J. A., Rendón, P. L., & Boullosa, R. R. (2010). *Complex modes of vibration due to small-scale damping in a guitar top-plate.* México D.F.: Journal of Applied Research and Technology.

Wright, J. R., & Cooper, J. E. (2015). *Introduction to Aircraft Aeroelasticity and Loads.* West Sussex: John Wiley and Sons, Ltd.

9 Appendices

9.1 Galerkin method

Orthogonality of the equations against the basis functions f

$$r(\omega,\theta) \perp f_i(y) \qquad i = 1,\dots,n$$

$$\int_0^l r(w,\theta) f_i(y)dy = 0 \qquad i = 1,\dots,n$$

9.2 Matlab code

9.2.1 wing_flutter

```
%
%  Aeroelastic model of swept slender wing
%
%  Bending-torsional structural model of the swept wing.
%
%  ()'     - d/dy, spatial derivative
%  d()/dt  - time derivative
%
%  State variables:
%   w(t,y)     - vertical displacement of the EA line of the wing [m]
%   theta(t,y) - twisting angle of the wing [rad].
%   y - spanwise coordinate [m]
%
%  Equations of motion
%
%  m*d^2w/dt^2 + S*d^2theta/dt^2  +  cw*dw/dt      +  kw*w      =  L
(1)
%  S*d^2w/dt^2 + I*d^2theta/dt^2  +  ct*dtheta/dt  +  kt*theta  =  My
+ u   (2)

%  L  - lift force per unit span  [N/m]
%  My - aerodynamic moment per unit span  [Nm/m]
%  u - control: aileron's moment

%  Bending equation (1)
%  Boundary conditions:   w(0) = w'(0) = 0;   w''(SemiSpan) =
w'''(SemiSpan) = 0;
%
  clear
  control = 0;     % 0 - LQR turned off, 1 - on

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  [ wingA320 ] = wing_wingA320data();
  [ engnA320 , wingA320 ] = wing_engnA320data( wingA320 );
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Air data
  Altitude = 8000.0;         % Altitude of flight [m]

  [ rho , a_sound , Temp ] = wing_Atmosphere( Altitude );

  % Computing flutter (or divergence)

  sprintf( 'Computing flutter (or divergence) critical velocity U_crit
by bisection method\n' );


  U_inf =  0.0               % Undisturbed velocity [m/s]
  d_U_inf = 50.0;

  bisect = 0;
  iter = 1;
  %for U_inf = 0.0 : 50.0 : 400.0 ,
  for ii = 1 :100 ,

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [ wingA320 ] = wing_AE_Model( wingA320 , U_inf , rho , a_sound );
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    nw = wingA320.n;
    A = wingA320.A;
    B = wingA320.B;
    size_A=size(A);
    size_B=size(B);
    n = size_A(1);
    m = size_B(2);

    Q = 1.0*eye(n);
    R = 1.0*eye(m);

    Ku = zeros( m , n );

    if( control == 1 )
      [Ku,Su] = lqr( A , B , Q , R );
      A=A-B*Ku;
    end

    [v,d] = eig(A); %v-> eigenvectors, d->eigenvalues


    %format long
    %diag(d)

    [ MaxRealEig , iMaxRealEig ] = max(real(diag(d)));
    ImagEig = imag(diag(d(iMaxRealEig,iMaxRealEig)));
    U_inf_t(iter) = U_inf;
    MaxRealEig_t(iter) = MaxRealEig;
    ImagEig_t(iter) = ImagEig;

    sprintf( ' U_inf = %15.11f [m/s]    MaxRealEig =%20.15f   ImagEig
=%18.12f     it= %4d\n' , U_inf , MaxRealEig , ImagEig , iter );

    if( bisect == 1 )
      if( MaxRealEig * MaxRealEig_1 <= 0.0 )
        U_inf_2 = U_inf;
        MaxRealEig_2 = MaxRealEig;
        %U_inf = 0.5*( U_inf_1 + U_inf_2 );
        U_inf = U_inf_1 - ( U_inf_2 - U_inf_1 ) * MaxRealEig_1 / (
MaxRealEig_2 - MaxRealEig_1 );
      elseif( MaxRealEig * MaxRealEig_2 <= 0.0 )
        U_inf_1 = U_inf;
        MaxRealEig_1 = MaxRealEig;
        %U_inf = 0.5*( U_inf_1 + U_inf_2 );
        U_inf = U_inf_1 - ( U_inf_2 - U_inf_1 ) * MaxRealEig_1 / (
MaxRealEig_2 - MaxRealEig_1 );
      end
    end


    if( iter > 3 && bisect ~= 1 )
      if( MaxRealEig_t(iter) * MaxRealEig_t(iter-1) <= 0.0 )
        U_crit = U_inf_t(iter-1) - ( U_inf_t(iter) - U_inf_t(iter-1) )
* MaxRealEig_t(iter-1) / ( MaxRealEig_t(iter) - MaxRealEig_t(iter-1)
);
        Mach = U_crit / a_sound;
        U_kmh = U_crit * 3.6;

        sprintf( '\nU_crit = %15.11f [m/s]    Mach =%5.3f   U_kmh
=%7.2f\n\n' , U_crit , Mach , U_kmh );

        vr_w = real( v(1:nw,iMaxRealEig) );
        vi_w = imag( v(1:nw,iMaxRealEig) );
        vr_t = real( v(nw+1:2*nw,iMaxRealEig) );
        vi_t = imag( v(nw+1:2*nw,iMaxRealEig) );

        lam_r = MaxRealEig;
        lam_i = ImagEig;
        x_axis=linspace(1,nw,nw);

        plot(x_axis, vr_w  , 'b--o' ,x_axis, vi_w , 'b--x', x_axis,
vr_t  , 'r--o'  ,x_axis, vi_t  , 'r--x' );
%           axis( [ 1  n ] );
          title( 'Critical flutter mode');
          legend('Real part w','Imaginary part w','Real part
theta','Imaginary part theta');
          xlabel('[-]');
          ylabel('Modes');
          grid on
          pause


        U_inf_1 = U_inf - d_U_inf;
        U_inf_2 = U_inf;
        MaxRealEig_1 = MaxRealEig_t(iter-1);
        MaxRealEig_2 = MaxRealEig_t(iter);

        U_inf = 0.5*( U_inf_1 + U_inf_2 );
        bisect = 1;

        %bisect = 2;

      end
    end

    %printf( 'U_inf = %15.11f [m/s]    MaxRealEig =%20.15f   ImagEig
=%18.12f\n' , U_inf , MaxRealEig , ImagEig );

    iter=iter+1;

    %[v,d] = eig( A - B*Ku );
    %MaxRealEigLoop = max(real(diag(d)))

%     fflush( stdout );

    if( bisect ~= 1 )
      U_inf = U_inf + d_U_inf;
    end

    if( bisect == 1 && abs( U_inf_2 - U_inf_1 ) < 1.e-10 )
      break
    end

  end

  [ U_inf_t , MaxRealEig_t , ImagEig_t ] = wing_sort_vec( U_inf_t ,
MaxRealEig_t , ImagEig_t , iter-1 );

    plot( U_inf_t , MaxRealEig_t , 'r--o' , U_inf_t , 0.001*ImagEig_t
, 'b--x' );
   legend('Real Part Eigenvalues','Imaginary Part Eigenvalues');
   title('Critical Velocity');
   xlabel('V [m/s]');
   ylabel('Eigenvalue');
   grid on;
   pause


  % U_inf  = U_inf + 1.0;
  % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % [ wingA320 ] = wing_AE_Model( wingA320 , U_inf , rho , a_sound );
  % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % A = wingA320.A;
  % B = wingA320.B;
  % nw = wingA320.n;

  % Simulation and plotting
  dt = 0.005;

  T_max = 100.0;
  it_max = T_max / dt;
  beta = 0.505;   % Euler integration parameter
  Ai = inv( eye(n)/dt - beta*( A - B*Ku ) );

  % Initial conditions
  x = zeros(n,1);

  %x(1*nw-1) = 1.0e-1;    % w
  %x(2*nw-1) = 1.0e-1;    % theta
  %x(3*nw-1) = 1.0e-3;    % dw/dt
  %x(3*nw-10) = 1.0e-3;   % dw/dt
  %x(4*nw-10) = 1.0e-3;   % dtheta/dt
  %x(4*nw-1) = 1.0e-2;    % dtheta/dt

  azimuth = 0.0;
  azimuth = 70.0;
  elev = 20.0;

  th_scale = 10000.0;
  w_scale  = th_scale * pi;

  %w_scale  = 0.0;
  %th_scale = 0.0;

  u = 0.0;
  t = 0;
  for it = 1 : it_max

    u0 = Ku * x;
    % T u' = -u + u0
    % u' = ( -u + u0 ) / T
    % ( un - u )/dt = ( -u + u0 ) / T
    % un = u + dt*( -u + u0 ) / T
    %T = 0.05;
    %u = u + dt*( -u + u0 )/T;
    u = u0;
    iflag = 0;
    u_max = 30.0 * pi/180.0;    % Max aileron deflection [rad]
    if( abs(u) > u_max )
      iflag = 1;
    end
    u = min( max( -u_max , u ) , u_max );

    %%%%%%%  Euler implicit integration %%%%%%%
    dx = Ai*( A*x - B*u );
    x = x + dx;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     [ wingA320 ] = wing_plotWing( u , x , w_scale , th_scale ,
wingA320 );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
     [ engnA320 ] = wing_plotEngine( x , w_scale , th_scale , engnA320
, nw );
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

    % Plotting

     x_min = min( min( wingA320.xw ) ) - 1.0;
     x_max = max( max( wingA320.xw ) ) + 1.0;
     y_min = min( min( wingA320.yw ) ) - 0.0;
     y_max = max( max( wingA320.yw ) ) + 1.0;

    % Engine
     if( engnA320.m_e > 1.0e-10 )
       surf( engnA320.x_ean , engnA320.y_ean , engnA320.z_ean );
       hold on
       surf( engnA320.x_ea1 , engnA320.y_ea1 , engnA320.z_ea1 );
       hold on
       surf( engnA320.x_ea2 , engnA320.y_ea2 , engnA320.z_ea2 );
      hold on
       surf( engnA320.x_ea3 , engnA320.y_ea3 , engnA320.z_ea3 );
       hold on
     end

    % Wing
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     surf( wingA320.xw , wingA320.yw , wingA320.zw );
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%     hold on

    % Aileron
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     hnd = surf( wingA320.xw_al , wingA320.yw_al , wingA320.zw_al );
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     set( hnd , 'FaceColor','red' );
%
%     hold on

    %axis( "square" );
    %colormap gray
    %colormap cool
    %colormap spring
    %colormap summer
    %colormap autumn
%     colormap winter
%     if( control == 1 )
%       colormap cool
%       if( iflag == 1  )
%         colormap gray
%       end
%     end
    %colormap ocean
    %shading interp
%     shading flat
%
%     hold off

    %set(hndl,'LineWidth',1)
```

```
%     axis( [ x_min  x_max  y_min  y_max  -3.0  5.0 ] ); %,
'linewidth' , 2 );

%     En = x'*x;
%     En = 0.5*x(2*nw+1:4*nw)'*wingA320.M*x(2*nw+1:4*nw) +
0.5*x(1:2*nw)'*wingA320.K*x(1:2*nw);
%     txt = sprintf( 'Airbus A320 wing       U inf = %7.3f [m/s]
Mach = %5.3f      Alt = %7.1f [m]       LQR control = %1d\n\n u =
%8.4f    En = %12.5e' , U_inf , U_inf/a_sound , Altitude , control ,
u*180.0/pi , En );
%     title( txt , "fontname", "Helvetica", "fontsize" , 18 );
%     view ([ azimuth  elev ] );
    %axis off

    %fflush( stdout );

    % Controlling plot by keyboard
%     key = kbhit(1);
%
%     % Turn control on/off
%     if( key == '1' )
%       control = 1;
%       [Ku,Su] = lqr( A , B , Q , R );
%     end
%     if( key == '0' )
%       control = 0;
%       Ku = 0*Ku;
%     end
%
%     % Exciting bending
%     if( key == '3' )
%       x(3*nw-1) = -1.e-3;
%     end
%
%     % Exciting torsion
%     if( key == '5' )
%       %x(4*n-1) = 1.e-3;
%       x(3*nw-floor(nw/2)) = 1.e-3;
%     end


    % Setting view
%     if( key == '4' )
%       azimuth = azimuth - 5.0;
%     end
%     if( key == '6' )
%       azimuth = azimuth + 5.0;
%     end
%
%     if( key == '8' )
%       elev = elev - 1.0;
%     end
%     if( key == '2' )
%       elev = elev + 1.0;
%     end
%
%     if( key == 'i' )
%       elev = 90.0;
%     end
%     if( key == 'o' )
%       elev = 0.0;
%     end
%     if( key == 'p' )
%       elev = -90.0;
%     end
%     if( key == 'q' )
%       azimuth = 0.0;
%     end
%     if( key == 'w' )
%       azimuth = 90.0;
%     end
%     if( key == 'e' )
%       azimuth = 180.0;
%     end

    %azimuth = azimuth + 0.50;
    if( azimuth >= 360 )
       azimuth = 0.0;
    end


    key = 0.0;

    t = t + dt;

  end


return;
```

9.2.2 wing_wingA320data

```
unction [ wingA320 ] = wing_wingA320data( n )


rad2deg = 180.0/pi;

A320_y = [ 594  595  495  270  270  498  494  533  534  594  594 ];
A320_x = [ 182  316  316  383  350  233  156  156  216  182  182 ];

A320_y = A320_y - 594;
A320_x = A320_x - 182;
A320_y = A320_y * -1;
A320_x = A320_x * -1;
A320_y = A320_y / 325;
A320_x = A320_x / 325;
A320_y = A320_y * 15.07;
A320_x = A320_x * 15.07;

A320_x = A320_x + 2.5;


 sweep = atan( 6.0 / 15.07 )*rad2deg;
% sweep = 30.0;

%hold off
```

```
%plot( A320_y , A320_x );
%axis( [ 0 15.0 ] );
%grid
%pause

A320_y_f = [ 340  264  264  283  278  258  240  235  254  16  16  237
340  340 ];
A320_z_f = [  39   53   53   32   15    4   15   30   56  88  92   70
63   39 ];

A320_y_f = A320_y_f - 340;
A320_z_f = A320_z_f - 51;
A320_y_f = A320_y_f * -1;
A320_y_f = A320_y_f / 324;
A320_z_f = A320_z_f / 324;
A320_y_f = A320_y_f * 15.07;
A320_z_f = A320_z_f * 15.07;

 dihedral = atan( 1.82 / 15.07 )*rad2deg; %atan = inverse tangent in
radians
% dihedral = -60;
% dihedral = atan( 1.82 / 15.07 )*rad2deg

%plot( A320_y_f , A320_z_f );
%axis( [ 0 15.1 ] );
%grid
%pause

x0_w =  2.50;

x1_w = -3.72;
y1_w =  4.60;

x2_w = -5.30;
x3_w = -6.80;
y2_w =  15.0;

SemiSpan = y2_w;

%%%%%%%
n = 50       %  Number of wing sections
%%%%%%%

dy = SemiSpan/(n-1);          %  Section width [m]

i_1 = floor( y1_w / dy );
der_le = ( x2_w - x0_w ) / y2_w;
der_te = ( x3_w - x1_w ) / ( y2_w - y1_w );
%pause
for  i = 1 : i_1
  eta = (i-1)*dy/SemiSpan;
  y(i) = (i-1)*dy;
  x_le(i) = der_le * y(i) + x0_w;
  x_te(i) = x1_w;
  %b(i) = b0 - 1.973*eta;
  chord(i) = x_le(i) - x_te(i);
end

for  i = i_1 + 1 : n
```

```
  eta = (i-1)*dy/SemiSpan;
  y(i) = (i-1)*dy;
  x_le(i) = der_le * y(i) + x0_w;
  x_te(i) = min( x1_w ,  x1_w + der_te * ( y(i) - y1_w ) );
  %b(i) = b0 - 1.973*eta;
  chord(i) = x_le(i) - x_te(i);
end
%y'
%x_le'
%x_te'
%pause

%hold off
%plot( y , x_le , 'r-o' , y , x_te , 'r-o' );
%axis( [ 0 15.0 ] );
%grid
%pause

AileronSpan = 3.0;                                 % Aileron's span
[m]
n_alrn = floor( n * AileronSpan / SemiSpan )       % Number of
aileron's section
%pause
c_alrn = 0.3;

% Structural data

mass = 0.0;
for  i = 1 : n
  eta = (i-1)*dy/SemiSpan;
  psi = chord(i)/chord(1);
  a(i)     = -0.15;                          % Elastic center (EC)
position w.r.t. chord center, + backward  (dimensionless)
  %xcg(i)  = 0.135 - 0.25*eta;                % Position of gravity
center w.r.t. elastic center (EC), + backward [m]
  xcg(i)   = 0.060 * psi;                    % Position of gravity
center w.r.t. elastic center (EC), + backward [m]

  m(i,i)   = 100.0 * psi;                    % Mass distributions of
the wing (constant) [kg/m]
  I(i,i)   = 200.0 * psi^2;                  % Moment of inertia of the
wing    [kg*m^2/m]
  S(i,i)   = -m(i,i) * xcg(i);               % Static moment [kg*m/m]

  %EI(i)    = 7.0e+06 * ( 1.0 - 0.05*eta );   % Bending stiffness
distributions of the wing (constant)  [N*m^2];
  %GJ(i)    = 1.0e+07 * ( 1.0 - 0.05*eta );   % Torsional stiffness
[N*m^2/rad]
  EI(i)    = 7.0e+06 * psi;                  % Bending stiffness
distributions of the wing (constant)  [N*m^2];
  GJ(i)    = 9.5e+06  * psi;                  % Torsional stiffness
[N*m^2/rad]

  cw(i,i) = 0.0;                             % Bending structural
damping
  ct(i,i) = 0.0;                             % Torsion structural
damping

  mass     = mass + m(i,i);
end
```

```
mass
%pause

wingA320.n        = n;
wingA320.x_le     = x_le;
wingA320.x_te     = x_te;
wingA320.SemiSpan = SemiSpan;
wingA320.y        = y;
wingA320.dy       = dy;
wingA320.chord    = chord;
wingA320.sweep    = sweep;
wingA320.dihedral = dihedral;
wingA320.a        = a;
wingA320.m        = m;
wingA320.S        = S;
wingA320.I        = I;
wingA320.EI       = EI;
wingA320.GJ       = GJ;
wingA320.cw       = cw;
wingA320.ct       = ct;
wingA320.n_alrn   = n_alrn;
wingA320.c_alrn   = c_alrn;
```

9.2.3 wing_eig_sort

```
function  [ eig , vec ] = wing_eig_sort( eig , vec )
  size_eig=size(eig);
  n = size_eig(1);
  do
    ip = 0;
    for  i = 1 : n-1
      if( eig(i) > eig(i+1) )
        eigt = eig(i+1);
        eig(i+1) = eig(i);
        eig(i) = eigt;
        vect = vec(:,i+1);
        vec(:,i+1) = vec(:,i);
        vec(:,i) = vect;
        ip = 1;
      end
    end
  until( ip == 0 )
```

9.2.4 wing_sort_vec

```
function  [ x , a , b ] = wing_sort_vec( x , a , b , n )

%   do
    ip = 0;
    for  i = 1 : n-1
      if( x(i) > x(i+1) )
        xt = x(i+1);
        x(i+1) = x(i);
        x(i) = xt;
        at = a(:,i+1);
        a(:,i+1) = a(:,i);
        a(:,i) = at;
        bt = b(:,i+1);
        b(:,i+1) = b(:,i);
        b(:,i) = bt;
        ip = 1;
      end
    end
%   until( ip == 0 )
```

9.2.5 lqr

```
function [k,s]=lqr(a,b,q,r,nn)

%LQR    Linear quadratic regulator design for continuous-time systems.
%   [K,S] = LQR(A,B,Q,R)  calculates the optimal feedback gain matrix
K
%   such that the feedback law  u = -Kx  minimizes the cost function:
%
%       J = Integral {x'Qx + u'Ru} dt
%
%   subject to the constraint equation:
%       .
%       x = Ax + Bu
%
%   Also returned is S, the steady-state solution to the associated
%   algebraic Riccati equation:
%                 -1
%       0 = SA + A'S - SBR  B'S + Q
%
%   [K,S] = LQR(A,B,Q,R,N) includes the cross-term N that relates
%   u to x in the cost function.

%   J.N. Little 4-21-85
%   Revised 8-27-86 JNL
%   Copyright (c) 1985, 1986 by the MathWorks, Inc.

%error(nargchk(4,5,nargin));
%error(abcdchk(a,b));

[m,n] = size(a);
[mb,nb] = size(b);
[mq,nq] = size(q);
if (m ~= mq) || (n ~= nq)
    error('A and Q must be the same size')
end
[mr,nr] = size(r);
if (mr ~= nr) || (nb ~= mr)
    error('B and R must be consistent')
end

if nargin == 5
    [mn,nnn] = size(nn);
    if (mn ~= m) || (nnn ~= nr)
        error('N must be consistent with Q and R')
    end
    % Add cross term
    q = q - nn/r*nn';
    a = a - b/r*nn';
else
    nn = zeros(m,nb);
```

```
end

% Check if q is positive semi-definite and symmetric
if any(eig(q) < 0) || (norm(q'-q,1)/norm(q,1) > eps)
    error('Q must be symmetric and positive semi-definite')
end
% Check if r is positive definite and symmetric
if any(eig(r) <= 0) || (norm(r'-r,1)/norm(r,1) > eps)
    error('R must be symmetric and positive definite')
end

% Start eigenvector decomposition by finding eigenvectors of
Hamiltonian:
[v,d] = eig([a b/r*b';q, -a']);
d = diag(d);
[d,index] = sort(real(d));    % sort on real part of eigenvalues

%if (~( (d(n)<0) && (d(n+1)>0) ))
if (~( (d(n)<1.e-15) && (d(n+1)>-1.e-15) ))
   printf('Can''t order eigenvalues, (A,B) may be uncontrollable.
Checking rank C = [B A*B ... A^(n-1)*B ]\n')
   C = zeros(m,n*nb);
   c = b;
   C(:,1:nb) = c;
   for i = 1 : n-1 ,
      c = a*c;
      C(:,i*nb+1:i*nb+nb) = c;
   end
   %C
   rank_C = rank(C);
   if( rank_C < n )
     rank_C
     error('rank_C < n , (A,B) are uncontrollable.')
   else
     error('rank_C = n (OK), but there is something wrong with
ordering - check it out!')
   end
end

chi = v(1:n,index(1:n));      % select vectors with negative
eigenvalues
lambda = v((n+1):(2*n),index(1:n));
s = -real(lambda/chi);
k = r\(nn'+b'*s);

end
```

9.2.6 wing_AE_Model

```
function [ wing ] = wing_AE_Model( wing , U_inf , rho , a_sound )

  kw = wing_BendStifMatr( wing.EI , wing.dy , wing.n , wing.m );

  kt = wing_TorsnStifMatr( wing.GJ , wing.dy , wing.n , wing.I );
  [ Ka , Ca , M_alrn ] = wing_AerodynamicModel( U_inf , rho , a_sound
, wing );

  %
  %       |  m    S  |
  %   M = |          |
  %       |  S    I  |
  %
  %       |  cw   0  |
  %   C = |          |
  %       |  0   ct  |
  %
  %       |  kw   0  |
  %   K = |          |
  %       |  0    kt |
  %
  %        | L_dwdt   L_dthetadt  |
  %   Ca = |                      |
  %        | My_dwdt  My_dthetadt |
  %
  %        | L_dwdy   L_theta     |
  %   Ka = |                      |
  %        | My_dwdy  My_theta    |
  %
  %
  %        |     0    |
  %   Bu = |          |
  %        |  M_alrn  |
  %
  %   x = [ w , theta ]
  %
  %
  %   M*d2xdt2 + C*dxdt + Kx = Ca*dxdt + Ka*x + M_alrn*u
  %
  %
  %       |    0    1   |
  %   A = |             |
  %       |  -M\K -M\C  |
  %
  %       |      0      |
  %   B = |             |
  %       |  M\M_alrn   |
  %
  %
  %

  n = wing.n;

  M = [  wing.m  -wing.S  ;
        -wing.S   wing.I  ];

  C = [  wing.cw   zeros(n) ;
         zeros(n)  wing.ct  ];

  K = [    kw      zeros(n) ;
         zeros(n)     kt    ];

  Bu = [ zeros(n,1) ;
          M_alrn'   ]; % Aileron on the n_alrn*dy sections of the wing
  A = [  zeros(2*n)   eye(2*n)  ;
        -M\(K-Ka)    -M\(C-Ca)  ];

  B = [ zeros(2*n,1) ; M\Bu ];


  wing.M = M;
  wing.C = C;
  wing.K = K;
  wing.Bu = Bu;

  wing.A = A;
  wing.B = B;
```

9.2.7 wing_AerodynamicModel

```
function [ Ka , Ca , M_alrn ] = wing_AerodynamicModel( U_inf , rho ,
a_sound , wing )
%
% Aerodynamic model: strip, quasi-steady

  a      = wing.a;
  chord  = wing.chord;
  sweep  = wing.sweep;
  n      = wing.n;
  n_alrn = wing.n_alrn;
  c_alrn = wing.c_alrn;

  % AOA - angle of attack [rad]
  % alpha = ( theta - dw/dy * tan(Lambda) + b/U_inf*( 0.5-a
)*dtheta/dt - (dw/dt)/U_inf ) * cos(Lambda)

  % L  = 0.5 * rho * U_inf^2 * c * dCzdAlpha * alpha
  % My = 0.5 * rho * U_inf^2 * c * e * dCzdAlpha * alpha +
c*pi/(8*U_inf)*dtheta/dt

  % L  = 0.5*rho*U_inf^2*c*dCzdAlpha*( theta*cos(Lambda) -
dw/dy*sin(Lambda) + b/U_inf*( 0.5-a )*dtheta/dt*cos(Lambda) -
(dw/dt)/U_inf*cos(Lambda) )
  % My = 0.5*rho*U_inf^2*c*e*dCzdAlpha*( theta*cos(Lambda) -
dw/dy*sin(Lambda) + b/U_inf*( 0.5-a )*dtheta/dt*cos(Lambda) -
(dw/dt)/U_inf*cos(Lambda) )

  % L_theta =  0.5*rho*U_inf^2*c*dCzdAlpha*cos(Lambda) * theta
  % L_dwdt  = -0.5*rho*U_inf*c*dCzdAlpha*cos(Lambda) * dw/dt

  % My_theta =  0.5*rho*U_inf^2*c*e*dCzdAlpha*cos(Lambda) * theta
  % My_dwdt  = -0.5*rho*U_inf*c*e*dCzdAlpha*cos(Lambda) * dw/dt

  % L_dwdy  = -0.5*rho*U_inf^2*c*dCzdAlpha*sin(Lambda) * dw/dy
  % My_dwdy = -0.5*rho*U_inf^2*c*e*dCzdAlpha*sin(Lambda) * dw/dy

  %dy = 15.0/(n-1);

  Lambda = sweep * pi/180.0;
  dCzdAlpha_0 = 4.0;          % Lift slope           [1/rad]
  dCzddelta_alrn_0 = 2.0;     % Aileron derivative   [1/rad]

  Mach = U_inf / a_sound;
  Prandtl = min( 4.0 , 1.0/sqrt( abs( 1.0 - Mach^2 ) ) );
  dCzdAlpha = dCzdAlpha_0 * Prandtl;
  dCzddelta_alrn = dCzddelta_alrn_0 * Prandtl;

  %L_total = 0.0;

  %for  Mach = 0.0 : 0.1 : 2.0
  %  Mach
  %  tanh( Mach^3 )
  %end
  %pause

  for  i = 1 : n
    e(i) = 0.25;          % Aerodynamic center (AC) position w.r.t.
elastic center (EC), + forward  (dimensionless)
    %e(i) = 0.25 * ( 1.0 - tanh( Mach^2.5 ) );

    q = 0.5 * rho * U_inf * chord(i) * dCzdAlpha * cos(Lambda);
    L_theta(i,i)      =  q * U_inf;
    My_theta(i,i)     =  q * U_inf * e(i);
    L_dwdt(i,i)       = -q;
    My_dwdt(i,i)      = -q * e(i);
    L_dthetadt(i,i)   =  1*q * 0.5*chord(i)*( 0.5 - a(i) );
    My_dthetadt(i,i)  =  1*q * ( -e(i)*0.5*chord(i)*( 0.5 - a(i) ) -
chord(i)*pi/8.0 );

    %L_total = L_total + L_theta(i,i) * dy;

    M_alrn(i) = 0.0;
    if( i >= n-n_alrn )
      chord_alrn = c_alrn*chord(i);
      M_alrn(i) = -0.5 * rho * U_inf^2 * chord_alrn * (0.25) *
dCzddelta_alrn * cos(Lambda);
      %M_alrn(i) = 0.0001 * M_alrn(i);
      %My_theta(i,i) =  My_theta(i,i) + M_alrn(i);
    end
  end

  %L_total

  for  i = 2 : n
    L_dwdy(i,i)     = -0.5*rho*U_inf^2*chord(i)*dCzdAlpha*sin(Lambda);
    L_dwdy(i,i-1)   = -L_dwdy(i,i);
    My_dwdy(i,i)    = -
0.5*rho*U_inf^2*chord(i)*e(i)*dCzdAlpha*sin(Lambda);
    My_dwdy(i,i-1)  = -My_dwdy(i,i);
  end
  L_dwdy(1,1)   = L_dwdy(2,2);
  My_dwdy(1,1)  = My_dwdy(2,2);

  Ca = [ L_dwdt   L_dthetadt  ;
         My_dwdt  My_dthetadt ];

  Ka = [ L_dwdy   L_theta     ;
         My_dwdy  My_theta    ];
```

9.2.8 wing_BenStifMatr

```
function kw = wing_BendStifMatr( EI , dy , n , m )

  %  FD (Finite difference) discretization of bending stiffness
operator  d^4(w)/dy^4
  dy4 = dy^4;
  for  i = 2 : n-1
    if i >= 3 ,
      kw(i,i-2) =  EI(i-1) / dy4;
    end
    if i >= 2 ,
      kw(i,i-1) = -2.0*( EI(i-1) + EI(i) ) / dy4;
    end
    kw(i,i) = ( EI(i-1) + 4.0*EI(i) + EI(i+1) ) / dy4;
    if i <= n-1 ,
      kw(i,i+1) = -2.0*( EI(i) + EI(i+1) ) / dy4;
    end
    if i <= n-2 ,
      kw(i,i+2) = EI(i+1) / dy4;
    end
  end

  % w(0) = 0 , w'(0) = 0
  kw(1,1) = 6.0 * EI(1) / dy4;
  kw(2,2) = kw(2,2) + EI(1) / dy4;

  % w''(SemiSpan) = 0 , w'''(SemiSpan) = 0
  kw(n-1,n-1) = ( EI(n-2) + 4.0*EI(n-1) ) / dy4;
  kw(n-1,n)   = -2.0*EI(n-1) / dy4;

  kw(n,n-2)   =  ( EI(n-1) + EI(n) ) / dy4;
  kw(n,n-1)   = -( 2.0*EI(n-1) + 2.0*EI(n) ) / dy4;
  kw(n,n)     =  ( EI(n-1) + EI(n) ) / dy4;

  %format bank
  %kw

  %[vw,d] = eig(kw,m);
  %dw = diag(d);
  %[dw,vw] = wing_eig_sort( dw , vw );

  %for  i = 1 : n
  %  printf( 'i = %2d   eig = %25.10f\n' , i , dw(i) )
  %end

  %% Plot bending mode shapes
  %hold off
  %iw = 0;
  %plot( vw(:,iw+1) , 'r-' , vw(:,iw+2) , 'g-' , vw(:,iw+3) , 'b-' ,
vw(:,iw+4) , 'm-' );
  %axis( [ 1  n  -1  1 ] );
  %title( 'Bending mode shapes 1-4',  "fontname", "Helvetica",
"fontsize" , 18  );
  %grid;
  %omega = sqrt(sort(diag(d)));
  %omega(1);
  %Tw = 2*pi/omega(1);
  %Freq_w = 1/Tw;
  %%printf( 'Bending: Omega= %8.4f [rad/s]   f =%8.4f [Hz]\n' ,
omega(1) , Freq_w );
```

9.2.9 wing_TorsnStifMatr

```
function  kt = wing_TorsnStifMatr( GJ , dy , n , I );

  dy2 = dy^2;
  for  i = 2 : n
    kt(i,i) = 0.0;
    if i > 1 ,
      kt(i,i-1) = -GJ(i-1) / dy2;
    end
    kt(i,i) = ( GJ(i-1) + GJ(i) ) / dy2;
    if i < n ,
      kt(i,i+1) = -GJ(i) / dy2;
    end
  end
  kt(1,1) = 2*GJ(1) / dy2;       % theta(0) = 0

  kt(n,n-1) = -GJ(n-1) / dy2;    % theta'(SemiSpan) = 0
  kt(n,n) = GJ(n-1) / dy2;
  %kt
  %pause

  %   Torsional equation
  %   Boundary conditions:   theta(0) = 0;   theta'(SemiSpan) = 0;
  %
  %   Finite Difference discretization of torsional stiffness operator
d^2(theta) / dy^2

  %format short

  %[vt,d] = eig( kt , I );
  %dte = diag(d);
  %[dte,vt] = wing_eig_sort( dte , vt );

  %for  i = 1 : n
  %  printf( 'i = %2d   eig = %25.10f\n' , i , dte(i) )
  %end

  % Plot torsional mode shapes
  %hold off
  %iw = 0;
  %plot( vt(:,iw+1) , 'r-' , vt(:,iw+2) , 'g-' , vt(:,iw+3) , 'b-' ,
vt(:,iw+4) , 'm-' );
  %axis( [ 1  n  -1  1 ] );
  %title( 'Torsional mode shapes 1-4',  "fontname", "Helvetica",
"fontsize" , 18  );
  %grid;
  %pause

  %omega = sqrt(sort(diag(d)));
  %omega(1);
  %Tt = 2*pi/omega(1);
  %Freq_t = 1/Tt;
  %printf( 'Torsion: Omega= %8.4f [rad/s]   f =%8.4f [Hz]\n' ,
omega(1) , Freq_t );
```

9.2.10wing_Atmosphere

```
function  [ rho , a_sound , Temp ] = wing_Atmosphere( Altitude )

rho_0 = 1.225;
rho = rho_0 * ( 1.0 - abs( Altitude)/44300.0 )^4.256;
Temp = max( 273.0 - 56.5 , 288.0 - 0.0065 * Altitude );
a_sound = 20.05 * sqrt( Temp );
```

9.2.11wing_plotWing

```
function [ wing ] = wing_plotWing( u , x , w_scale , th_scale , wing )


n        = wing.n;
x_le     = wing.x_le;
x_te     = wing.x_te;
y        = wing.y;
chord    = wing.chord;
a        = wing.a;
dihedral = wing.dihedral;
sweep    = wing.sweep;
n_alrn   = wing.n_alrn;
c_alrn   = wing.c_alrn;


%Am = 5.e-3;
%lam_i = abs( lam_i );
%fi = 0.0;
%arg = lam_i*t + fi;

Dihedral = dihedral * pi/180.0;
Lambda = sweep * pi/180.0;

for i = 1 : n-1
  %w1  = Am*exp( lam_r*t ) * ( vr_w(1) * sin( arg ) + vi_w(1) * cos(
arg ) );
  %th1 = Am*exp( lam_r*t ) * ( vr_t(1) * sin( arg ) + vi_t(1) * cos(
arg ) );
  %w2  = Am*exp( lam_r*t ) * ( vr_w(i) * sin( arg ) + vi_w(i) * cos(
arg ) );
  %th2 = Am*exp( lam_r*t ) * ( vr_t(i) * sin( arg ) + vi_t(i) * cos(
arg ) );
  w1  = x(i);
  th1 = x(n+i);
  w1  = w1  * w_scale;
  th1 = th1 * th_scale;
  w2  = x(i+1);
  th2 = x(n+i+1);
  w2  = w2 * w_scale;
  th2 = th2* th_scale;
  z1 = y(i) * tan( Dihedral );
  z2 = y(i+1) * tan( Dihedral );

  xw(1,i) = x_le(i)   *  cos( th1 );
  xw(2,i) = x_le(i+1) *  cos( th2 );
  xw(3,i) = x_te(i+1) *  cos( th2 );
  xw(4,i) = x_te(i)   *  cos( th1 );
  xw(5,i) = xw(1,i);

  yw(1,i) = y(i);
  yw(2,i) = y(i+1);
  yw(3,i) = y(i+1);
  yw(4,i) = y(i);
  yw(5,i) = yw(1,i);

  x0 = y(i)   * tan( Lambda );
  x1 = y(i+1) * tan( Lambda );

  zw(1,i) = z1 + w1 + sin( th1 ) * ( xw(1,i) - x0 );
  zw(2,i) = z2 + w2 + sin( th2 ) * ( xw(2,i) - x1 );
  zw(3,i) = z2 + w2 + sin( th2 ) * ( xw(3,i) - x1 );
  zw(4,i) = z1 + w1 + sin( th1 ) * ( xw(4,i) - x0 );
  zw(5,i) = zw(1,i);

  delta = u;
  delta = delta * th_scale * 0.1;

  %if( 0 )
  if( i >= n-n_alrn )

    if( i > n-n_alrn-2 )
      xw(3,i) = x_te(i+1) + c_alrn*chord(i);
      xw(4,i) = x_te(i)   + 0.5;
      zw(3,i) = z2 + w2 + sin( th2 ) * ( xw(3,i) - x1 );
      zw(4,i) = z1 + w1 + sin( th1 ) * ( xw(4,i) - x0 );
    end

    ca = 1.0;  % Aileron chord

    i1 = i - ( n-n_alrn ) + 1;
    xw_al(1,i1) = xw(3,i);
    xw_al(2,i1) = xw(4,i);
    xw_al(3,i1) = x_te(i+1);
    xw_al(4,i1) = x_te(i);
    xw_al(5,i1) = xw_al(1,i1);

    yw_al(1,i1) = y(i);
    yw_al(2,i1) = y(i+1);
    yw_al(3,i1) = y(i+1);
    yw_al(4,i1) = y(i);
    yw_al(5,i1) = yw_al(1,i1);

    zw_al(1,i1) = z1 + w1 + sin( th1 ) * ( xw(3,i) - x1 );
    zw_al(2,i1) = z2 + w2 + sin( th2 ) * ( xw(4,i) - x0 );
    zw_al(3,i1) = z2 + w2 + sin( th2 ) * ( x_te(i+1) - x1 ) +
ca*sin(delta);
    zw_al(4,i1) = z1 + w1 + sin( th1 ) * ( x_te(i)   - x0 ) +
ca*sin(delta);
    zw_al(5,i1) = zw_al(1,i1);
  end
  %end

end

wing.xw     = xw;
wing.yw     = yw;
wing.zw     = zw;
wing.xw_al  = xw_al;
wing.yw_al  = yw_al;
wing.zw_al  = zw_al;
```

9.2.12wing_plotEngine

```
function  [ engn ] = wing_plotEngine(  x , w_scale , th_scale , engn ,
nw )

w_e  = w_scale*x( engn.j_e );
th_e = th_scale*x( nw + engn.j_e );

n_e = engn.n_e;

xp_e1 = engn.xp_e1;
yp_e1 = engn.yp_e1;
zp_e1 = engn.zp_e1;

for i = 1 : n_e ,
  xp_ea1(i,1) = xp_e1(i,1)*cos( th_e ) - zp_e1(i,1)*sin( th_e );
  xp_ea1(i,2) = xp_e1(i,2)*cos( th_e ) - zp_e1(i,2)*sin( th_e );
  xp_ea1(i,3) = xp_e1(i,3)*cos( th_e ) - zp_e1(i,3)*sin( th_e );
  xp_ea1(i,4) = xp_e1(i,4)*cos( th_e ) - zp_e1(i,4)*sin( th_e );

  yp_ea1(i,1) = yp_e1(i,1);
  yp_ea1(i,2) = yp_e1(i,2);
  yp_ea1(i,3) = yp_e1(i,3);
  yp_ea1(i,4) = yp_e1(i,4);

  zp_ea1(i,1) = w_e + xp_e1(i,1)*sin( th_e ) + zp_e1(i,1)*cos( th_e );
  zp_ea1(i,2) = w_e + xp_e1(i,2)*sin( th_e ) + zp_e1(i,2)*cos( th_e );
  zp_ea1(i,3) = w_e + xp_e1(i,3)*sin( th_e ) + zp_e1(i,3)*cos( th_e );
  zp_ea1(i,4) = w_e + xp_e1(i,4)*sin( th_e ) + zp_e1(i,4)*cos( th_e );
end

engn.x_ea1 = xp_ea1;
engn.y_ea1 = yp_ea1;
engn.z_ea1 = zp_ea1;


xp_e2 = engn.xp_e2;
yp_e2 = engn.yp_e2;
zp_e2 = engn.zp_e2;

 for i = 1 : n_e ,
   xp_ea2(i,1) = xp_e2(i,1)*cos( th_e ) - zp_e2(i,1)*sin( th_e );
   xp_ea2(i,2) = xp_e2(i,2)*cos( th_e ) - zp_e2(i,2)*sin( th_e );
   xp_ea2(i,3) = xp_e2(i,3)*cos( th_e ) - zp_e2(i,3)*sin( th_e );
   xp_ea2(i,4) = xp_e2(i,4)*cos( th_e ) - zp_e2(i,4)*sin( th_e );

   yp_ea2(i,1) = yp_e2(i,1);
   yp_ea2(i,2) = yp_e2(i,2);
   yp_ea2(i,3) = yp_e2(i,3);
   yp_ea2(i,4) = yp_e2(i,4);

   zp_ea2(i,1) = w_e + xp_e2(i,1)*sin( th_e ) + zp_e2(i,1)*cos( th_e
);
   zp_ea2(i,2) = w_e + xp_e2(i,2)*sin( th_e ) + zp_e2(i,2)*cos( th_e
);
   zp_ea2(i,3) = w_e + xp_e2(i,3)*sin( th_e ) + zp_e2(i,3)*cos( th_e
);
   zp_ea2(i,4) = w_e + xp_e2(i,4)*sin( th_e ) + zp_e2(i,4)*cos( th_e
);
 end

engn.x_ea2 = xp_ea2;
engn.y_ea2 = yp_ea2;
engn.z_ea2 = zp_ea2;

xp_e3 = engn.xp_e3;
yp_e3 = engn.yp_e3;
zp_e3 = engn.zp_e3;

for i = 1 : n_e ,
  xp_ea3(i,1) = xp_e3(i,1)*cos( th_e ) - zp_e3(i,1)*sin( th_e );
  xp_ea3(i,2) = xp_e3(i,2)*cos( th_e ) - zp_e3(i,2)*sin( th_e );
  xp_ea3(i,3) = xp_e3(i,3)*cos( th_e ) - zp_e3(i,3)*sin( th_e );
  xp_ea3(i,4) = xp_e3(i,4)*cos( th_e ) - zp_e3(i,4)*sin( th_e );

  yp_ea3(i,1) = yp_e3(i,1);
  yp_ea3(i,2) = yp_e3(i,2);
  yp_ea3(i,3) = yp_e3(i,3);
  yp_ea3(i,4) = yp_e3(i,4);

  zp_ea3(i,1) = w_e + xp_e3(i,1)*sin( th_e ) + zp_e3(i,1)*cos( th_e );
  zp_ea3(i,2) = w_e + xp_e3(i,2)*sin( th_e ) + zp_e3(i,2)*cos( th_e );
  zp_ea3(i,3) = w_e + xp_e3(i,3)*sin( th_e ) + zp_e3(i,3)*cos( th_e );
  zp_ea3(i,4) = w_e + xp_e3(i,4)*sin( th_e ) + zp_e3(i,4)*cos( th_e );
end

engn.x_ea3 = xp_ea3;
engn.y_ea3 = yp_ea3;
engn.z_ea3 = zp_ea3;

xp_epn = engn.xp_epn;
yp_epn = engn.yp_epn;
zp_epn = engn.zp_epn;

n_pn = engn.n_pn;

for i = 1 : n_pn ,
  xp_eapn(i,1) = xp_epn(i,1)*cos( th_e ) - zp_epn(i,1)*sin( th_e );
  xp_eapn(i,2) = xp_epn(i,2)*cos( th_e ) - zp_epn(i,2)*sin( th_e );
  xp_eapn(i,3) = xp_epn(i,3)*cos( th_e ) - zp_epn(i,3)*sin( th_e );
  xp_eapn(i,4) = xp_epn(i,4)*cos( th_e ) - zp_epn(i,4)*sin( th_e );

  yp_eapn(i,1) = yp_epn(i,1);
  yp_eapn(i,2) = yp_epn(i,2);
  yp_eapn(i,3) = yp_epn(i,3);
  yp_eapn(i,4) = yp_epn(i,4);

  zp_eapn(i,1) = w_e + xp_epn(i,1)*sin( th_e ) + zp_epn(i,1)*cos( th_e
);
  zp_eapn(i,2) = w_e + xp_epn(i,2)*sin( th_e ) + zp_epn(i,2)*cos( th_e
);
  zp_eapn(i,3) = w_e + xp_epn(i,3)*sin( th_e ) + zp_epn(i,3)*cos( th_e
);
  zp_eapn(i,4) = w_e + xp_epn(i,4)*sin( th_e ) + zp_epn(i,4)*cos( th_e
);
end

engn.x_ean = xp_eapn;
engn.y_ean = yp_eapn;
engn.z_ean = zp_eapn;
```

9.2.13wing_engnA320data

```
function  [ engnA320 , wing ] = wing_engnA320data( wing )

n        = wing.n;
SemiSpan = wing.SemiSpan;
m        = wing.m;
S        = wing.S;
I        = wing.I;


A320_x_eng = [  45  45  77  120  120  160  160  119  118   80   45
45   45  ];
A320_z_eng = [ 102  82  79   84   90   96  108  115  119  129  122
103  102  ];

A320_x_eng = A320_x_eng + -80;
A320_z_eng = A320_z_eng + -102;
A320_x_eng = A320_x_eng / 23;
A320_z_eng = A320_z_eng / 23;
A320_x_eng = A320_x_eng * 1.1;
A320_z_eng = A320_z_eng * 1.1;

%hold off
%plot( A320_x_eng , A320_z_eng );
%grid
%pause

L_e  =  5.50;

x_e1 =  1.70;
R_e1 =  0.95;

x_e0 =  0.0;
R_e0 =  1.10;

x_e2 = -1.50;
R_e2 =  0.90;
```

```
x_e3 = -2.80;  %  -3.8
R_e3 =  0.40;

% Original
 x0_e =  0.0;
 y0_e =  3.35;
 z0_e = -1.0;

%x0_e = -1.0;
%y0_e =  7.35;
%z0_e = -0.5;

% x0_e = 0.0;
% y0_e = 3.35;
% z0_e = -3.0;




j_e = floor( n * y0_e / SemiSpan )
j_e*(SemiSpan/(n-1))
y0_e = j_e*(SemiSpan/(n-1));
%pause


n_e = 20;

dalf = 2*pi/(n_e-1);
alf1 = 0.50*pi;
alf2 = alf1 - dalf;
for i = 1 : n_e ,
  xp_e1(i,1) = x0_e + x_e1;
  xp_e1(i,2) = x0_e + x_e0;
  xp_e1(i,3) = x0_e + x_e0;
  xp_e1(i,4) = x0_e + x_e1;
  xp_e1(i,5) = xp_e1(i,1);

  yp_e1(i,1) = y0_e + R_e1 * cos( alf1 );
  yp_e1(i,2) = y0_e + R_e0 * cos( alf1 );
  yp_e1(i,3) = y0_e + R_e0 * cos( alf2 );
  yp_e1(i,4) = y0_e + R_e1 * cos( alf2 );
  yp_e1(i,5) = yp_e1(i,1);

  zp_e1(i,1) = z0_e + R_e1 * sin( alf1 );
  zp_e1(i,2) = z0_e + R_e0 * sin( alf1 );
  zp_e1(i,3) = z0_e + R_e0 * sin( alf2 );
  zp_e1(i,4) = z0_e + R_e1 * sin( alf2 );
  zp_e1(i,5) = zp_e1(i,1);

  alf1 = alf2;
  alf2 = alf1 - dalf;
end

dalf = 2*pi/(n_e-1);
alf1 = 0.50*pi;
alf2 = alf1 - dalf;
for i = 1 : n_e ,
  xp_e2(i,1) = x0_e + x_e0;
  xp_e2(i,2) = x0_e + x_e2;
  xp_e2(i,3) = x0_e + x_e2;
  xp_e2(i,4) = x0_e + x_e0;
  xp_e2(i,5) = xp_e2(i,1);

  yp_e2(i,1) = y0_e + R_e0 * cos( alf1 );
  yp_e2(i,2) = y0_e + R_e2 * cos( alf1 );
  yp_e2(i,3) = y0_e + R_e2 * cos( alf2 );
  yp_e2(i,4) = y0_e + R_e0 * cos( alf2 );
  yp_e2(i,5) = yp_e2(i,1);

  zp_e2(i,1) = z0_e + R_e0 * sin( alf1 );
  zp_e2(i,2) = z0_e + R_e2 * sin( alf1 );
  zp_e2(i,3) = z0_e + R_e2 * sin( alf2 );
  zp_e2(i,4) = z0_e + R_e0 * sin( alf2 );
  zp_e2(i,5) = zp_e2(i,1);

  alf1 = alf2;
  alf2 = alf1 - dalf;
end


alf1 = 0.50*pi;
alf2 = alf1 - dalf;
for i = 1 : n_e ,
  xp_e3(i,1) = x0_e + x_e2 + 1.0;
  xp_e3(i,2) = x0_e + x_e3;
  xp_e3(i,3) = x0_e + x_e3;
  xp_e3(i,4) = x0_e + x_e2 + 1.0;
  xp_e3(i,5) = xp_e3(i,1);

  yp_e3(i,1) = y0_e + R_e2 * cos( alf1 );
  yp_e3(i,2) = y0_e + R_e3 * cos( alf1 );
  yp_e3(i,3) = y0_e + R_e3 * cos( alf2 );
  yp_e3(i,4) = y0_e + R_e2 * cos( alf2 );
  yp_e3(i,5) = yp_e3(i,1);

  zp_e3(i,1) = z0_e + R_e2 * sin( alf1 );
  zp_e3(i,2) = z0_e + R_e3 * sin( alf1 );
  zp_e3(i,3) = z0_e + R_e3 * sin( alf2 );
  zp_e3(i,4) = z0_e + R_e2 * sin( alf2 );
  zp_e3(i,5) = zp_e3(i,1);

  alf1 = alf2;
  alf2 = alf1 - dalf;
end

n_pn = 2;

xp_epn(1,1) = x0_e + x_e0 + 1.6;
xp_epn(1,2) = x0_e + x_e0 - 0.4;
xp_epn(1,3) = x0_e + x_e2 - 0.4;
xp_epn(1,4) = x0_e + x_e2;
xp_epn(1,5) = xp_epn(1,1);

yp_epn(1,1) = y0_e;
yp_epn(1,2) = y0_e;
yp_epn(1,3) = y0_e;
yp_epn(1,4) = y0_e;
yp_epn(1,5) = yp_epn(1,1);
```

```
zp_epn(1,1) = z0_e + 0.9;
zp_epn(1,2) = z0_e + 1.47;
zp_epn(1,3) = z0_e + 1.47;
zp_epn(1,4) = z0_e + 0.9;
zp_epn(1,5) = zp_epn(1,1);

xp_epn(2,1) = x0_e + x_e0;
xp_epn(2,2) = x0_e + x_e2;
xp_epn(2,3) = x0_e + x_e2;
xp_epn(2,4) = x0_e + x_e0 - 0.4;
xp_epn(2,5) = xp_epn(2,1);

yp_epn(2,1) = y0_e;
yp_epn(2,2) = y0_e;
yp_epn(2,3) = y0_e;
yp_epn(2,4) = y0_e;
yp_epn(2,5) = yp_epn(2,1);

zp_epn(2,1) = z0_e +  0.9;
zp_epn(2,2) = z0_e +  0.9;
zp_epn(2,3) = z0_e +  1.47;
zp_epn(2,4) = z0_e +  1.47;
zp_epn(2,5) = zp_epn(2,1);


m_e = 2270.0;

wing.m(j_e,j_e) = wing.m(j_e,j_e) + m_e;
%m(j,j-1) = m(j,j-1) - m_e * z_e^2 / (2.0*dy)^2;
%m(j,j+1) = m(j,j+1) + m_e * z_e^2 / (2.0*dy)^2;
%m(j_e,j_e)   = m(j_e,j_e)   + m_e * z_e^2 / (dy^2);
%m(j_e,j_e+1) = m(j_e,j_e+1) - m_e * z_e^2 / (dy^2);
wing.S(j_e,j_e) = wing.S(j_e,j_e) - m_e * x0_e;
wing.I(j_e,j_e) = wing.I(j_e,j_e) + m_e * ( x0_e^2.0 + z0_e^2.0 );


engnA320.xp_e1  = xp_e1;
engnA320.yp_e1  = yp_e1;
engnA320.zp_e1  = zp_e1;
engnA320.xp_e2  = xp_e2;
engnA320.yp_e2  = yp_e2;
engnA320.zp_e2  = zp_e2;
engnA320.xp_e3  = xp_e3;
engnA320.yp_e3  = yp_e3;
engnA320.zp_e3  = zp_e3;
engnA320.n_e    = n_e;
engnA320.xp_epn = xp_epn;
engnA320.yp_epn = yp_epn;
engnA320.zp_epn = zp_epn;
engnA320.n_pn   = n_pn;
engnA320.j_e    = j_e;
engnA320.m_e    = m_e;
```