Escola Tècnica Superior d'Enginyeries Industrial i Aeronàutica de Terrassa
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Secció de Terrassa

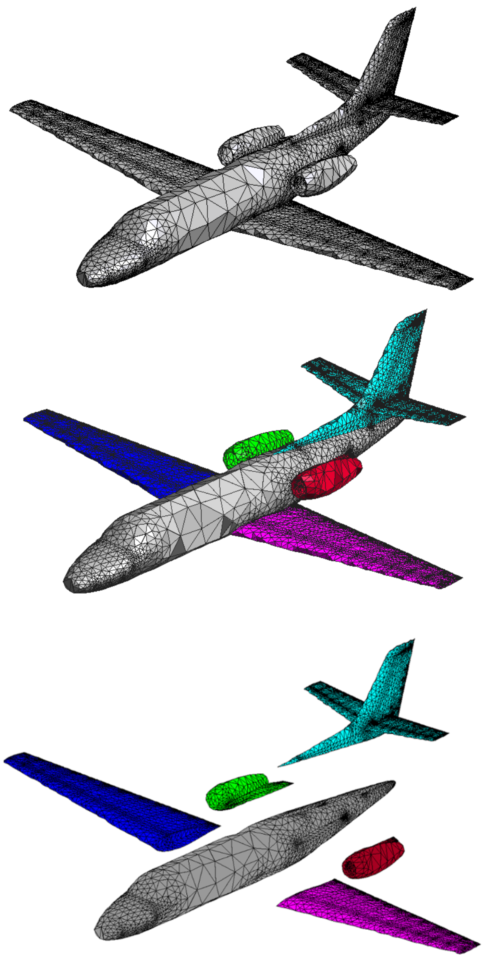
Centre: *ETSEIAT*

Lecture: *Departament de Resistència de Materialas*

Work Title:

Final Master Thesis

Study of the Model Order Reduction of an Entire Aircraft Configuration



Grupo: M2B-P2016/17

Delivery Date: 20th January, 2017

Students: Fabián Lajas Contreras

Director: Prof. Joaquín Hernández Ortega

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank to everyone who has helped me in the planification, development and achievement of this final master project, namely, to

- Joaquín A. Hernández Ortega
- Riccardo Rossi
- My parents and siblings

Nomenclature

$\ddot{\boldsymbol{d}}$	Acceleration.
$\boldsymbol{A}\,\boldsymbol{B}$	Array of constants.
$\boldsymbol{D}$	Dumping Matrix.
$\boldsymbol{d}$	Displacement.
λ_i	Eigenvalue.
$\boldsymbol{F}$	Force Vector.
$\boldsymbol{I}$	Identity Matrix.
$\boldsymbol{K}$	Stiffness Matrix, submodel Stiffness Matrix.
$\mathbf{K}_{BB}$	Boundary Stiffness Matrix.
$\mathbf{K}_{BI}$	Interface Stiffness Matrix.
$\mathbf{K}_{II}$	Internal Stiffness Matrix.
$\overline{\mathbf{X}}_I$	Reduced Modal Shape.
$\boldsymbol{M}$	Mass Matrix, submodel Mass Matrix.
$\mathbb{R}$	Real Numbers.
$\mathbf{M}_{BB}$	Boundary Mass Matrix.
$\mathbf{M}_{BI}$	Interface Mass Matrix.
$\mathbf{M}_{II}$	Internal Mass Matrix.
$\boldsymbol{\phi}$	Modal Vector.
$\boldsymbol{\Omega}$	Spectral Matrix.
$\boldsymbol{\omega}$	Natural Frequency.
$\boldsymbol{p}$	External Forces.
$\boldsymbol{q}$	Amplitude.
$\mathbf{R}_{CBTM}$	Craig-Bampton Transformation Matrix.
$\tilde{\mathbf{K}}$	Reduced Stiffness Matrix.
$\tilde{\mathbf{M}}$	Reduced Mass Matrix.
ρ	Material density.
$\tilde{\mathbf{K}}_{BB}$	Reduced Inner Stiffness Matrix.
$\tilde{\mathbf{K}}_T$	Total reduced Stiffness Matrix.
$\tilde{\mathbf{M}}_B$	Reduced Inner Mass Matrix.
$\tilde{\mathbf{M}}_T$	Total reduced Mass Matrix.
$\overline{\mathbf{R}}_{CBTM}$	Reduced Craig-Bampton Transformation Matrix.
$\boldsymbol{u}_B$	Boundary Results
$\boldsymbol{u}_C$	Compressed Results
$\boldsymbol{u}$	Total Results
$\dot{\boldsymbol{d}}$	Velocity.
$\mathbf{x}$	Total degrees of Freedom.
$\mathbf{x}_B$	Subsystem internal degrees of freedom.
$\mathbf{x}_I$	Boundary degrees of freedom.
$\mathbf{0}$	Array or matrix composed by zeros.
G	Torsional Inertia Product.
I_x	Inertia Product in the x axis.
I_y	Inertia Product in the y axis.
I_z	Inertia Product in the z axis.
l	Length of a slender beam.
m	Number of reduced DOFs.
n	Number of elements present in an enumeration.
n_B	Number of boundary DOFs.
t	Shell Thickness.
A	Cross sectional area for a slender beam.
E	Young Modulus.
t	Time.

1 Introduction

1.1 Aim of the Study

The primary aim of this study is to understand and implement in a computer program the basic theory underlying *model order reduction* of Finite Element (FE) structural models. More concretely, attention is focused on *modal analysis* via *substructuring* techniques, and the interest lies in comparing the performance of such substructuring methods with the standard modal analysis of linear elastodynamic FE models. The method is applied to 3 aeronautical structures, ranging in complexity from a simple truss structure to a complete airframe of a private jet aircraft. For the truss structure, all required operations are carried out with a Matlab code developed by the author. For the other two structures, the finite element information (mass and stiffness matrices) is retrieved from an open source FE software called *KRATOS*, and then processed by a Matlab script in order to determine the vibrational modes and the natural frequencies. It should be highlighted that the construction of the geometric models of the three tested airframes have been also developed, from scratch, by the author.

1.2 State-Of-The-Art

1.2.1 Origins of Substructuring Methods

Substructuring methods were invented in the early 1960s by aircraft engineers to carry out a first-level breakdown of complex systems such as a complete aircraft shown in Figure 1. The original factors that motivated the development of this kind of techniques were, according to [1]:

- **To Facilitate division of labor**: Substructures with different functions are done by different teams. This fact would allow different groups to work separately with different parts of the project, this fact would allow to perform geometry modifications, nevertheless it is important to remind that the interface nodes that each substructure contains must remain the same.
- **Take advantage of repetition**: It is usual for this kind of projects to have symmetric parts in the structure, for instance, it is common to deal with aircrafts which contain symmetry planes that split by the half the entire structure, recognizing repetitions, saves model creation and, of course, time.
- **Overcome computer limitations**: This kind of techniques allow to analyze entire huge models part by part, saving computational time and surpassing the memory limitations that a usual machine may have.

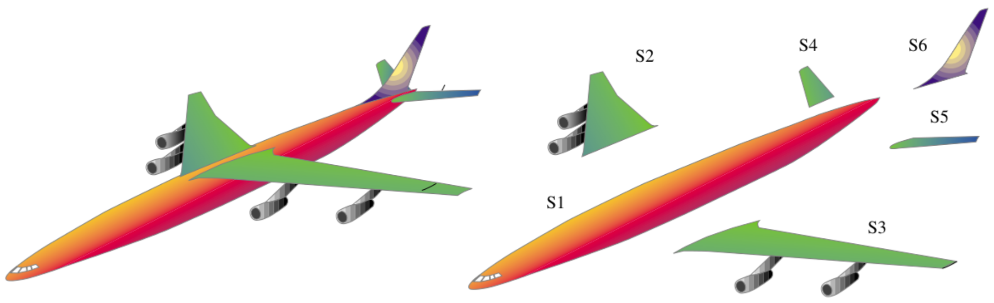


Figure 1: Example of a substructured system; the complete structure has been broken down into six different components. Image courtesy of [1].

1.2.2 Reduced Order Models based on substructuring

In this project, the reduced order models (ROMs) based on substructuring are intended to avoid memory overload as well as to save computational time. Several ROMs of this type have been developed during the 1960s and the 1970s. They differ from each other in the choice of the matrix that is used in the reduction process. According to [7], some of the most important ROMs are:

- *The Guyan-Irons reduction algorithm, 1965*, this system can be seen both as a degree of freedom economizer and as a substructuring method.
- *The Craig-Bampton method, 1968*, also commonly referred to as *the component modes method* or *the modal synthesis*, which is mainly used as a dynamic substructuring method. This is the strategy chosen for this project.
- *The McNeal hybrid synthesis method, 1971*, which uses a concept called general reduction scheme, see [7] for further information. T
- *Rubin method, 1975.* This methodology is a modification of the McNeal hybrid synthesis method, commented above.

1.3 Scope

As mentioned in Section 1.1, attention is confined to model order reduction via modal analysis in the elastodynamic linear range. Therefore, nonlinear material behavior and buckling phenomena are excluded in this study. Finite element discretization is carried out using beam and shell elements —albeit the proposed methodology is general and can be applied also to finite element models with solid elements.

1.4 Requirements

The minimum requirements for this project are the analysis of several aeronautical-like structures in order to test the validity of the created code. This code will be able to compare the results obtained for a standard modal analysis and a reduced order analysis. The expected results for such kind of problems are the modal frequencies and the vibrational modes, as it has been commented in *Section 1.3* .

1.5 Employed tools

In this study, the following software is going to be used:

- Solid Works ^{TM} v2016. Software used to edit and modify the geometries. This application will be run in a Windows ^{TM} machine.
- Libre CAD . As it was commented in the previous tool, this software is used to perform CAD drawings; in this case, though, the software is going to be used to extract the geometry points from a certain airplane, as it will be seen in *Section 4.3* .
- GiD ^{TM} 13. Software in charge of meshing the previously edited geometry and prepare the data for the FE software Kratos. For this study, this application will be run in an Ubuntu 14.04 LTS machine.
- Matlab ^{TM} R2015a. Computerized platform optimized for solving engineering and scientific problems. In this project, this software will be used in an Ubuntu 14.00 LTS machine.
- Kratos. Open source framework for resolution of multiphysics finite element problems. Like the previous programs, this software will be run in an Ubuntu 14.04 LTS machine.

1.6 Outline of the study

The main structure of this Final Master Project will begin with a complete explanation of the theories used. The next part will describe some applications coded using housemade software in order to test these theories. Three main problems will be tested: A beam geometry problem, a Reissner-Mindlin Flat Shell Element problem, and a Reissner-Mindlin Flat Shell Element in an aeronautical-like structure. After the resolution of these three problems, the following part is reserved to comment possible research lines that can be started after this study. Finally, the final conclusions for this project will be presented.

2 Theoretical Basis

First of all, it is required to define the concepts that take part each time a substructural analysis is done. This kind of techniques require the understanding of several disciplines: Advanced Mathematical Concepts, Mechanical Analysis, FEM Theory and Computing Language Skills. In this project it is intended to focus in the modal analysis, that can also be identified as a ROM method. This method will be computed by the program. The selected FE elements for each kind of problem are going to be also analyzed in this project. Finally, the implemented substructuring method will be defined.

2.1 Modal Analysis

Modal Analysis is a common ROM technique that consists in the extraction of the vibrational modes present in a structure. In this section the basic mathematical concepts underlying modal analysis procedures are described. We begin by presenting the classical, semi-discrete finite element equation governing the dynamic behavior of a solid body:

Given $\boldsymbol{F} : [0,T] \to \mathbb{R}^n$, find $\boldsymbol{d} : [0,T] \to \mathbb{R}^n$ such that:

$$\boldsymbol{M}\ddot{\boldsymbol{d}} + \boldsymbol{D}\dot{\boldsymbol{d}} + \boldsymbol{K}\boldsymbol{d} = \boldsymbol{F} \tag{1}$$

$$\boldsymbol{d}(0) = \boldsymbol{d^0} \tag{2}$$

$$\dot{\boldsymbol{d}}(0) = \dot{\boldsymbol{d}}^{\boldsymbol{0}} \tag{3}$$

The following solution can be proposed for this equation:

$$\boldsymbol{d} = \boldsymbol{d^p} + \boldsymbol{d^z} \tag{4}$$

Where the $\boldsymbol{d^z} : [0,T] \to \mathbb{R}^n$ is a general solution of the homogeneous equation,

$$\boldsymbol{M}\ddot{\boldsymbol{d}} + \boldsymbol{D}\dot{\boldsymbol{d}} + \boldsymbol{K}\boldsymbol{d} = \boldsymbol{0} \tag{5}$$

The conditions for an undamped free vibration can be defined as $(\boldsymbol{D} = 0, \boldsymbol{F} = 0)$. Considering these conditions, the undamped equation can be established as:

$$\boldsymbol{M}\ddot{\boldsymbol{d}}^{\boldsymbol{z}} + \boldsymbol{K}\boldsymbol{d^z} = \boldsymbol{0} \tag{6}$$

Once the main equation has been simplified, it is possible to define:

$$\boldsymbol{q}(t) = \boldsymbol{A}cos(\omega t) + \boldsymbol{B}sin(\omega t) \tag{7}$$

The form of the solution can be expressed as follows:

$$\boldsymbol{d}(t) = \boldsymbol{\phi}\boldsymbol{q}(t) = \boldsymbol{\phi}(\boldsymbol{A}cos(\omega t) + \boldsymbol{B}sin(\omega t)), where : \boldsymbol{\phi} \in \mathbb{R}^{\mathbf{n}} \tag{8}$$

Applying this kind of solution, it is possible to modify the dynamic equation:

$$\boldsymbol{M}\left(\boldsymbol{\phi}\frac{d^2q(t)}{dt^2}\right) + \boldsymbol{K}\,\boldsymbol{\phi}\,q(t) = 0 \tag{9}$$

Due to the harmonic nature of the solution, it is possible to establish:

$$\frac{d^2q(t)}{dt^2} = -\omega^2 q(t) \tag{10}$$

So the dynamic equation can be now expressed as:

$$\left(-\boldsymbol{\omega}^2\boldsymbol{M} + \boldsymbol{K}\right)\boldsymbol{\phi}\boldsymbol{q}(t) = \boldsymbol{0} \tag{11}$$

As it can be seen in Eq. (11), the dynamic equation has become an eigenvalue problem. This kind of problems have two main solutions:

- $\boldsymbol{\phi}\,(t) = 0$, which is the trivial solution.
- The i_{th} nontrivial solutions, each one of them per every dynamic D.O.F. It consists of the natural frequency of vibration ω_i, that can be related with its corresponding *eigenvalue* by the following expression: ${\omega_i}^2 = \lambda_i$, and its associated vibration mode ϕ_i, also called *eigenvector*.

Once the problem has been completely defined, several aspects should be commented:

- The natural frequency can also be called ***resonant frequency***. The eigenvector can also be called a ***natural mode***.

- The smallest frequency: ω_i, it is also called the **fundamental frequency of vibration**.
- Each one of the non-trivial solutions accomplish the following equation:

$$\boldsymbol{\omega}_i{}^2 \boldsymbol{M}\boldsymbol{\phi}_i = \boldsymbol{K}\boldsymbol{\phi}_i \tag{12}$$

It can be defined the complete problem such as the solution of the undamped free vibration problem ($\boldsymbol{D} = 0, \boldsymbol{F} = 0$).

Given $\boldsymbol{d}^\mathbf{0} \in \mathbb{R}^\mathbf{n}$, find $\boldsymbol{d} : [0, T] \rightarrow \mathbb{R}^\mathbf{n}$, such that:

$$\boldsymbol{M}\ddot{\boldsymbol{d}} + \boldsymbol{K}\boldsymbol{d} = \mathbf{0} \tag{13}$$

$$\boldsymbol{d}(0) = \boldsymbol{d}^\mathbf{0} \tag{14}$$

$$\dot{\boldsymbol{d}}(0) = \dot{\boldsymbol{d}}^\mathbf{0} \tag{15}$$

Solution:

$$\boldsymbol{d} = \sum_{i=1}^{n} \boldsymbol{\phi}_i \left(\boldsymbol{q}_i^\mathbf{0} cos(\omega_i t) + \frac{\dot{\boldsymbol{q}}_i^\mathbf{0}}{\omega_i} sin(\omega_i t) \right) \tag{16}$$

where:

$$\boldsymbol{q}_i^0 = \boldsymbol{\phi}_i^T \boldsymbol{d}^\mathbf{0} \;\; and \;\; \dot{\boldsymbol{q}}_i^0 = \boldsymbol{\phi}_i^T \dot{\boldsymbol{d}}^\mathbf{0} \tag{17}$$

More information regarding modal analysis can be found in [8].

2.2 Basic FEM concepts

Next we offer a quick review of the elements that will be employed in this project. If more information is required, the reader is referred to [9] and [10]. For this project, the employed FE elements are the Euler-Bernouilli beam element, and the Reissner-Mindlin Flat Shell Element .

2.2.1 The 2-Noded Euler Bernouilli Beam Element

The Euler-Bernouilli beam element is commonly used to solve structural problems in which the models are composed by slender bodies. In Figure 2, the representation of a single element is displayed. This theory relies on the following hypotheses:

1. The vertical displacement, also called *deflection* (w), of the points contained in a cross-section of the element is small and equal to the displacement of the beam axis.
2. The lateral displacement, also refered as v (along the y axis in Figure 2) is considered as zero.
3. Cross-sections normal to the *beam axis* remain plane and orthogonal to the beam axis after the deformation; this is usually called the *normal orthogonality condition.*

For this project the beams are formed by circular cross-sections. Once all the previous assumptions have been made, by using the FEM theory, it is possible to compute the stiffness and mass matrix for this kind of element —see equation Eq. (18) and Eq. (19). It is important to remember that the beam element consist of 2-noded element with 6 DOF for each node, so the computed matrix will be composed by 12 elements. For more information regarding this subject, the reader is referred to [11], in [2], [12], and in [13].

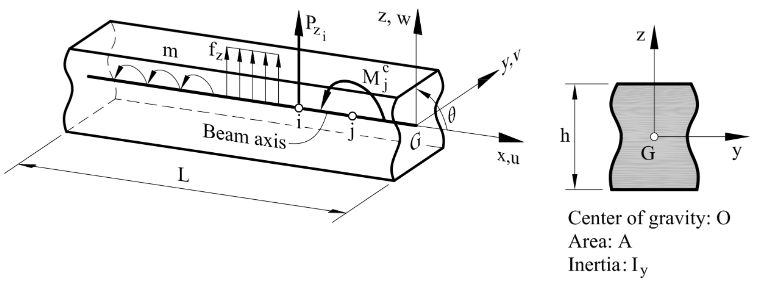


Figure 2: Two-noded Euler-Bernoulli beam element, image couurtesy of [2].

$$
\mathbf{K}_{eL} = \begin{bmatrix}
\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 \\
 & \frac{12EI_z}{l^3} & 0 & 0 & 0 & \frac{6EI_z}{l^2} & 0 & -\frac{12EI_z}{l^3} & 0 & 0 & 0 & \frac{6EI_z}{l^2} \\
 & & \frac{12EI_y}{l^3} & 0 & -\frac{6EI_y}{l^2} & 0 & 0 & 0 & -\frac{12EI_y}{l^3} & 0 & -\frac{6EI_y}{l^2} & 0 \\
 & & & \frac{GI_x}{l} & 0 & 0 & 0 & 0 & 0 & -\frac{GI_x}{l} & 0 & 0 \\
 & & & & \frac{4EI_y}{l} & 0 & 0 & 0 & \frac{6EI_y}{l^2} & 0 & \frac{2EI_y}{l} & 0 \\
 & & & & & \frac{4EI_z}{l} & 0 & -\frac{6EI_z}{l^2} & 0 & 0 & 0 & \frac{2EI_z}{l} \\
 & & & & & & \frac{EA}{l} & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & \frac{12EI_z}{l^3} & 0 & 0 & 0 & -\frac{6EI_z}{l^2} \\
 & & Sym. & & & & & & \frac{12EI_y}{l^3} & 0 & -\frac{6EI_y}{l^2} & 0 \\
 & & & & & & & & & \frac{GI_x}{l} & 0 & 0 \\
 & & & & & & & & & & \frac{4EI_y}{l} & 0 \\
 & & & & & & & & & & & \frac{4EI_z}{l}
\end{bmatrix} \tag{18}
$$

$$\mathbf{M}_{eL} = \rho A l \begin{bmatrix} \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{6} & 0 & 0 & 0 & 0 & 0 \\ & \frac{13}{35} & 0 & 0 & 0 & \frac{11l}{210} & 0 & \frac{9}{70} & 0 & 0 & 0 & -\frac{13l}{420} \\ & & \frac{13}{35} & 0 & -\frac{11}{210} & 0 & 0 & 0 & \frac{9}{70} & 0 & \frac{13l}{420} & 0 \\ & & & \frac{r_t^2}{3} & 0 & 0 & 0 & 0 & 0 & \frac{r_t^2}{6} & 0 & 0 \\ & & & & \frac{l^2}{105} & 0 & 0 & 0 & -\frac{13l}{420} & 0 & -\frac{l^2}{140} & 0 \\ & & & & & \frac{l^2}{105} & 0 & \frac{13l}{420} & 0 & 0 & 0 & -\frac{l^2}{140} \\ & & & & & & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & \frac{13}{35} & 0 & 0 & 0 & -\frac{11l}{210} \\ & & Sym. & & & & & & \frac{13}{35} & 0 & \frac{11l}{210} & 0 \\ & & & & & & & & & \frac{r_t^2}{3} & 0 & 0 \\ & & & & & & & & & & \frac{l^2}{105} & 0 \\ & & & & & & & & & & & \frac{l^2}{105} \end{bmatrix} \tag{19}$$

2.2.2 Reissner-Mindlin Flat Shell Element

For this study, the Kratos environment will be in charge of computing the stiffness and the mass matrix for each shell element —that is, Kratos will act as some sort of *black box*. A graphic representation of quadrilateral and triangular meshes is shown in Figure 3,

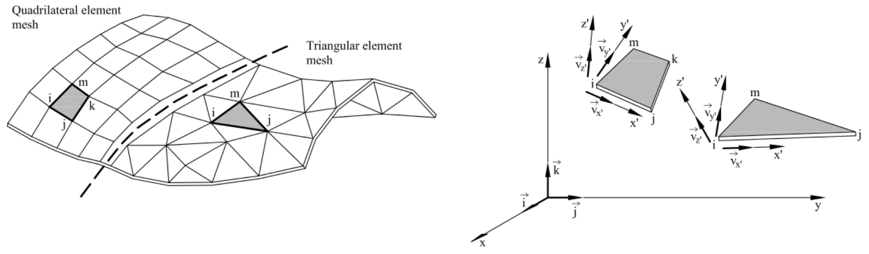


Figure 3: Mesh discretization, it can be distinguished two kinds of shells, the quadrilateral mesh at left, and the triangular mesh at right. For this study, the triangular element will be used. Image courtesy of [2].

Shell elements are special elements that combine torque field with plane stress fields. They work as the combination of a *Kirchhoff Thin Plate*, which captures bending, and a membrane, that captures axial deflections [2]. We have used in all the simulations presented henceforth the *TLLL* element, which refers to the Flat Shell Triangles combining 3-noded linear plane stress triangle; theoretical details concerning this element can be found in [2]. This kind of element contains the following features:

1. It consist in a 3-noded linear plane stress triangle.
2. The deflection is linearly interpolated in terms of the three corner values.
3. The Rotations are computed by using a linear interpolation in terms of mid-side values.

4. Linear assumed transverse shear strain field is assumed as in other Reissner-Mindlin element formulation: the *TLQL*. More information about this shell model can be found in [2].

2.3 Substructuring Techniques

Substructuring methods are frequently used in dynamic analysis; the main reasons that motivated the development of these methods are:

- The interest in lower frequency eigensolutions, which is common in structural analysis, and it may prove to be advantageous to reduce the starting problem to a smaller dimension.
- In large projects, such as aircrafts, separate parts of the complete system can be analized by each project team, later these parts will be used to reconstruct the model of the whole system. Therefore, the interconnecting surfaces must be defined carefully in order to ensure the compatibility of the different parts of the model, as it can be seen in Figure 4.

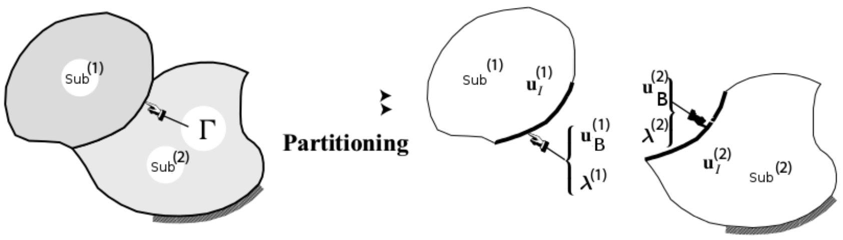


Figure 4: Partitioning of a Structure into Two Substructures. In the figure Γ refers to the boundary between two substructures. Image courtesy of [3].

2.3.1 Craig-Bampton Method

The Craig-Bampton method, (Craig and Bampton 1968), also commonly referred to as the *component modes method or the modal synthesis* is mainly used as a dynamic substructuring method. It consist in reducing the complexity —via standard modal analysis— of each part of the model. The first step in the formulation of this method is the separation into inner nodes and boundary nodes:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_I \\ \mathbf{x}_B \end{bmatrix} , \; \boldsymbol{K} = \begin{bmatrix} \mathbf{K}_{II} & \mathbf{K}_{IB} \\ \mathbf{K}_{BI} & \mathbf{K}_{BB} \end{bmatrix} , \; \boldsymbol{M} = \begin{bmatrix} \mathbf{M}_{II} & \mathbf{M}_{IB} \\ \mathbf{M}_{BI} & \mathbf{M}_{BB} \end{bmatrix} \; \textit{and} \; \boldsymbol{F} = \begin{bmatrix} \mathbf{0} \\ \mathbf{p}_B \end{bmatrix} \tag{20}$$

Considering the modal analysis equation, Eq. (21), the total system can be rewritten by using the previously defined equations; the final result is Eq. (22).

$$\boldsymbol{K}\mathbf{x} - \boldsymbol{\omega}^2 \boldsymbol{M}\mathbf{x} = \boldsymbol{F} \tag{21}$$

$$\begin{bmatrix} \mathbf{K}_{II} & \mathbf{K}_{IB} \\ \mathbf{K}_{BI} & \mathbf{K}_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{x}_I \\ \mathbf{x}_B \end{bmatrix} - \boldsymbol{\omega}^2 \begin{bmatrix} \mathbf{M}_{II} & \mathbf{M}_{IB} \\ \mathbf{M}_{BI} & \mathbf{M}_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{x}_I \\ \mathbf{x}_B \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{p}_B \end{bmatrix} \tag{22}$$

Considering that if there were no inertia forces, the internal degrees of freedom could be computed by static condensation, it is possible to define the $\mathbf{x}_I$ as:

$$\mathbf{x}_I = -\mathbf{K}_{II}^{-1}\mathbf{K}_{IB}\mathbf{x}_B \tag{23}$$

Boundary DOFs $\mathbf{x}_B$ will be considered as static DOFs, and an eigenvaule problem will be solved by considering the internal DOFs as the free ones:

$$\mathbf{K}_{II}\,\mathbf{X}_I = \boldsymbol{\omega}_I^2\,\mathbf{M}_{II}\,\mathbf{X}_I \tag{24}$$

Of course, according to the modal problem, the fixed interface modes are stored in the columns of matrix $\mathbf{X}_I$ so that:

$$\mathbf{X}_I\,\mathbf{K}_{II}\,\mathbf{X}_I = diag(\omega_{I\,1}^2\ \ ...\ \ \omega_{I\,nI}^2) = \boldsymbol{\Omega}_I^2 \tag{25}$$

$$\mathbf{X}_I\,\mathbf{K}_{II}\,\mathbf{X}_I = \boldsymbol{I} \tag{26}$$

It is possible to build the total DOFs x by using the following relation:

$$\mathbf{x} = \mathbf{R}_{CBTM}\begin{bmatrix}\boldsymbol{\eta}\\ \mathbf{x}_B\end{bmatrix} \tag{27}$$

where $\mathbf{R}_{CBTM}$ is what it is called the Craig Bampton Transformation Matrix:

$$\mathbf{R}_{CBTM} = \begin{bmatrix}\mathbf{X}_I & -\mathbf{K}_{II}^{-1}\,\mathbf{K}_{IB}\\ \mathbf{0} & \boldsymbol{I}\end{bmatrix} \tag{28}$$

and η refers to the intensity parameters of the substructure's internal vibration modes. Once all these elements have been defined, the initial model of each substructure will be reduced. Only a certain number of internal modes I will be chosen:

$$\overline{\mathbf{X}}_I = \begin{bmatrix}\mathbf{x}_{I(1)} & ... & \mathbf{x}_{I(m)}\end{bmatrix} \tag{29}$$

On the other hand, in this reduction process, it is necessary to maintain the boundary DOFs in order to assure deformation compatibility at the interfaces; the final form of CBTM will be:

$$\overline{\mathbf{R}}_{CBTM} = \begin{bmatrix}\overline{\mathbf{X}_I} & -\mathbf{K}_{II}^{-1}\,\mathbf{K}_{IB}\\ \mathbf{0} & \boldsymbol{I}\end{bmatrix} \tag{30}$$

The dimension of this matrix will be $n \times (m + n_B)$. The reduced stiffness and mass matrix will be extracted by performing the following operations:

$$\tilde{\mathbf{K}} = \overline{\mathbf{R}}_{CBTM}^T\,\boldsymbol{K}\,\overline{\mathbf{R}}_{CBTM}\ \ and\ \ \tilde{\mathbf{M}} = \overline{\mathbf{R}}_{CBTM}^T\,\boldsymbol{M}\,\overline{\mathbf{R}}_{CBTM} \tag{31}$$

For each submodel:

$$\tilde{\mathbf{M}} = \begin{bmatrix} \tilde{\mathbf{\Omega}}_I^2 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{K}}_{BB} \end{bmatrix} \quad \textit{and} \quad \tilde{\mathbf{K}} = \begin{bmatrix} \boldsymbol{I} & \tilde{\mathbf{M}}_B \\ \tilde{\mathbf{M}}_B & \tilde{\mathbf{K}}_{BB} \end{bmatrix} \tag{32}$$

If a problem with N substructures is considered, every substructure will have a transformed stiffness matrix and a transformed mass matrix. For the i-th substructure, it is defined as $\tilde{\mathbf{K}}_i$ and $\tilde{\mathbf{M}}_i$. The total number of boundaries present in the model can be defined as M. Then the final assembled matrix for the entire system can be defined as: $\tilde{\mathbf{K}}_T$ and $\tilde{\mathbf{M}}_T$:

$$\tilde{\mathbf{K}}_T = \begin{bmatrix} \tilde{\mathbf{\Omega}}^2_{I\{1\ldots m\}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{K}}_{BB\{1\ldots n_B\}} \end{bmatrix} \tag{33}$$

$$\tilde{\mathbf{M}}_T = \begin{bmatrix} \boldsymbol{I}_{\{1\ldots m\}} & \tilde{\mathbf{M}}_{BB\{1\ldots m,1\ldots n_B\}} \\ \tilde{\mathbf{M}}_{BB\{1\ldots n_B,1\ldots m\}} & \tilde{\mathbf{M}}_{BB\{1\ldots n_B\}} \end{bmatrix} \tag{34}$$

These matrix will have the following expanded form:

$$\tilde{\mathbf{K}}_T = \begin{bmatrix} \tilde{\mathbf{\Omega}}^2_{I\ 1} & \ldots & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \tilde{\mathbf{\Omega}}^2_{I\ m} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \ldots & \mathbf{0} & \tilde{\mathbf{K}}_{BB\ 1} + \tilde{\mathbf{K}}_{BB\ j} & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & \ldots & \tilde{\mathbf{K}}_{BB\ 1} + \tilde{\mathbf{K}}_{BB\ j} \end{bmatrix} \tag{35}$$

$$\tilde{\mathbf{M}}_T = \begin{bmatrix} \boldsymbol{I}_1 & \ldots & \mathbf{0} & \tilde{\mathbf{M}}_{B\ 1} & \ldots & \mathbf{0} \\ \vdots & \boldsymbol{I}_j & \vdots & \tilde{\mathbf{M}}_{B\ j} & \ddots & \tilde{\mathbf{M}}_{B\ n_B} \\ \mathbf{0} & \ldots & \boldsymbol{I}_m & \mathbf{0} & \ldots & \tilde{\mathbf{M}}_{B\ m,n_B} \\ \tilde{\mathbf{M}}_{B\ 1} & \tilde{\mathbf{M}}_{B\ j} & \mathbf{0} & \tilde{\mathbf{M}}_{BB\ 1} + \tilde{\mathbf{M}}_{BB\ j} & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \tilde{\mathbf{M}}_{B\ n_B} & \tilde{\mathbf{M}}_{B\ n_B,m} & \mathbf{0} & \ldots & \tilde{\mathbf{M}}_{BB\ 1} + \tilde{\mathbf{M}}_{BB\ j} \end{bmatrix} \tag{36}$$

Now that the entire system is assembled, the required analysis is applied. Then the modal analysis is employed once the entire system is assembled. After that, it will be necessary to decompress the obtained modal vectors . This process can be done by using the equation Figure 37. In this equation $\boldsymbol{u}_C$ refers to the compressed results, $\boldsymbol{u}_B$ denotes the boundary results, and finally $\boldsymbol{u}$ are the final decompressed results of the analysis.

$$\begin{bmatrix} \boldsymbol{u} \end{bmatrix} = \overline{\mathbf{R}}_{CBTM} \begin{bmatrix} \boldsymbol{u}_C \\ \boldsymbol{u}_B \end{bmatrix} \tag{37}$$

3 Application for Beam Geometry

In this part, the main code structure required to solve a modal analysis using a substructured technique will be introduced. It will be used MATLAB ™ in order to create the main algorithm. On the other hand, GID ™ will be used for the meshing process. In the following section the code structure will be deeply studied. Later, a beam problem will be solved and the results obtained will be compared and commented.

3.1 Basic Code Structure

The program will compute the modal analysis of an aeronautic structure: the fuselage, the wings, the stabilizers, the landing gears, or even entire body configurations. As it was said in *Section* **??** , the results obtained by two methods will be compared. It is for this reason that the created code will contain the following algorithms:

- the Classic modal analysis algorithm, which consists in the analysis of the entire geometry without using any kind of substructuring method. This process may prove slow, specially for large geometries. Hereafter, we shall refer to this process as **classical algorithm** .
- The Craig-Bampton modal analysis algorithm. This algorithm will analyze the entire geometry using the substructuring method. Hereafter, it will be referred as **Craig-Bampton algorithm** .

The code structure comprises several functions and subfunctions created using the vectorization technique. According to [14], vectorization consists in the transformation of a code in order to use nonscalar objects. Using this technique, speedup factors of almost 10 can be reached. In Figure 5, the first code structure is presented, which is composed by the following kinds of objects:

1. **ORANGE TAGS** are used as starting and ending execution points, and do not refer to functions. For this kind of tag the following parts are defined:
   (a) *START*: This part is mainly used as the starting point of the program.
   (b) *END*: As it was commented in the last item, this tag works as a state of the code. In this case it works as the ending point of the program.
2. **BLACK TAGS** refer to decision points. These decision points are triggered by a boolean object that must be correctly filled before starting the program execution.
   (a) *EDITMODE*: Refers to a conditional state. Depending on the value of the boolean object, it will take one or another path.
   (b) *SUBSTRUC*: *EDITMODE* will behave in the same way as the previous tag.
3. **RED TAGS** are used to mark functions that perform an auxiliary functionality.
   (a) *VALIDATOR*: Validator refers to an external functionality that assures that all the inputs present in the program are correctly introduced.
4. **BLUE TAGS** refer to the main functionalities run by the program.
   (a) *INPUT SETTINGS*: In this part, the main settings of the program are defined.

(b) *INPUT GEOMETRY RECIPE*: In this part, the geometry to analyze is defined.

(c) *CREATE GEOMETRY*: This function is in charge of creating the geometry using the information introduced in the previous state.

(d) *READMSH*: In this part, the mesh of the problem and the main conditions (materials and fixed DOFs), are extracted.

(e) *MATRIX ASSEMBLY*: This part is only activated if the *SUBSTRUC* switch is turned-off, it is in charge of computing the elemental stiffness and mass matrix, perform the assembly and applying the conditions defined in *READMSH*.

(f) *SUBSTRUC ALGORITHM*: This part is activated only if the *SUBSTRUC* switch is turned-on. This part will divide the mesh components into different substructures; it also will dump all the information into the hard drive in order to save memory.

(g) *MATRIX ASSEMBLY SUB*: As in the previous tag, *SUBSTRUC ALGORITHM*, this part can only be accessed by switching-on *SUBSTRUC*; it will read each one of the substructures, perform the Craig-Bampton transformation, and finally, compute the final assembly matrix.

(h) *EIGENVAL ALGORITHM*: This part will solve an eivenvalue problem.

(i) *MODAL SHAPE ALGORITHM*: This part will compute the modal shapes extracted by the previous tag. *EIGENVAL ALGORITHM*.

5. **GREEN TAGS** This kind of tag refers to the external software that it is used to perform special functionalities, such as the meshing algorithm, that is run by GiD ™ and the matrix computing and printing system, which is done by Kratos.

   (a) *GID MESHER*: External functionality can only be accessed by switching-off *EDITMODE*; it calls the software GiD ™ in order to load the created geometry and perform the meshing process.

   (b) *GID VIEWER*: External functionality that can only be accessed by switching-on *EDITMODE*; it calls the software GiD ™ in order to visualize the created geometry.

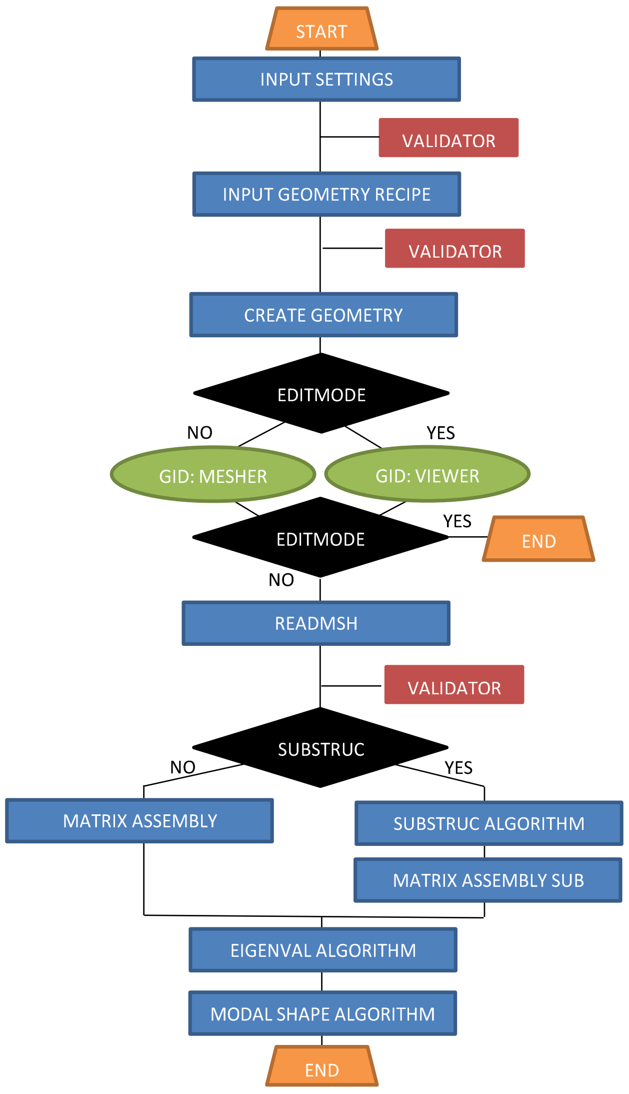


Figure 5: This is the first iteration of the code.

3.2 Beam Problem Definition

In this section, a modal analysis using the **Craig-Bampton algorithm** is performed. A 3D beam structure will be tested using the models defined in *Section 2.2.1* . The beam element is a reasonable solution for truss-like geometries such as the helicopter fuselage or classic airplanes such as those shown in Figure 6. This part will serve as a test for the **Craig-Bampton algorithm** .

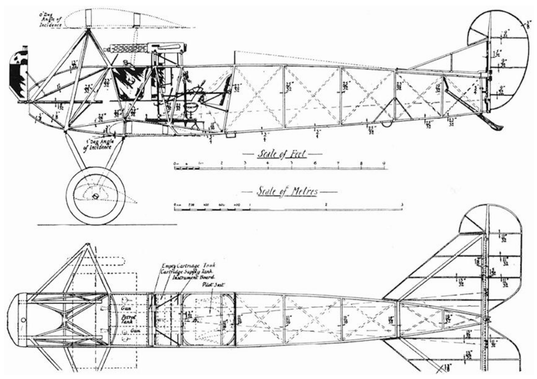


Figure 6: Fokker D.VII blueprints, it can be aprreciated the truss structure.

The geometry consists in a airplane-like structure composed by beams, as it can be seen in Figure 7. The model is formed by circular cross-section beams made of the same material. The physical properties chosen for this element are displayed in Table 1.

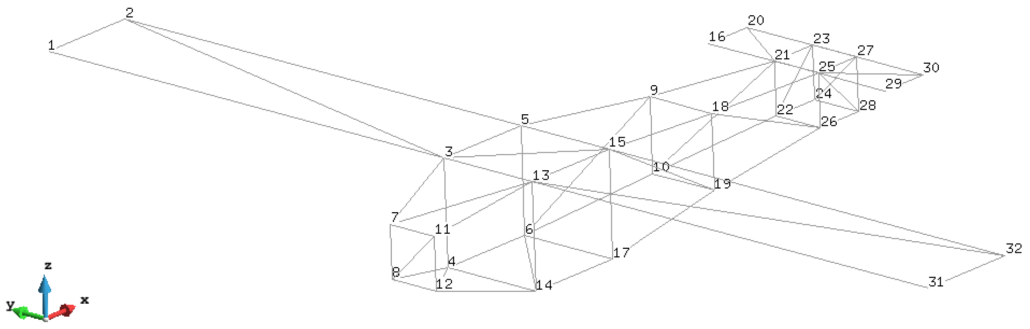


Figure 7: Total model, the structure nodes appear in black.

Table 1: Materials Table Chosen for this first problem.

$E\,[MPa]$	ν	$G\,[MPa]$	R_b $[m]$	A_b $[m^2]$	I_x $[m^4]$	I_y $[m^4]$	I_z $[m^4]$	ρ $[Mkg/m^3]$
206900	0.29	80193.7984	0.025	0.196e-2	6.136e-007	3.068e-007	3.068e-007	7.85e-003

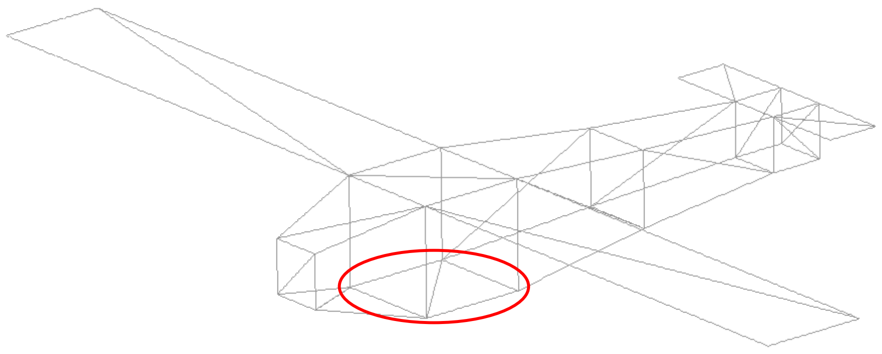


Figure 8: The restricted nodes for this modal analysis will be located at the bottom of the structure.

Just before solving the problem all the substructures are defined. In Figure 9 the chosen substructures for this example are presented.

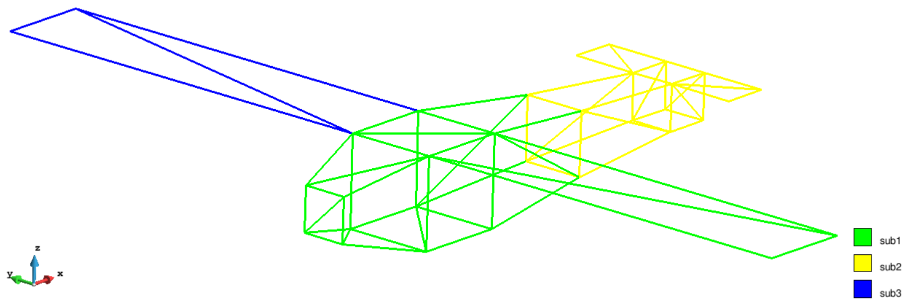


Figure 9: Complete model. Each substructure has been marked with a different colour: *Green* for the first substructure, *Yellow* for the second substructure, *Blue* for the third substructure.

3.3 Results Extracted for a Beam Problem

The number of inner DOFs have been set to 6. In Table 2, the natural frequencies for each one of the substructure are presented. The three lowest modal shapes for each substructure are shown in Figure 10. On the other hand, the tenth lowest natural frequencies for the total analysis can be found in Table 3. Finally the 4 lowest modal shapes are presented in Figure 11, Figure 12, Figure 13 and Figure 14.

Table 2: In this table, it can be seen the modal analysis performed at each substructure.

Freq [Hz]	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6
sub 1	2.2712	9.8239	14.2458	14.5011	15.5774	16.6988
sub 2	22.1827	52.4582	54.3966	81.5935	113.3861	124.2681
sub 3	2.4120	10.0043	14.4276	15.3735	16.2497	17.0245

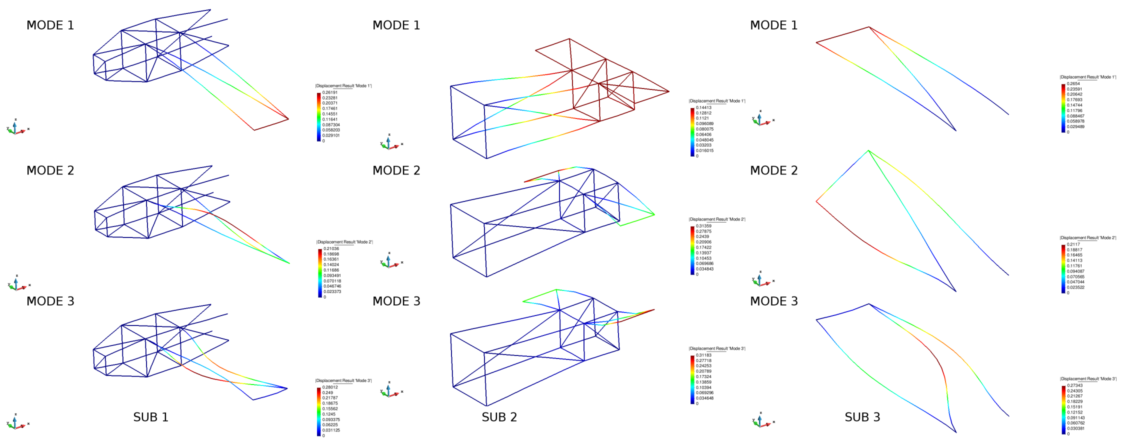


Figure 10: Modal Shapes Obtained for each substructure. It is important to comment that in these analysis, the boundary DOF's are considered as imposed DOFs.

Table 3: Natural Freqeucies, CA refers to **classical algorithm** , whereas CB refers to **Craig-Bampton algorithm** algorithm.

Freq $[Hz]$	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9	ω_{10}
CA	1.651	1.684	8.159	8.563	8.741	13.953	15.548	15.677	24.091	24.104
CB	1.647	1.676	8.530	8.550	11.791	15.532	15.647	18.330	23.958	24.089

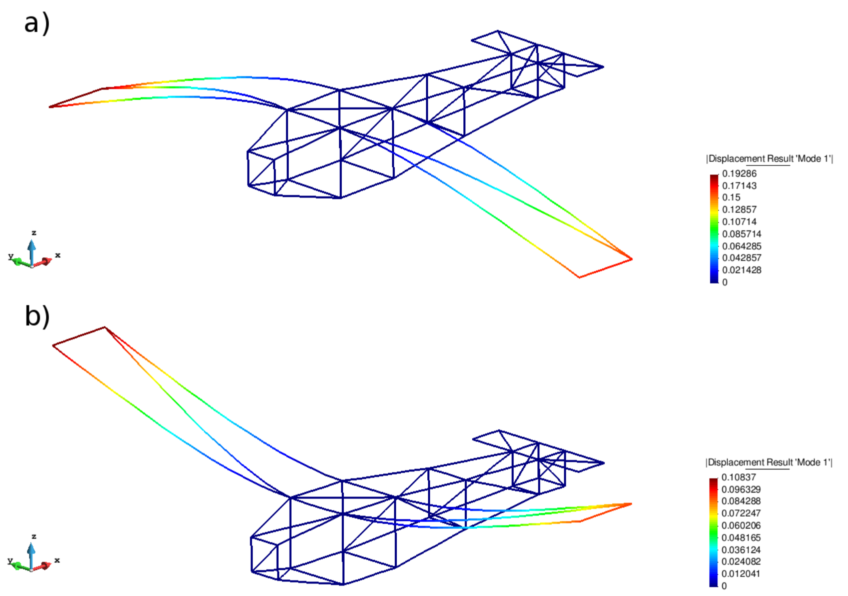


Figure 11: First Modal Shape, a) refers to the classic algorithm: CA, b) refers to the substructured algorithm: CB.

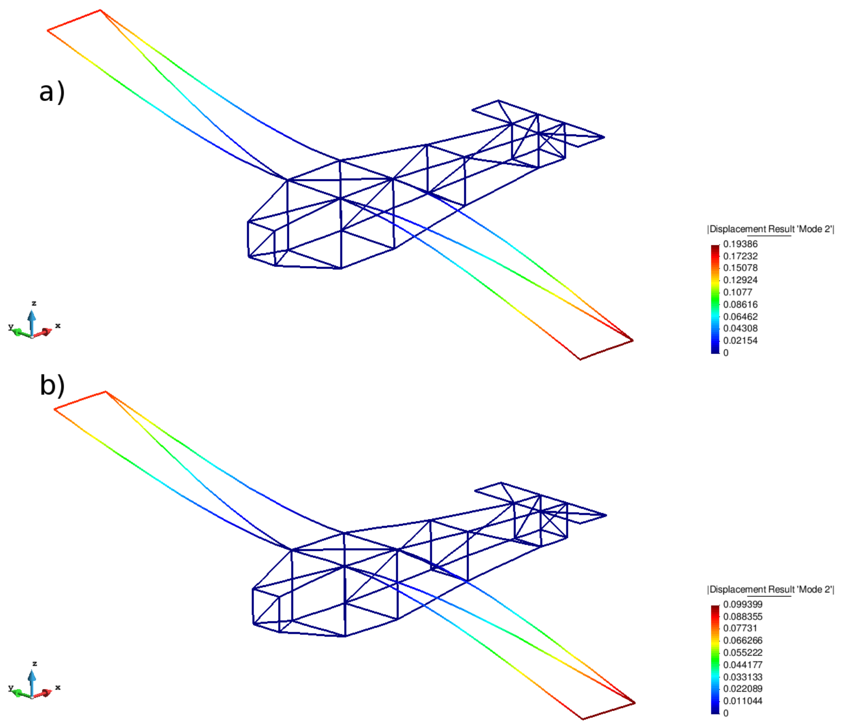


Figure 12: Second Modal Shape, a) refers to the classic algorithm: CA, b) refers to the substructured algorithm: CB.

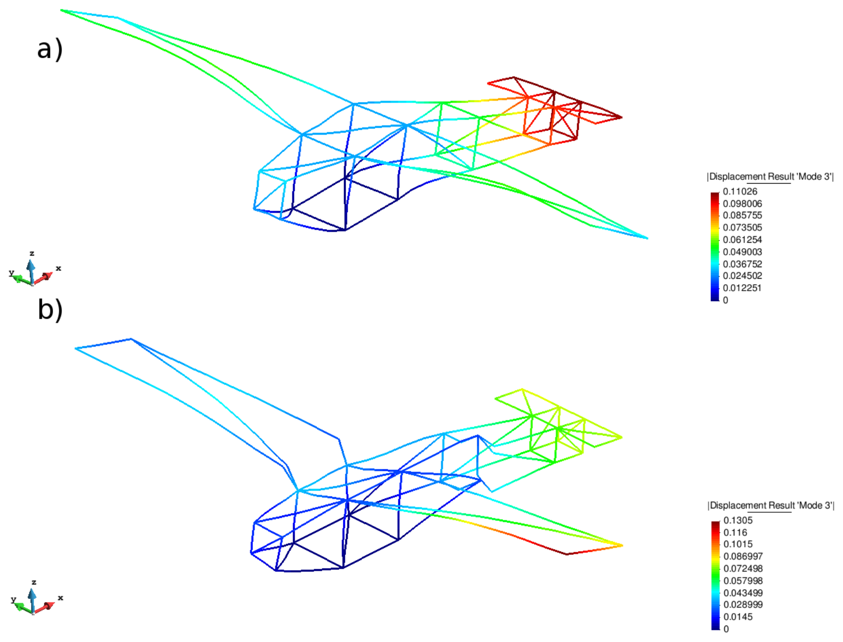


Figure 13: Third Modal Shape, a) refers to the classic algorithm: CA, b) refers to the substructured algorithm: CB.

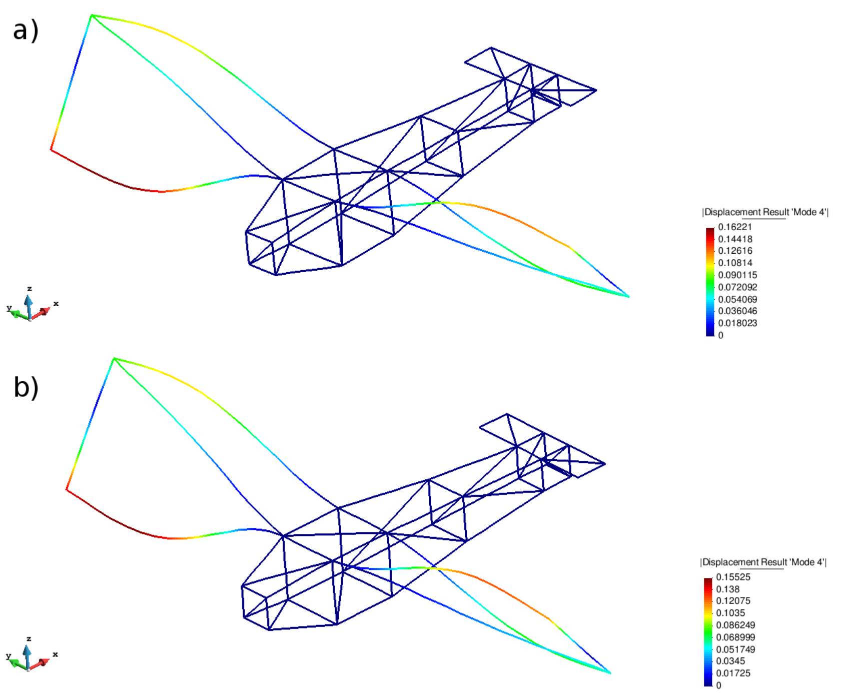


Figure 14: Fourth Modal Shape, a) refers to the classic algorithm: CA, b) refers to the substructured algorithm: CB.

3.4 Discussion of results

Inspection of the results shown in the foregoing reveals that, as expected, the natural frequencies computed by the CA and the CB analyses are quite similar (errors below 1 % are obtained for the first fifth modes). These low errors are also manifested in a significant resemblance between the modal shapes predicted by both methods (see Figure 11, Figure 12 and Figure 13). The only significant discrepancies are observed in the third mode. A plausible explanation for this notorious difference may be the loss of precision due to the compression-decompression process present in the **Craig-Bampton algorithm** .

4 Application for a Shell Geometry

Arriving at this point in the development the code, we decided to combine the flexibility of MATLAB ™ with the potential of KRATOS. A special module inside the original program was created in order to launch KRATOS. If this option is enabled, the program calls the KRATOS environment. This function computes and prints in a binary files each elemental mass and stiffness matrix. Once this process finishes, MATLAB ™ reads each file's information in order to continue with the modal analysis.

To begin with, it is necessary to explain how a basic problem works. KRATOS is a framework for building multi-disciplinary finite element programs. Several tools for easy implementation programs and a common platform are provided. This framework requires the knowledge of C++ and PYTHON ™. The kernel has been coded using C++ ; the PYTHON ™ scripting language, on the other hand, works as an interface that is used to define the main procedures of KRATOS. This feature improves the flexibility of this framework when required. In Figure 15, it is possible to see a diagram of this framework during a common runtime.

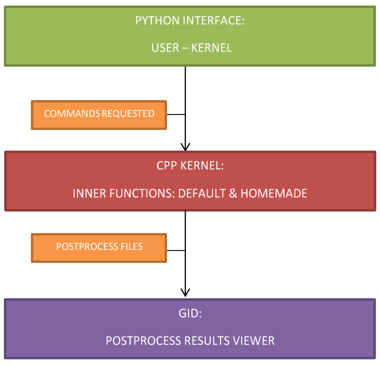
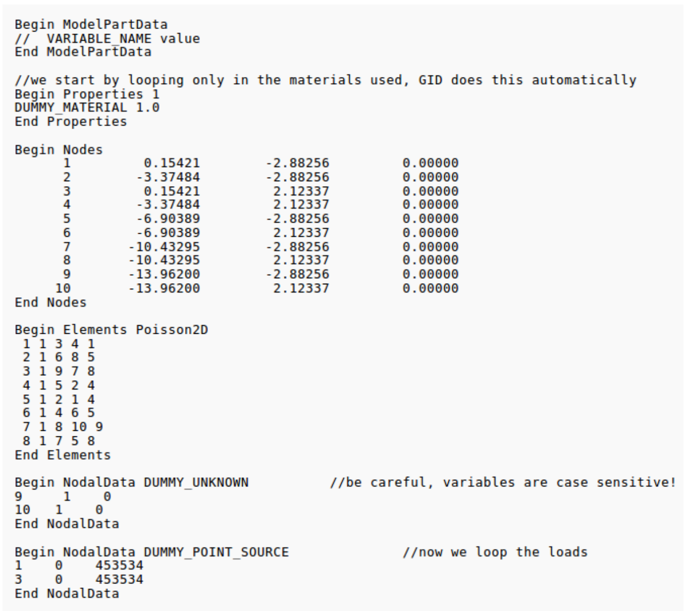


Figure 15: In this diagram it is shown the structure of a KRATOS simulation.

In order to launch a KRATOS problem, three basic files are required:

- *The geometry file.* This file has been encoded using a file extension called .mdpa, as it can be seen in Figure 16. Each file contains the header, where the element is defined; the nodes, which defines the 3D eulerian coordinates of the mesh; the connectivity, which defines the mesh connections between nodes; and the conditions, which defines the nodes that contain restrictions.
- *The materials file* consists in a PYTHON ™ script file that contains the definition of the materials used for this problem.
- *The main script file.* This is a PYTHON ™ file that contains all the functionalities required in order to run the KRATOS problem. This files have three main parts. The starting part is where the required modules are imported. In the second part, the solver of the problem and the postprocess files are configured. In the third part, the solver and the postprocess are called.



```
Begin ModelPartData
//  VARIABLE_NAME value
End ModelPartData

//we start by looping only in the materials used, GID does this automatically
Begin Properties 1
DUMMY_MATERIAL 1.0
End Properties

Begin Nodes
      1        0.15421       -2.88256        0.00000
      2       -3.37484       -2.88256        0.00000
      3        0.15421        2.12337        0.00000
      4       -3.37484        2.12337        0.00000
      5       -6.90389       -2.88256        0.00000
      6       -6.90389        2.12337        0.00000
      7      -10.43295       -2.88256        0.00000
      8      -10.43295        2.12337        0.00000
      9      -13.96200       -2.88256        0.00000
     10      -13.96200        2.12337        0.00000
End Nodes

Begin Elements Poisson2D
 1 1 3 4 1
 2 1 6 8 5
 3 1 9 7 8
 4 1 5 2 4
 5 1 2 1 4
 6 1 4 6 5
 7 1 8 10 9
 8 1 7 5 8
End Elements

Begin NodalData DUMMY_UNKNOWN         //be careful, variables are case sensitive!
9     1    0
10    1    0
End NodalData

Begin NodalData DUMMY_POINT_SOURCE             //now we loop the loads
1    0    453534
3    0    453534
End NodalData
```

Figure 16: Basic input file mdpa, it can be appreciated the nodes tag, the connectivity tag and the conditions tag. Image courtesy of [4].

4.1 Editing Kratos

As already mentioned, for a modal analysis in which Reissner-Mindlin Flat Shell Element is used, KRATOS will be employed for perform the FE analysis (determination of mass and stiffness matrices). The basic idea is that MATLAB ™ will call KRATOS. Then the framework will run the defined problem. At the end of the KRATOS simulation, the elemental matrices will be printed. Finally, once the printing process has ended, the original MATLAB ™ script will continue the execution by reading and using the information printed by KRATOS.

KRATOS is an open-source environment devoted to FEM analysis. This framework contains several modules, each one able to solve a different kinds of FEM problems. In this report, a LINUX version of this framework has been used. In this kind of machines each module is contained inside the directory *applications*, as can be seen in figure Figure 17. Unfortunately, KRATOS does not contain a function that allows to dump objects from the RAM memory during the simulation runtime. Therefore it becomes necessary to create a special function in C++ with this feature.

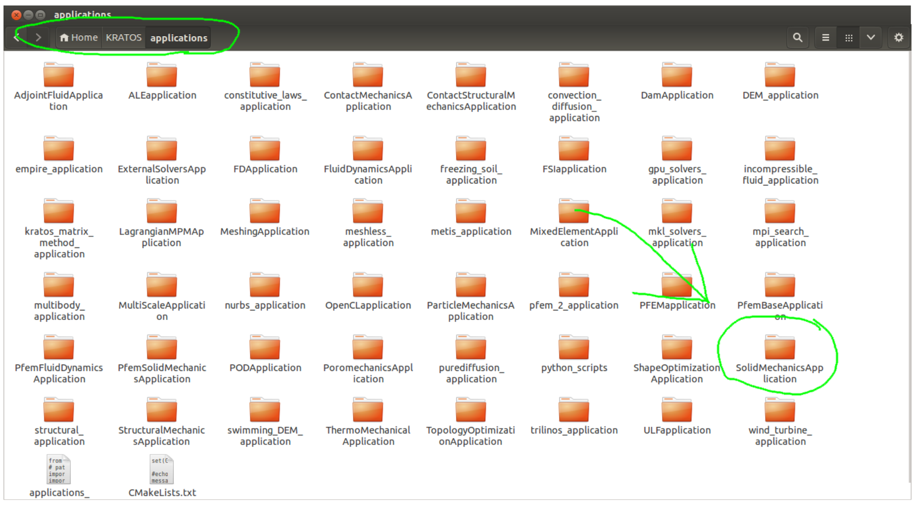


Figure 17: Applications folder inside the main KRATOS directory. In the upper part of the window, it can be seen the installation directory, whereas the arrow, also in green is pointing towards the *SolidMechanicsApplications* folder.

First of all, it is necessary to extract the Mass Matrix M and the Stiffness Matrix K. To do so, a new PYTHON ™ utility will be added inside the KRATOS folders. These extra utilities are called in the PYTHON ™ script that starts the application. The function will be implemented using the following recipe:

1. The following file will be modified: **KRATOS/applications/SolidMechanicsApplications/custom_python/add_custom_utilities_to_python.cpp**, as it can be seen in Figure 18, marked with green. This part will be in charge of defining the new python utility, and acts as some sort of interface that joins the PYTHON ™ layer with the C++ kernel code.

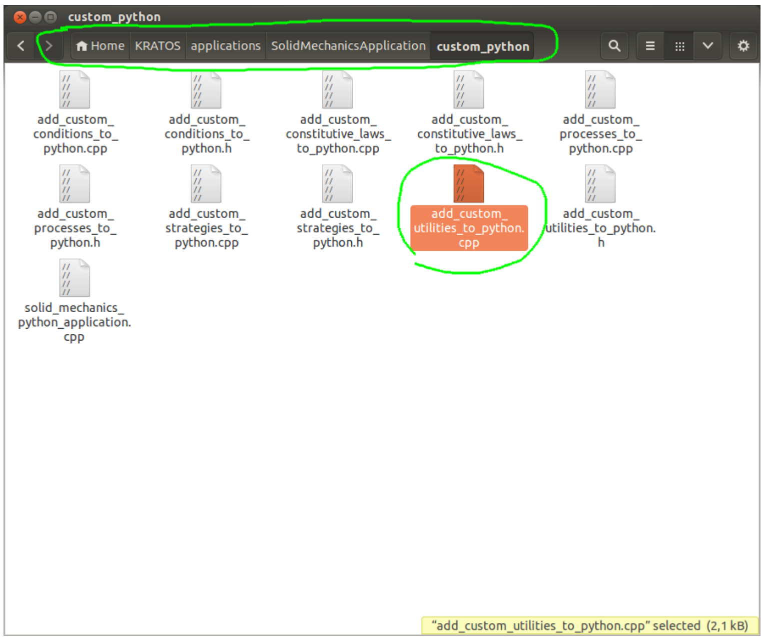

custom_python
Home
KRATOS
applications
SolidMechanicsApplication
custom_python
add_custom_conditions_to_python.cpp
add_custom_conditions_to_python.h
add_custom_constitutive_laws_to_python.cpp
add_custom_constitutive_laws_to_python.h
add_custom_processes_to_python.cpp
add_custom_processes_to_python.h
add_custom_strategies_to_python.cpp
add_custom_strategies_to_python.h
add_custom_utilities_to_python.cpp
add_custom_utilities_to_python.h
solid_mechanics_python_application.cpp
"add_custom_utilities_to_python.cpp" selected (2,1 kB)

Figure 18: Location of the file add_custom_utilities_to_python.cpp inside KRATOS directory.

2. Then the **KRATOS/applications/SolidMechanicsApplications/custom_utilities/print_matrix.h** is created, see Figure 19. In this part we include the C++ code that extracts and prints the elemental mass matrix and the elemental rigidity matrix for each element. Every data array computed is printed in an binary file.

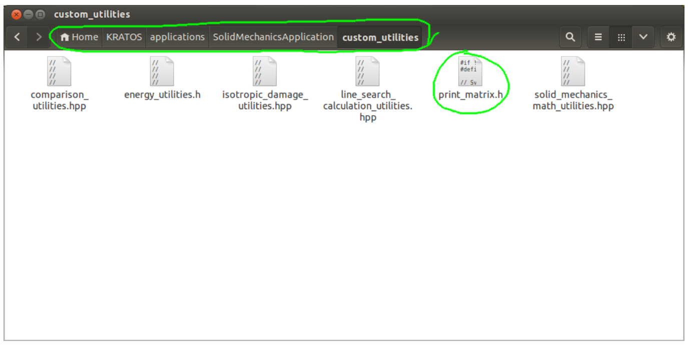

custom_utilities
Home
KRATOS
applications
SolidMechanicsApplication
custom_utilities
comparison_utilities.hpp
energy_utilities.h
isotropic_damage_utilities.hpp
line_search_calculation_utilities.hpp
print_matrix.h
solid_mechanics_math_utilities.hpp

Figure 19: Location of the file *print_matrix.h* inside KRATOS directory.

3. The final modification consists in the addition of the printing instructions inside the main PYTHON ™ script.

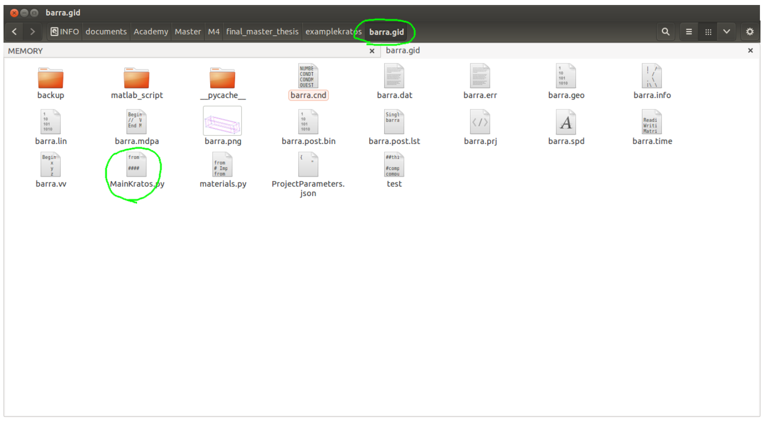


Figure 20: Application directory. The name of the application launcher is marked with green *MainKratos.py*.

KRATOS will be recompiled once the files have been successfully modified. Finally, the new functionality will be called from the main PYTHON ™ script.

4.2 Modified Code Structure

Once the custom utility has been edited, it is possible to run a problem composed by the Reissner-Mindlin Flat Shell Element . The modified code structure is presented in Figure 37. For the mentioned diagram, it has been decided to assign the same key scheme as in *Section 3.1* . In Figure 37, the following functionalities have been added:

1. **BLACK TAG** As it was commented in *Section 3.1* , it refers to decision points.
    (a) *KRATOS* This switch is used to enable the KRATOS process and to perform computations that require the use of data extracted from this framework.
    (b) *RECALMAT* This switch is used to avoid the KRATOS process. This switch is specially useful if a geometry has to be relaunched using the classic algorithm and the substructured algorithm in order to save time.
2. **BLUE TAG** It refers to the main functionalities run by the program, as it was referered in *Section 3.1* .
    (a) *KRATOS AUX FILES PRINTER.* This part is in charge of printing the necessary input files, defined in *Section 4.1* required to execute KRATOS.
3. **GREEN TAG** As it was defined in *Section 3.1* . This kind of tag refers to external software.
    (a) *LAUNCH KRATOS*, This part is in charge of launching the KRATOS framework.

It has to be commented that the original functions were also modified in order to make possible to read the Reissner-Mindlin Flat Shell Element . This modification consists in the addition of conditional sintax *if-else* in several key points of the code.

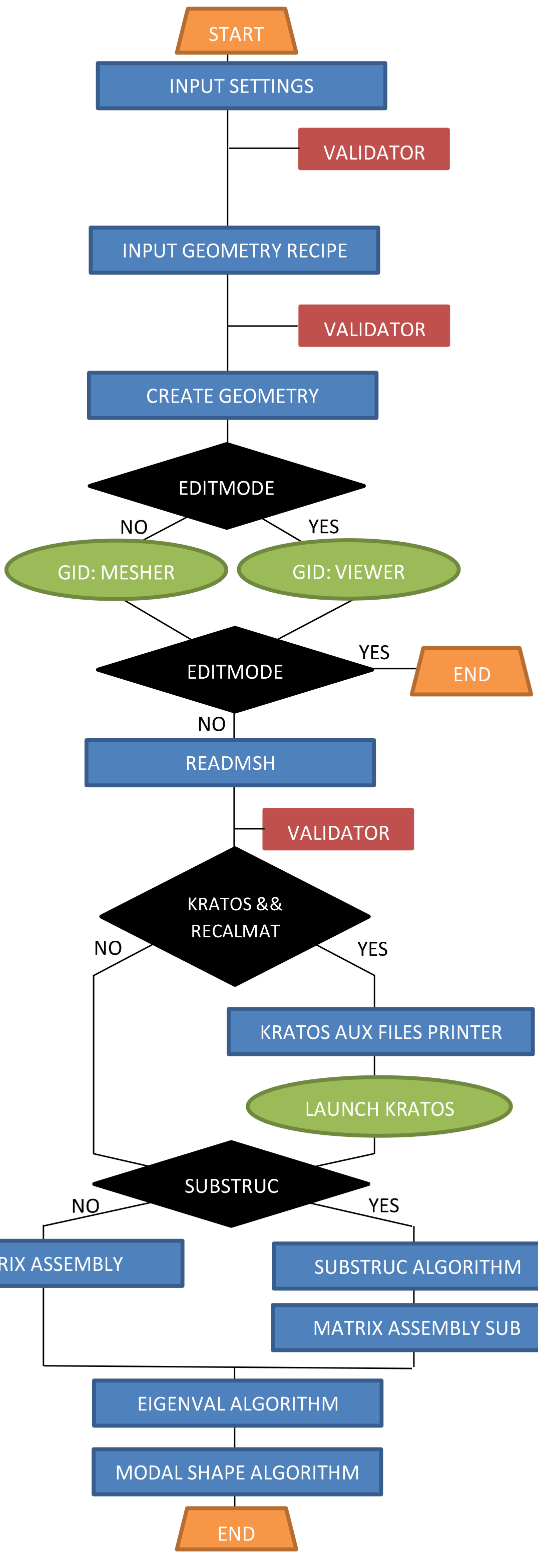


Figure 21: Substructural ALgorithm Diagram, including KRATOS module.

4.3 Reissner-Mindlin Flat Shell Element Problem Definition

For this problem the same analysis that was done in *Section 3.2* will be performed.

The chosen geometry corresponds to an approximation of the actual geometry of a Mustang P-51, displayed in Figure 22. Several parts have been removed, such as the propeller, the landing gear, the cockpit and the control surfaces. This decision was make in order to save editing time and computational effort.

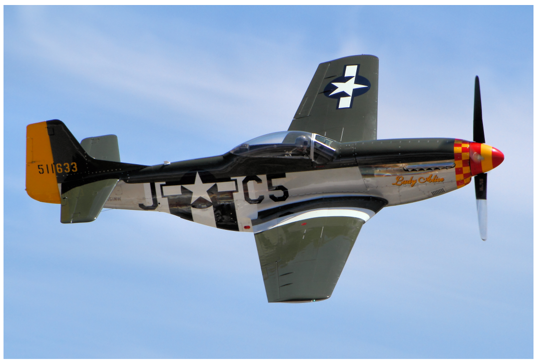


Figure 22: P51 mustang, chosen airplane for the second problem.

The geometric model has been created from scratch by using a GNU software called LIBRE CAD . Several representative points of the surface of the plane were extracted from the picture shown in Figure 22 by using such a software. The coordinates of such points were then imported into MATLAB ^{TM} by using a homemade script, and the final shape was drawn from such points using simple geometric transformations. The final result can be seen in Figure 23. For this problem, Reissner-Mindlin Flat Shell Element is also used. Based on the recommendation given in [15], [16] and [17], an Aluminium 2014 has been chosen; its properties are displayed in Table 4. It should be mentioned that the wings have been reduced into plates without considering nor the wings thickness neither the airfoils. Internal structure such as frames and stringer have been also washed out. To compensate for this lack of internal structure, a relatively high thickness has been set for all shell elements ($t = 23[mm]$). The restricted nodes will be defined similarly as in the previous problem, see Figure 24. Finally, the employed substructured model is depicted in Figure 25.

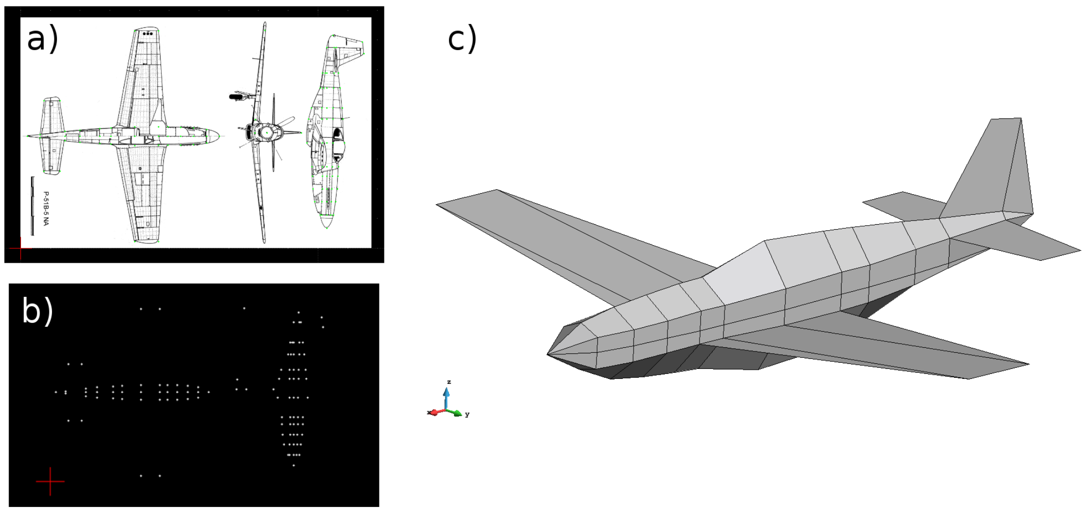


Figure 23: Image Base imported into LIBRE CAD *a)*. Points Extracted using the same software, *b)* later this points are converted into .svg format, which can be read using MATLAB ^{TM}. The final geometry is depicted in *c)*.

Table 4: Reissner-Mindlin Flat Shell Element properties

$E\,[MPa]$	ν	$t\,[mm]$	$\rho\,[kg/m^3]$
185	0.29	23.62	2795.6704

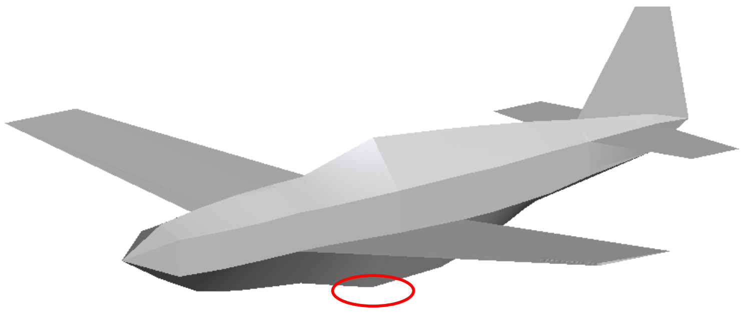


Figure 24: Restricted nodes chosen for this problem are marked in red —the ones with lowest Z coordinate.

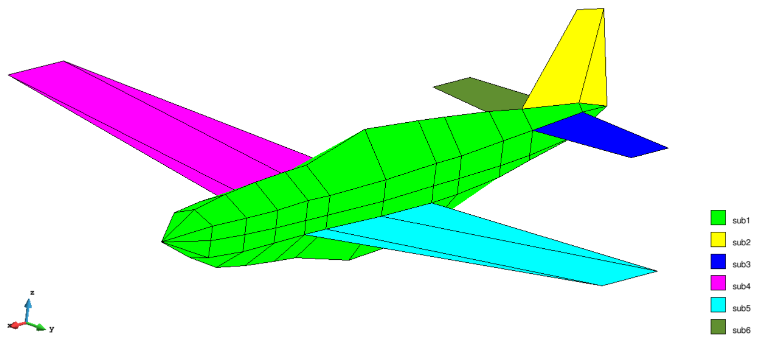


Figure 25: Substructured model chosen for this problem. Each color is identified with a different substructure.

4.4 Results extracted for a Reissner-Mindlin Flat Shell Element

For this problem the inner DOFs have been reduced to 6, in Table 5, the natural frequencies for each defined substructure are presented. Also the 3 lowest modal shapes for each substructure are presented in Figure 26, Figure 27 and Figure 28. The tenth lowest natural frequencies for the total analysis are presented in Table 6, while the 4 lowest modal shapes are presented in Figure 29, Figure 30, Figure 31 and Figure 32.

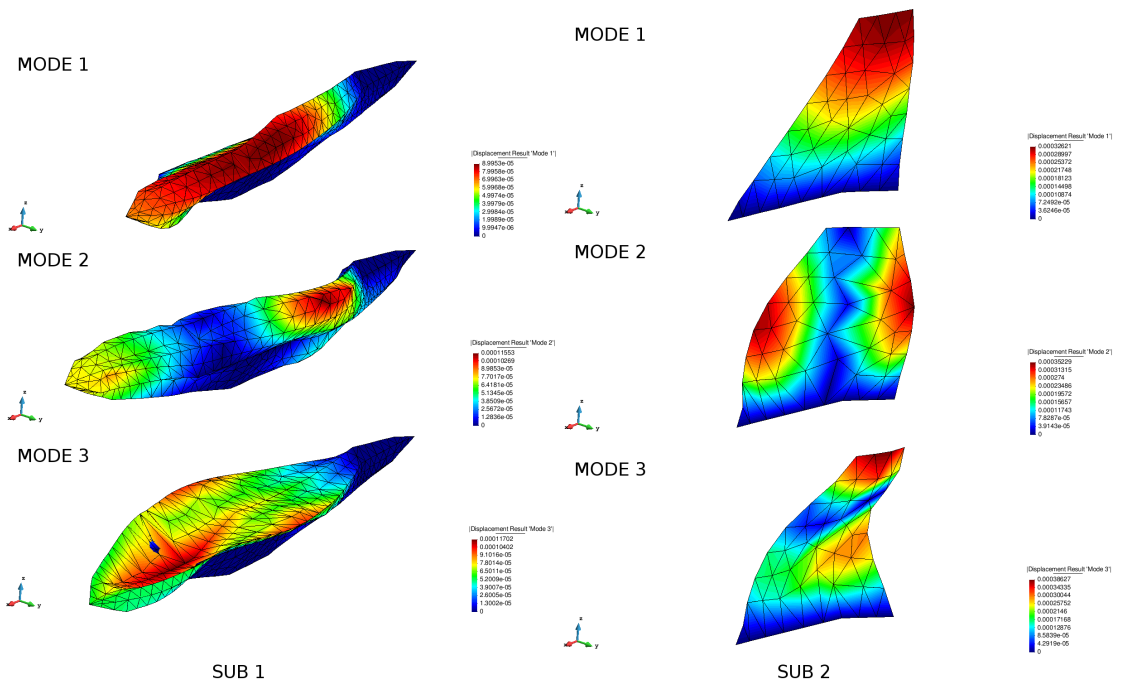


Figure 26: Modal Shapes Obtained for the first and second substructures. Recall that boundary DOF's are considered as imposed DOFs.

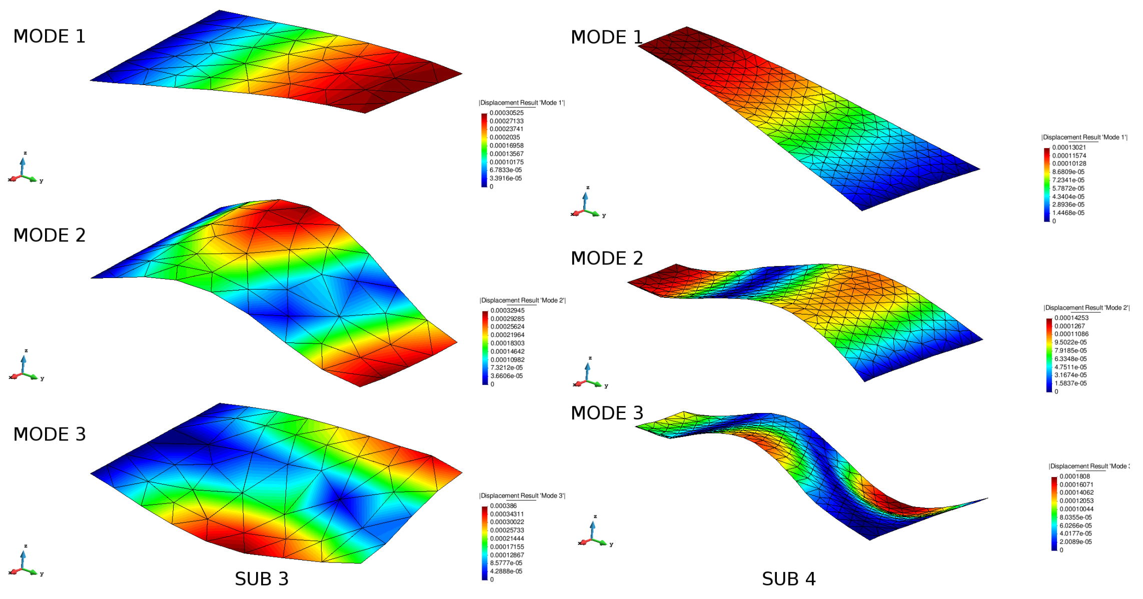


Figure 27: Modal Shapes Obtained for the third and fourth substructures.

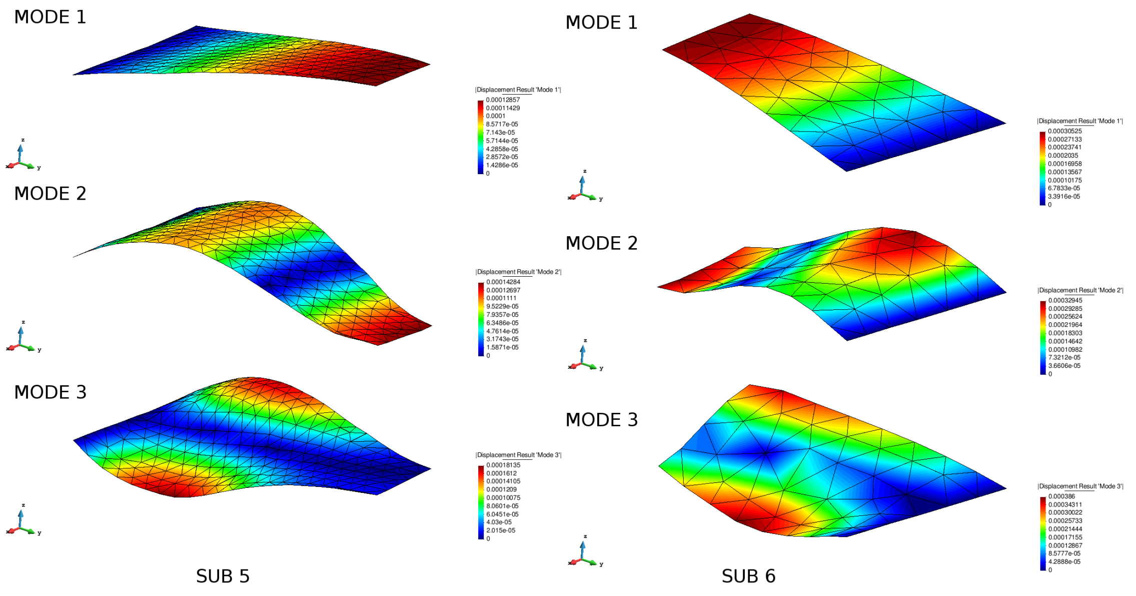


Figure 28: Modal Shapes Obtained for the fifth and sixth substructures.

Table 5: Modal frequencies of each substructure.

Freq [Hz]	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6
sub 1	1.048260	1.713267	1.795929	2.051887	2.119760	2.142968
sub 2	0.848251	1.967279	2.096361	2.837432	3.129295	3.348210
sub 3	0.692263	1.840554	2.112059	2.724539	2.945360	3.481899
sub 4	0.240465	0.627079	1.019696	1.034580	1.311245	1.427693
sub 5	0.237343	0.628036	1.016853	1.027050	1.300488	1.430072
sub 6	0.692263	1.840554	2.112059	2.724539	2.945360	3.481899

Table 6: Natural Frecuecies for the shell problem, CA refers to Classic Algorithm, whereas CB refers to Craig-Bampton algorithm.

Freq $[Hz]$	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9	ω_{10}
CA	0.2306	0.2347	0.3720	0.5363	0.6200	0.6335	0.6873	0.7247	0.7341	0.8455
CB	0.2306	0.2347	0.3732	0.5386	0.6201	0.6337	0.6873	0.7315	0.7443	0.8931

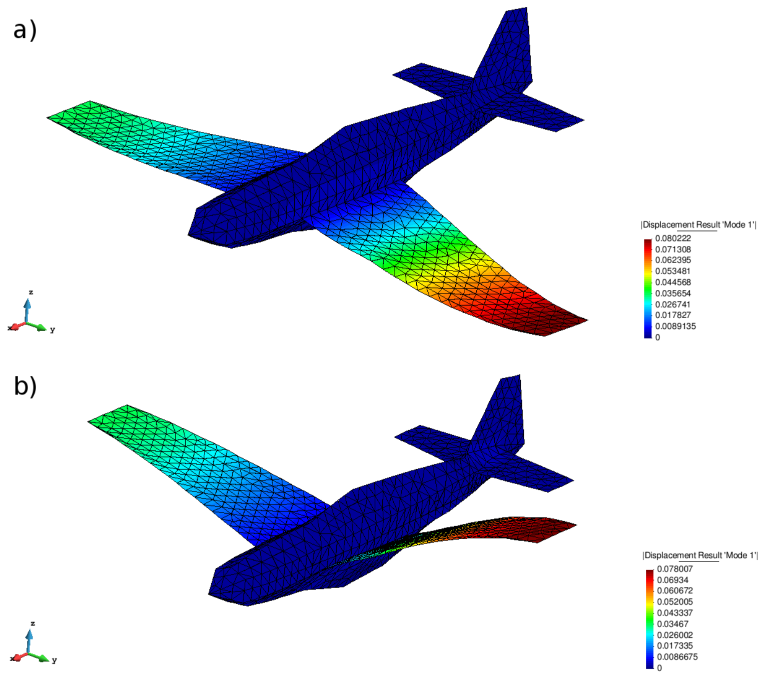


Figure 29: First Modal Shape, a) refers to the classic algorithm: CA, b) refers to the substructured algorithm: CB.

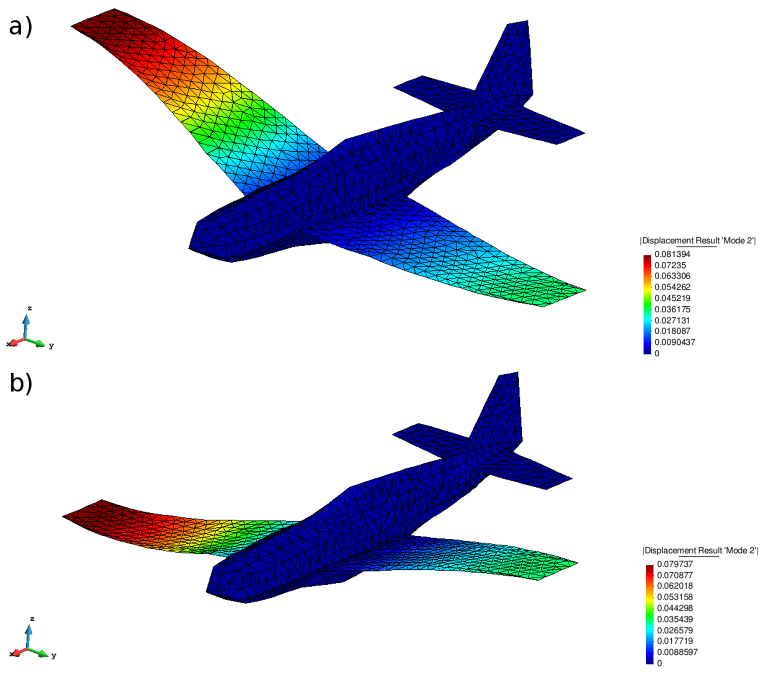


Figure 30: Second Modal Shape, a) refers to the classic algorithm: CA, b) refers to the substructured algorithm: CB.

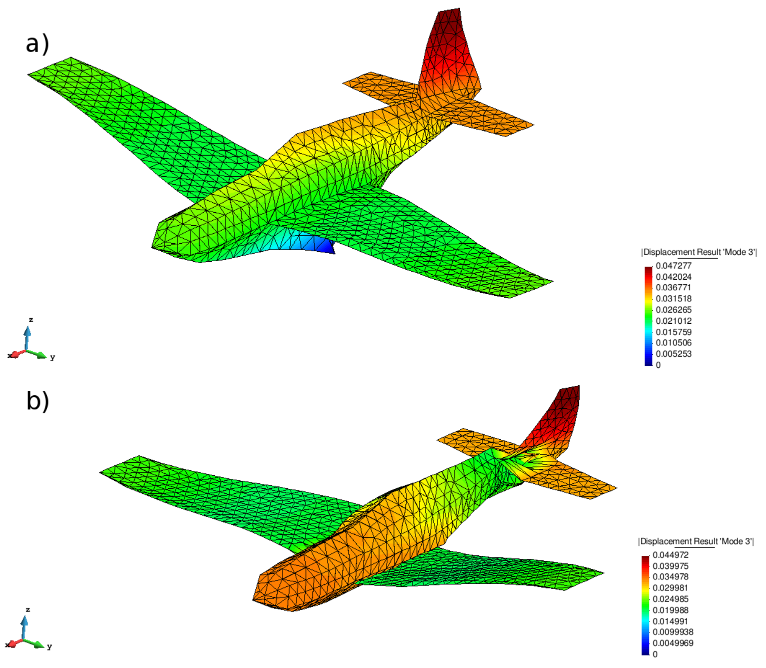


Figure 31: Third Modal Shape, a) refers to the classic algorithm: CA, b) refers to the substructured algorithm: CB.

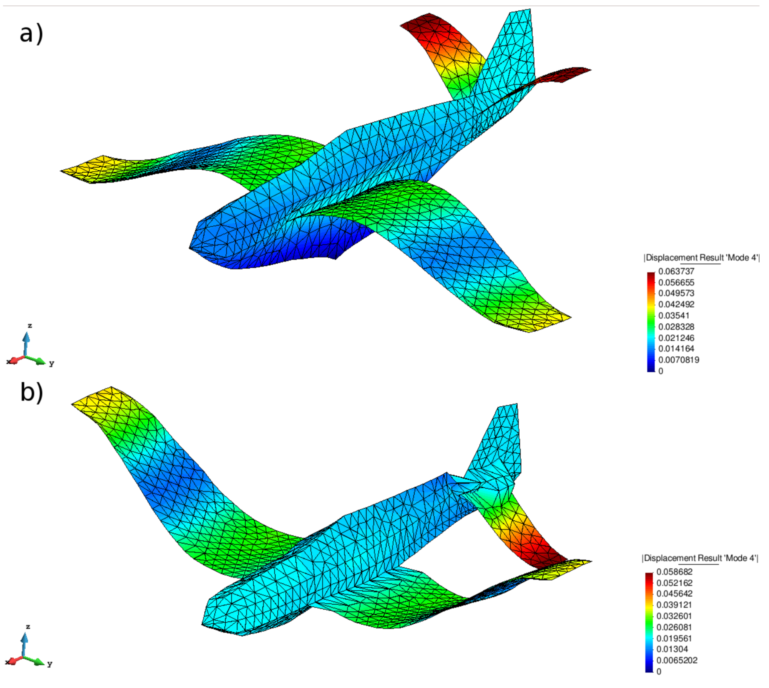


Figure 32: Fourth Modal Shape, a) refers to the classic algorithm: CA, b) refers to the substructured algorithm: CB.

4.5 Discussion of results

Inspection of the results shown in the foregoing reveals that, as expected, the natural frequencies computed by the CA and the CB analyses are quite similar (errors below 1 % are obtained for the first six modes). Concerning the deformed shapes associated to each mode, the only significant difference is detected in the third mode, see Figure 13 (wing tail). This discrepancy can be attributed to a simple decompression error, among others. Unfortunately, our analysis fails to reveal the actual reason for such a discrepancy.

5 Application for a Complete Aircraft Configuration

5.1 Chosing the Geometry

The aim of this section is to apply the methodology explained in the foregoing section (in which KRATOS and MATLAB are interconnected to automatically generate the required natural frequencies and associated modes) to study the vibration behavior of an entire aircraft structure. To minimize meshing problems, the geometry has been streamlined in several places of the airframe, as explained later in *Section 5.3.1* . In the following, we describe the criterion used to guide the choice of the final geometry-.

- The geometry should be as close as possible to a conventional commercial airplane, because the specifications are easily accessible —in contrast to military aircrafts .
- The geometric model should contain the basic elements present in a conventional airplane.

Excessively detailed elements such as the landing gear, the engines, lights, control surfaces and pneumatic system will be simply neglected.

- The geometry must have two differentiated parts: the inner structure (frames, stringers, spars, ribs ...) and any kind of structural element present in a conventional aircraft. The shell will be formed by the fuselage, the wing tail the stabilizer, and the engine fairings.

5.1.1 Kinds of Models

Several alternatives were considered when the project started. More than 10 geometries were selected in order to perform the study. In the following section, the most suitable models will be exposed. The selected geometries can be divided basically into three main kinds of CAD files:

- The first kind of geometries consist in classic propeller airplanes. Although there is a huge amount of models available in the Internet, it has to be considered that these geometries tend to contain a huge amount of unnecessary details. Also this kind of airplanes tend to be old military models.

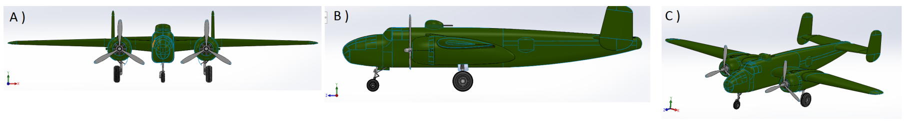


Figure 33: First kind of airplanes, classic models A), side view B), profile view C) top view

- The second kind of models refer to modern militar jet models. As the previous kind of airplanes, there is also a lot of available material. Nevertheless, this kind of aircrafts have not a trivial structure compared with commercial models. In fact, in this kind of designs, the internal structure is one of the most secret parts.

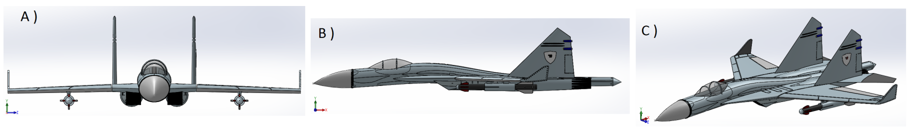


Figure 34: Second kind of airplanes, militar models A), side view B), profile view C) top view

- Finally the third kind of models consist in commercial airplanes. These are the preferred models for this project, because as mentioned above, specifications are easier to obtain.

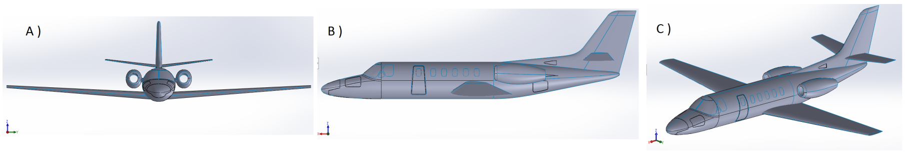


Figure 35: Third kind of airplanes, private aviation A), side view B), profile view C) top view

5.1.2 Chosen model

Following the guidelines provided in *Section 5.1.1* , we choose the third kind of airplane. Specifically, the selected model is a Cessna Citation, a jet commonly used for private aviation. The specifications for this model are shown in Table 7 and correspond to a Cessna Citation XLS model, which is the newest version of this airplane. This updated version contains new details such as different wing configuration, and minor modifications in the rear fuselage. This data have been extracted from [6]. In Figure 36, a representation of the chosen airplane can be seen.

Figure 36: Cessna Citation. More information of this product can be found in [5].

Table 7: Cessna Citations XLS Specifications, data courtesy of [6]

Exterior Dimensions		
Wingspan	56 ft 4 in	17.17 m
Length	52 ft 6 in	16.00 m
Height	17 ft 2 in	5.23 m
Interior Dimensions		
Cabin Height	68 in	1.73 m
Cabin Width	66 in	1.68 m
Cabin Length	18 ft 6 in	5.64 m
Baggage Capacity	800 lb	362.9 kg
Weights		
Maximum Takeoff	20200 lb	9163 kg
Basic Operating Weight	12860 lb	5833 kg
Useful Load	7540 lb	3420 kg
Performance		
Takeoff Field Length (MTOW)	3560 ft	1085 m
Time to Climb	FL 450 in 29 min	
Max Cruise Speed	441 ktas	817 km/h
Max Range (Ferry, LRC)	2100 nm	3889 km

5.2 Final Code Structure

Once all the design concepts have been established, a last code modification is necessary to manage the external geometry introduced by the user (see BLACK TAG in *Section 37*). The only condition that the imported geometry has to meet is that its finite element mesh should be furnished in ASCII format and written using GiD ^{TM} sintax.

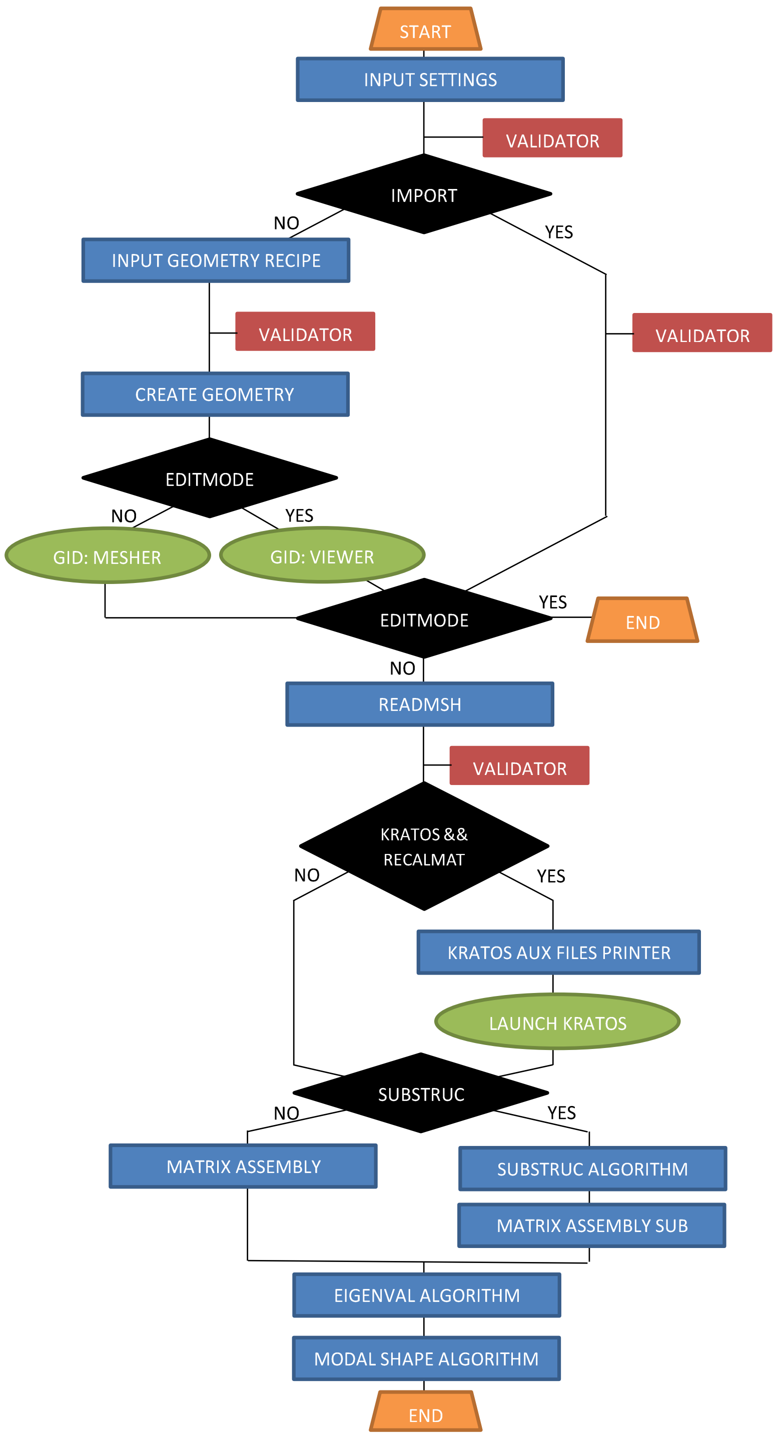


Figure 37: Final code structure

5.3 Complete Airplane Problem Definition

5.3.1 Design Process

In this section, we explain in more detail how we designed the final geometric model employed in the simulations. Firstly, the original geometry was extracted from the website [5]. However, this original geometry came with no specifications concerning the inner structure —only the outer shell geometry was available. In view of such circumstances, we were forced to edit the original outer geometry and incorporate a consistent (yet approximate) inner structure. We used SOLID WORKS ^{TM} for this task, and the design of the approximate inner structure were guided by specifications extracted from [18]. For instance, the design of the wing structure were inspired in a scheme contained in the Airbus A-220 guide (see Figure 39).

To simplify model edition, the structure was divided into several parts. Such a division served later as the substructuring required for model order reduction. The final sketch of the partitioning is displayed in Figure 38. It is tacitly assumed that each part includes both shell and structure, as described below:

- *Green Part*: It contains the entire wing of the airplane
- *Orange Part*: It contains the fuselage structure
- *Pink Part*: It represents the tail structure (Horizontal Stabilizer)
- *Grey Part*: It includes the Vertical Stabilizers
- *Blue Part*: It consists in the Engine Fairings.

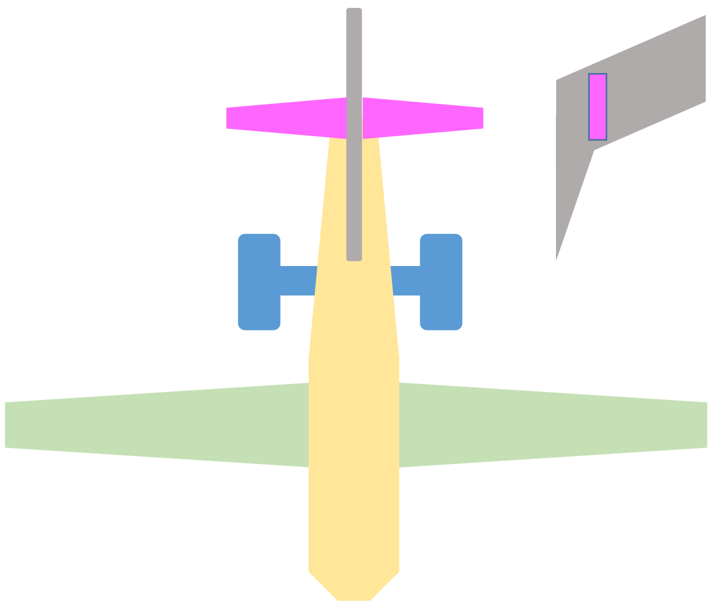


Figure 38: Airplane scheme. Five main parts can be distinguished: wings in *Green*, engine structure and fairing in *Blue*, tail in *Pink*, Stabilizers in *Grey*.

Several iterations were done before obtaining a basic structure and a more simplistic model of the mesh. In what follows, we describe more in detail each of the employed partitions.

Green Part

The green part represents the wings. The resulting model has a front spar and a rear spar plus ribs, see Figure 39.

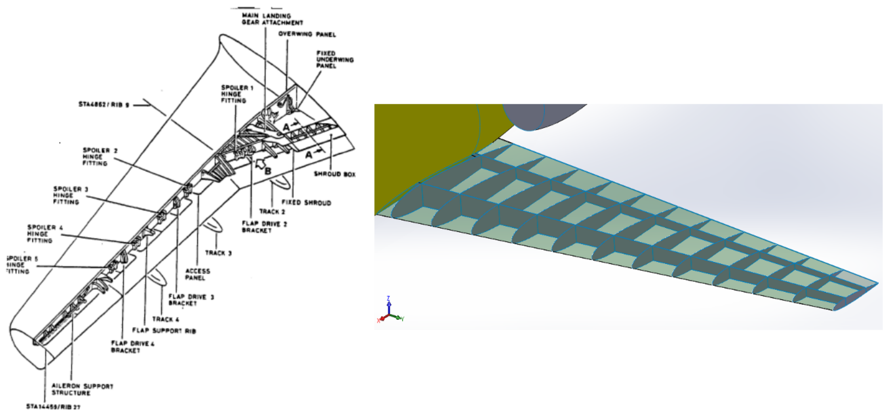


Figure 39: Wing structure: on the RIGHT, scheme structure extracted from the Airbus A-220 guide: on the LEFT, designed wing structure. Notice that no control surface is included.

Orange Part

As it has been commented before, the orange part is the fuselage. The first, tentative design consisted in:

- *Frames*: Entire solid pieces were created, with some simplifying assumptions such as considering the fuselage as a cylinder, as can be seen in Figure 40.
- *Stringers*. The solid stringers were generated by extrusion from two cross section profiles plus a guide curve, see Figure 41. A T profile was used for the beam designs, which is a common choice for standard airliners, see Figure 41. The final result is shown in Figure 41.

However, due to meshing problems in the interfaces, as well as in intricate places, the design described above was partially abandoned in favor of a more simplified model —one in which the structure and the shell are represented by the same geometry. In this design, the fuselage is divided into several parts in order to capture the cone and nose shapes, as can be appreciated in Figure 42. Also, in order to add an extra of rigidity, the wing torsion box, and the fairing torsion box were created and included in the inner structure.

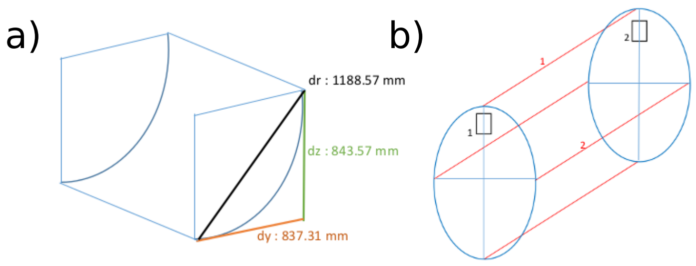


Figure 40: Scheme of the fuselage approximation into a cylinder: a) Scheme of the stringer creation used in the geometry design. b) Stringers Cross Sections in BLACK, Guide Curves in RED, Frames in BLUE.

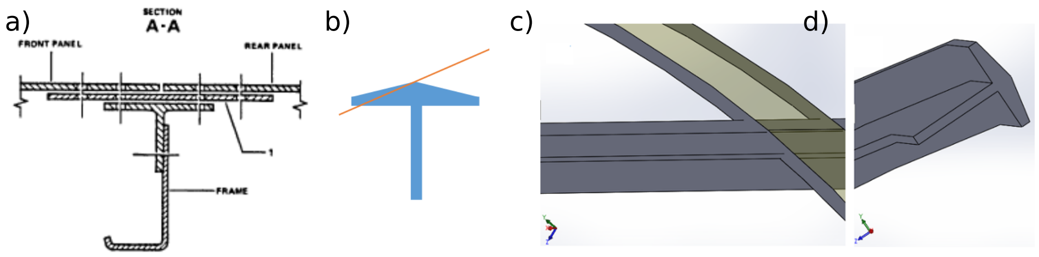


Figure 41: Stringer sections designed: a) Original section, based upon a common stringer profile extracted from a standard airliner. b) Chosen Approximation (T section with pointing upper side). c) Extruded shape applied in the model, with the union between the stringer and the frames. d) Cross section T.

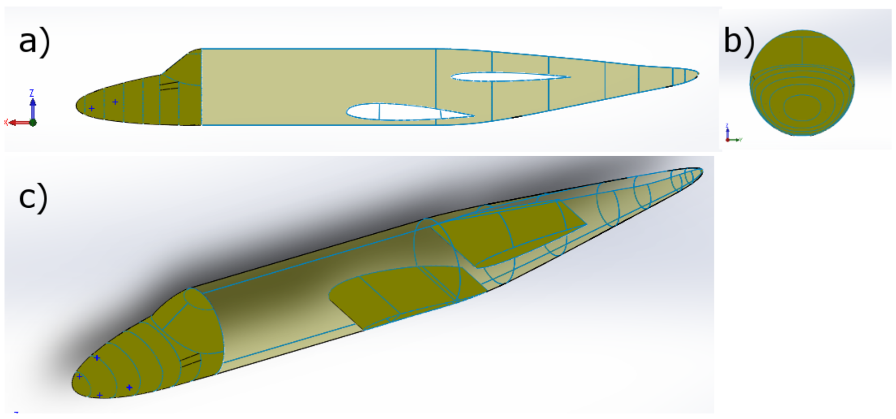


Figure 42: a) Profile of the fuselage, b) Front view of the fuselage, c) Isometric view.

Pink Part

This part refers to the Horizontal Stabilizer. The same scheme of *Section 5.3.1* is applied. First of all, the surface surrounding the tail is created, then the surfaces that define the front and rear spar, plus the ribs are drawn. The final result of this part can be seen in Figure 43.

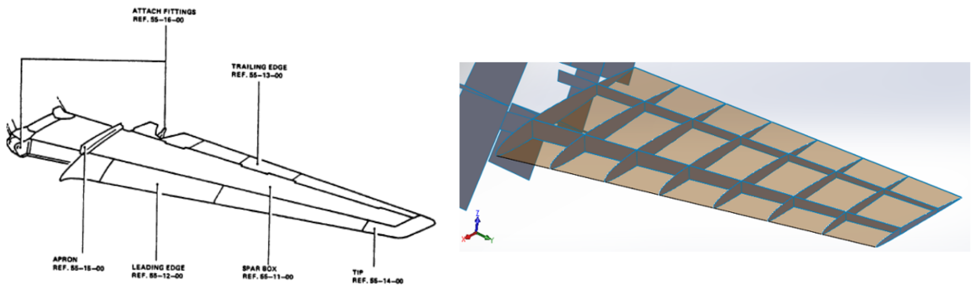

ATTACH FITTINGS REF. 55-16-00
TRAILING EDGE REF. 55-13-00
APRON REF. 55-15-00
LEADING EDGE REF. 55-12-00
SPAR BOX REF. 55-11-00
TIP REF. 55-14-00

Figure 43: Horizontal Tail Structure: on the LEFT scheme structure extracted from the Airbus A-220 guide, on the RIGHT, designed tail structure (control surfaces have been neglected).

Grey Part

This part represent the vertical stabilizer, it will be solved similarly to *Section 5.3.1* and *Section 5.3.1* . The same process givse as a result the solid shown in Figure 44.

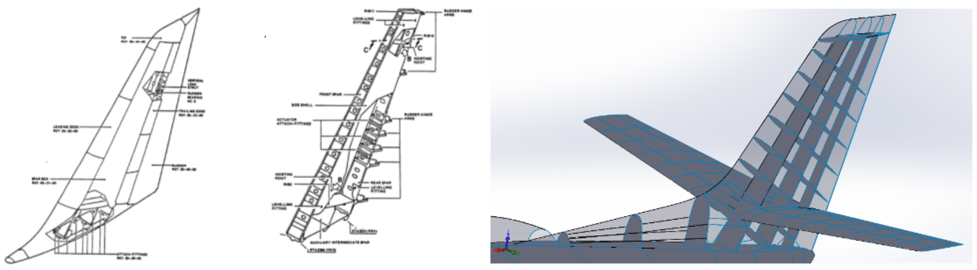


Figure 44: Vertical Stabilizer: on the LEFT, scheme structure extracted from the Airbus A-220 guide, on the RIGHT, designed vertical stabilizer structure, control surfaces such as the rudder have been neglected.

Blue Part

The blue part is related to the engine fairings and the pylons. As it was done with the lifting surfaces, a surrounding surface was created from the primitive geometry, and then, using special surface operations, we generated the structure of this part (see Figure 45).

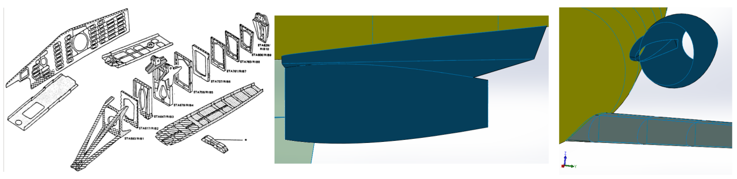


Figure 45: Fairing Structure and Pylon: on the LEFT, schematic structure of the pylons extracted from the Airbus A-220. On the RIGHT, representation of the model created with SOLID WORKS TM.

5.3.2 Design Problems

Several problems appeared when trying to generate the mesh (using the commercial software GID). This kind of problems appeared in excessively detailed parts present in the geometry. In order to fix those problems, geometry simplifications were introduced (see Figure **??**).

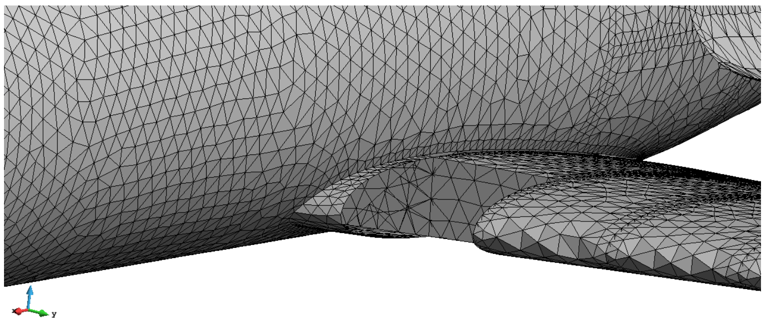


Figure 46: Union between the wing and the fuselage. Mesh compatibility were achieved after several iterations (geometry modifications).

5.3.3 Final Geometry Design

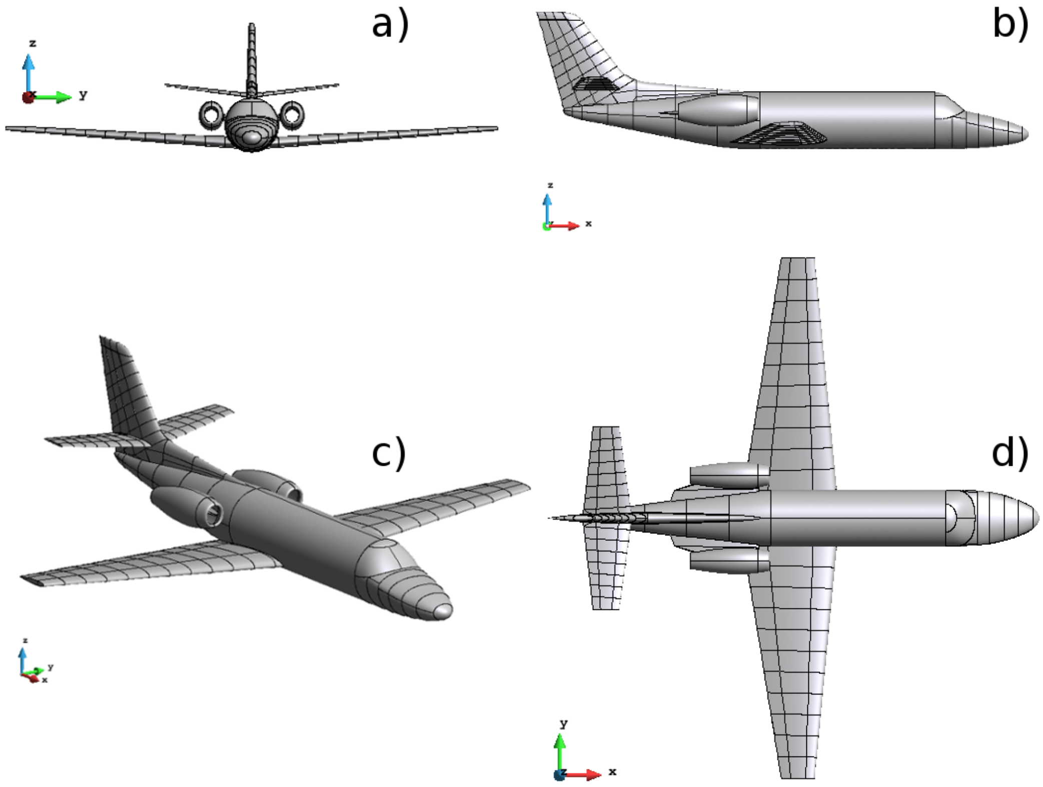


Figure 47: Main views of the final model.

The final design is shown in Figure 47. Material properties are summarized in Table 8. Notice that, given the lack of information concerning the exact geometry of the airplane —and also because of simplicity reasons—, the thickness of all components is considered equal. The distinct substructures employed in modal analysis are sketched in Figure 49.

Table 8: Material properties.

$E\,[MPa]$	ν	$t\,[mm]$	$\rho\,[kg/m^3]$
185	0.29	12	2795.6704

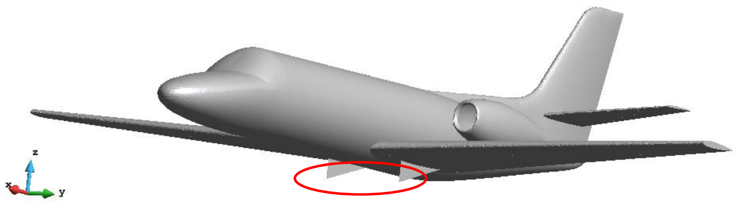


Figure 48: Dirichlet boundary conditions.

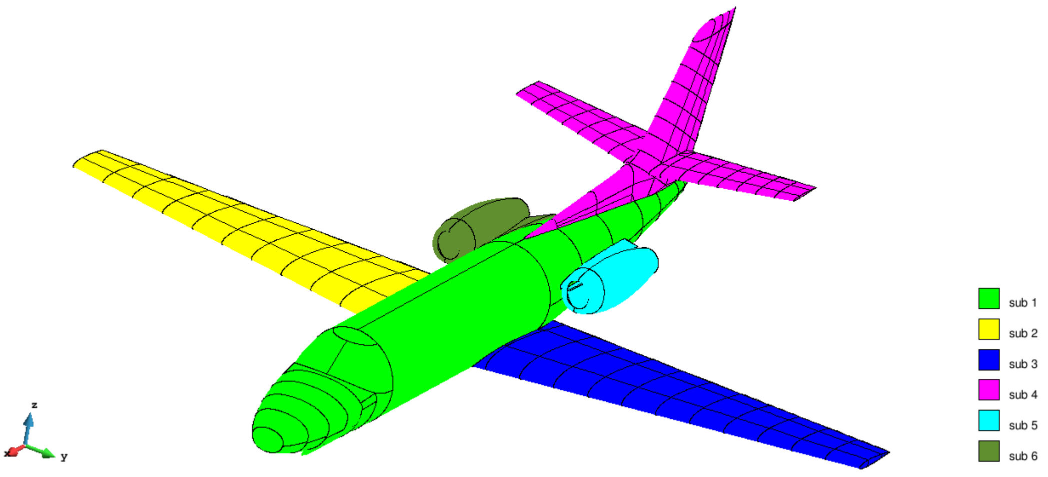


Figure 49: Structure partition.

5.4 Results

As in the previous examples, the number of inner DOFs have been reduced to 6. In table 9, the natural frequencies for each defined substructure are presented. On the other hand, the 3 lowest modal shapes of each substructure are presented in Figure 50, Figure 51 and Figure 52, and the 10 lowest natural frequencies for the total analysis are shown in Table 10. Finally, the 4 lowest modal shapes are displayed in Figure 53, Figure 54, Figure 55 and Figure 56.

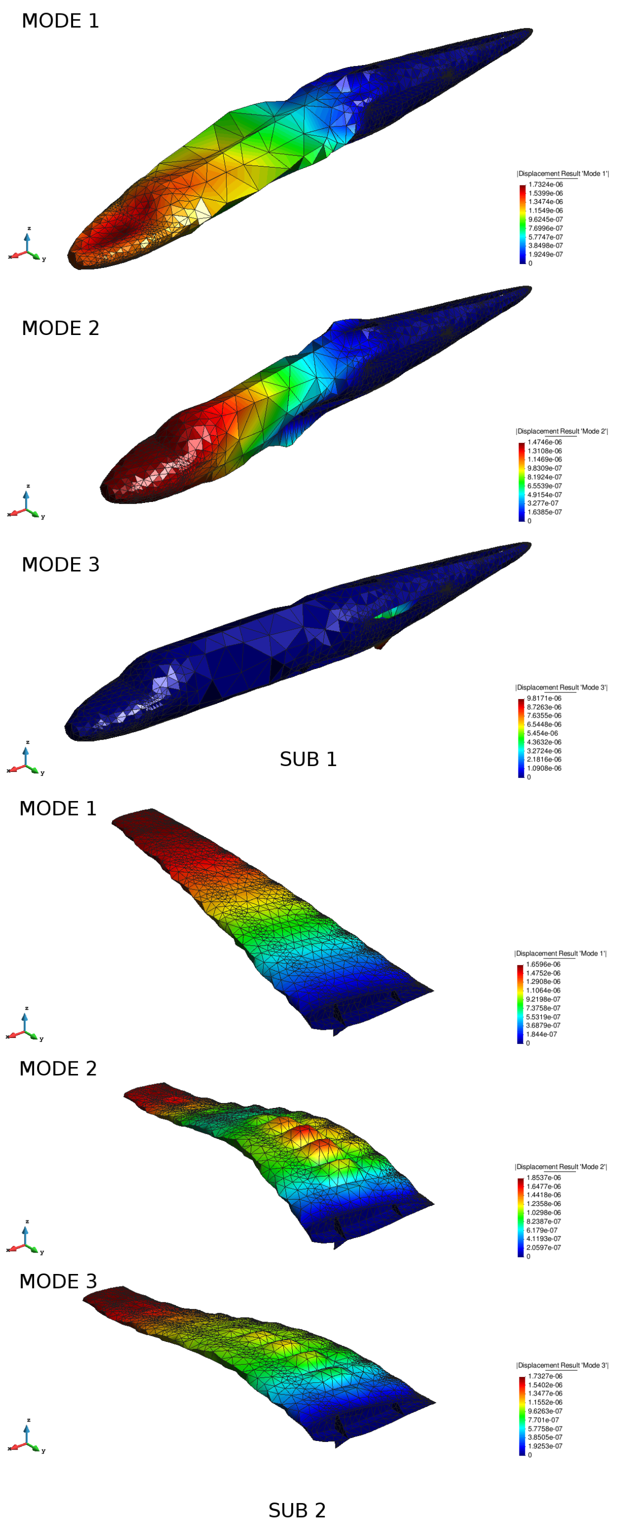


Figure 50: Modal Shapes for the first, and the second substructures.

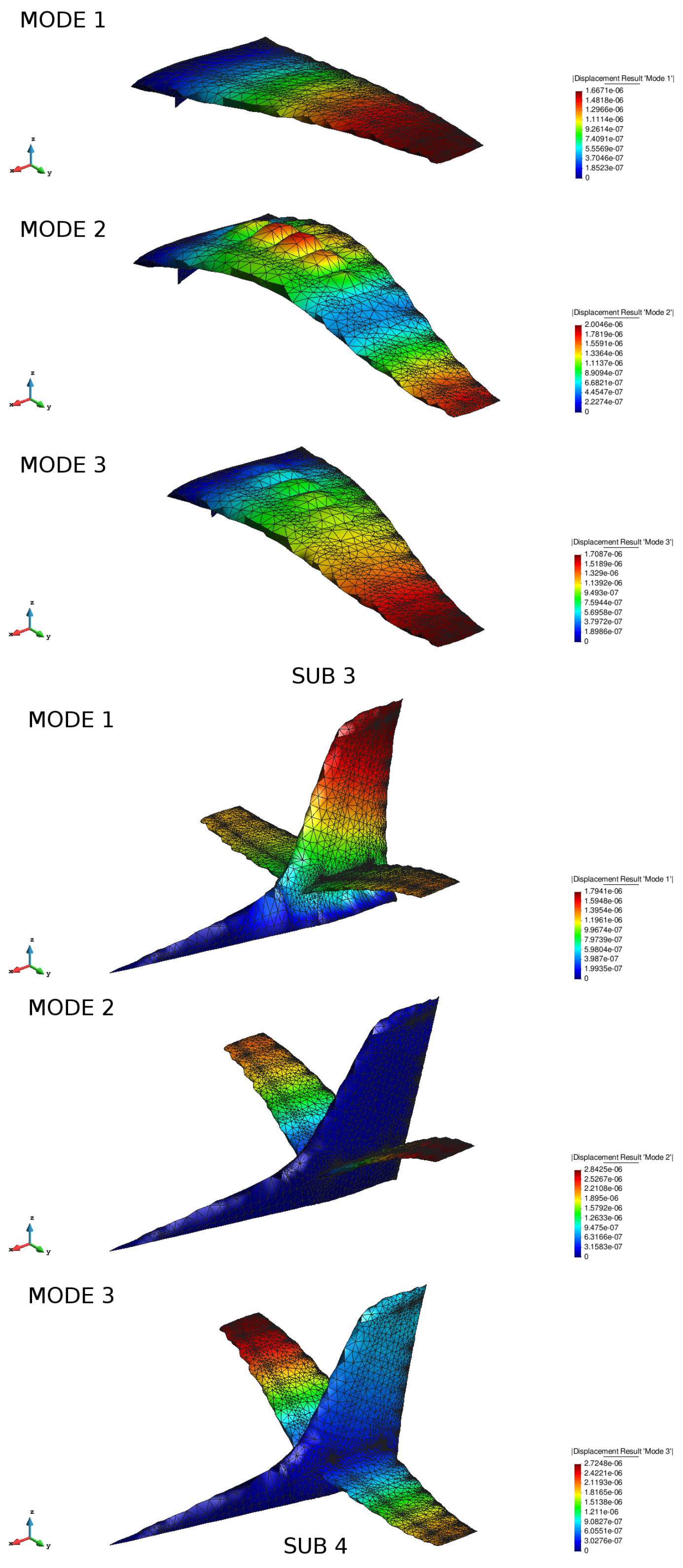


Figure 51: Modal Shapes for the third and fourth substructures.

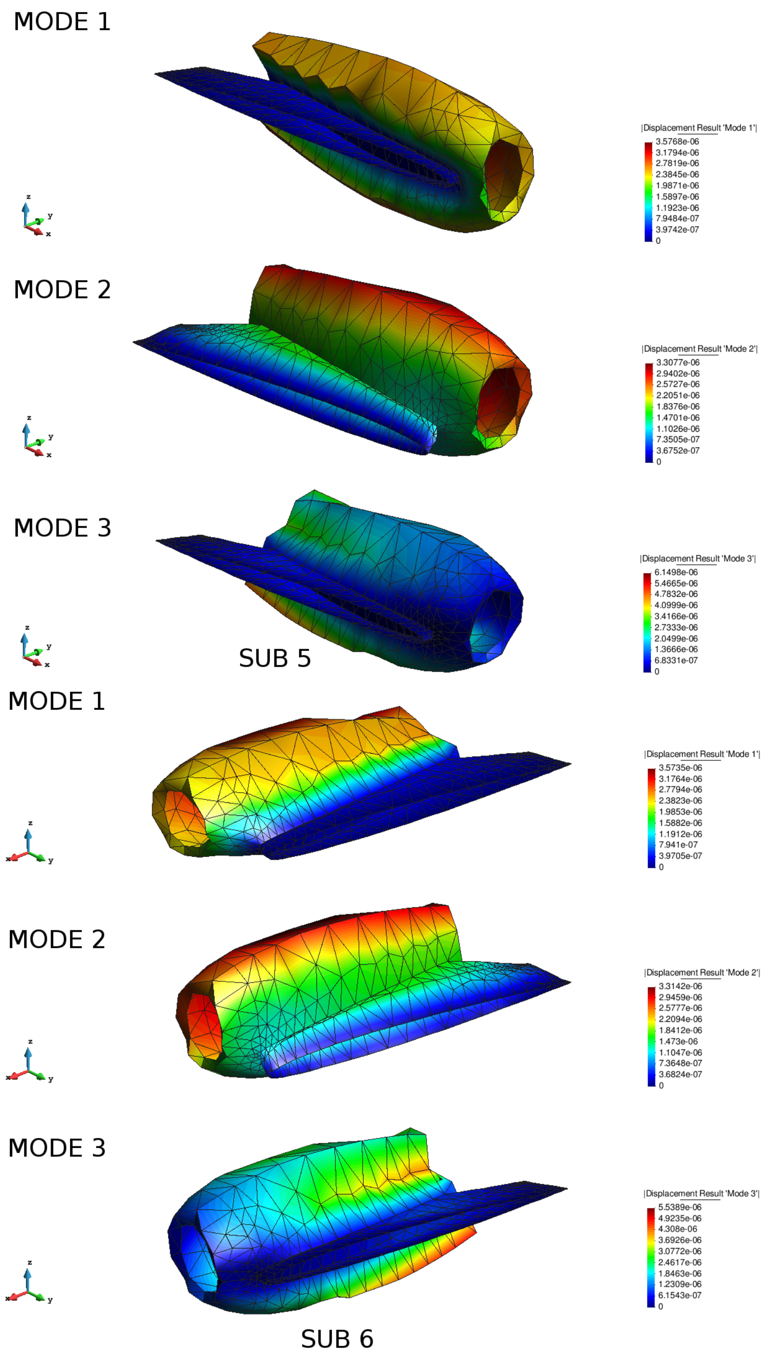


Figure 52: Modal Shapes for the fifth and sixth substructures.

Table 9: Natural frequencies for each substructure

Freq $[10^{-2}Hz]$	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6
sub 1	3.2843	3.2894	3.4046	3.6208	3.8247	4.2056
sub 2	1.7606	4.4526	4.4903	6.4212	6.5100	6.9598
sub 3	1.7506	4.4178	4.4790	6.4196	6.5049	6.9629
sub 4	4.2693	4.5386	4.6705	7.9333	8.9175	11.4050
sub 5	5.9430	6.3922	10.9259	13.2915	14.0920	14.5158
sub 6	5.8864	6.3430	11.1065	13.0727	13.8760	14.6839

Table 10: Natural frequencies. CA refers to Classic Algorithm, whereas CB refers to the Craig-Bampton algorithm.

Freq $[10^{-2}Hz]$	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9	ω_{10}
CA	0.2212	1.1885	1.4333	1.6685	1.7732	1.8537	2.7734	2.8878	3.3185	3.3326
CB	0.2212	1.1870	1.4284	1.6682	1.7709	1.8507	2.7607	2.8739	3.3122	3.3258

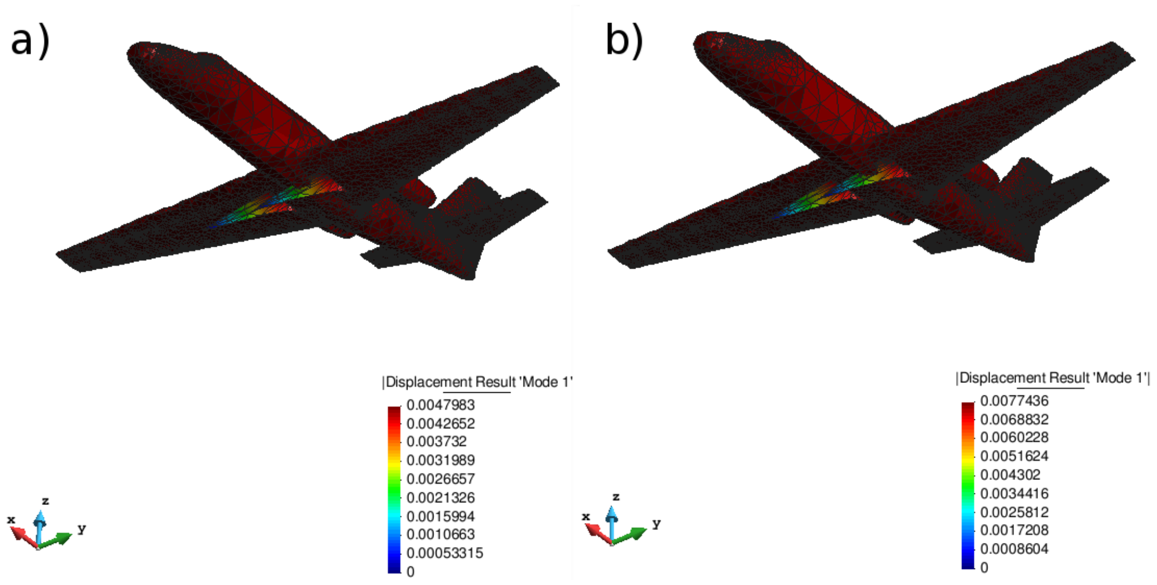


Figure 53: First Modal Shape

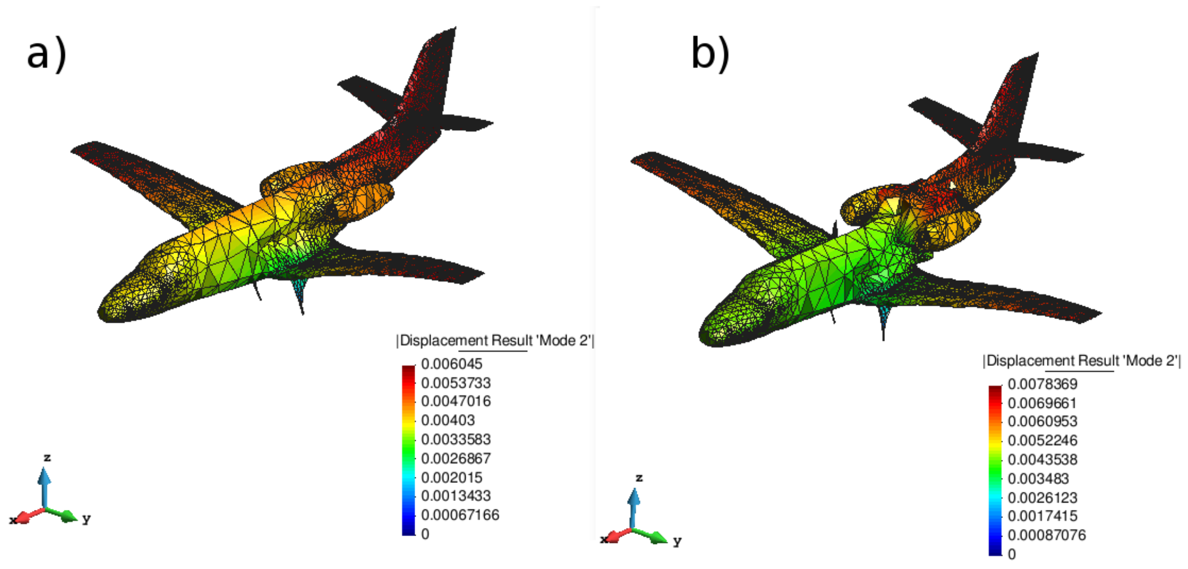


Figure 54: Second Modal Shape

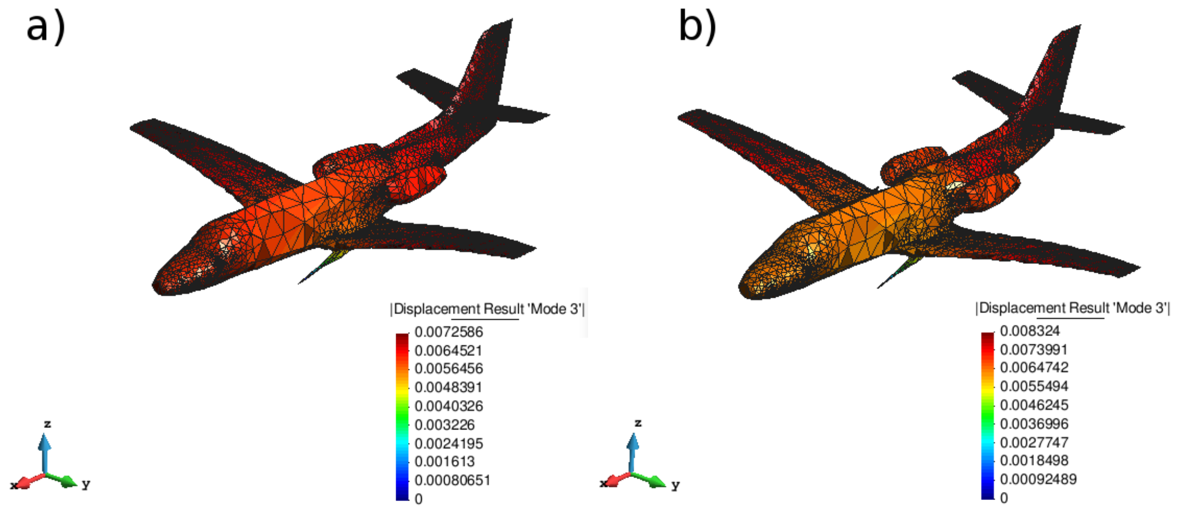


Figure 55: Third Modal Shape

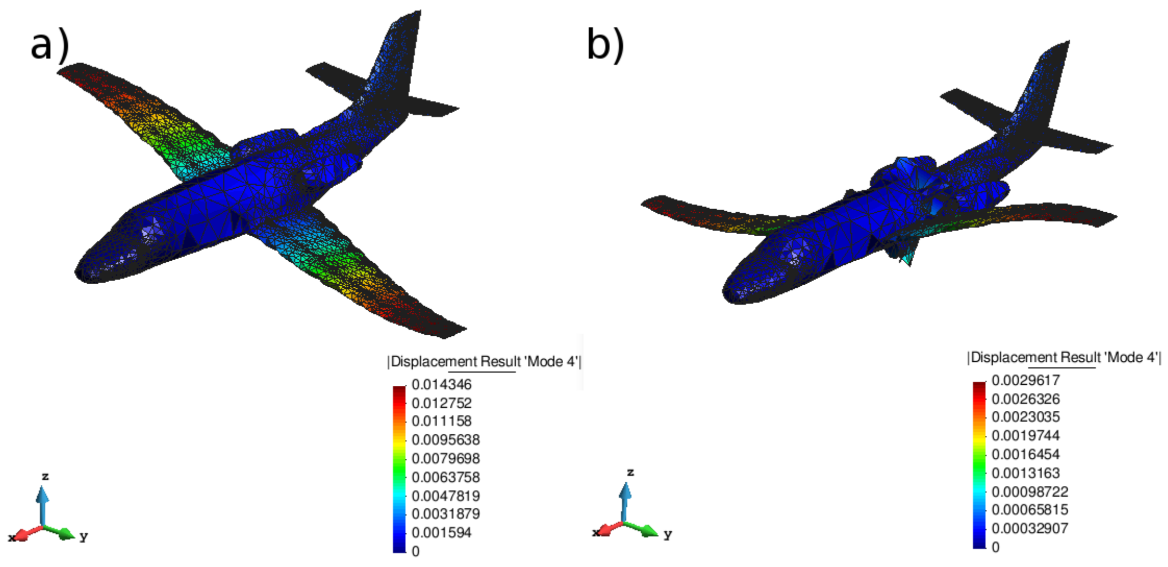


Figure 56: Fourth Modal Shape

5.5 Discussion

Similarly to the cases studied in previous sections, the agreement between the natural frequencies predicted by the complete and partitioned model is quite acceptable (errors are below 1 %, see table 10), a fact that provides definite evidence that the implementation of the Craig-Bampton algorithm is correct. Concerning the deformed shapes associated to each vibration mode, the resemblance is also rather reasonable —notice that sometimes the vibration modes exhibit the same "shape" yet opposite sign (Figure 56).

6 Concluding remarks

- From the values of natural frequencies obtained in three sets of simulations, it may be concluded that the implementation of the substructuring modal analysis methodology —the primary goal of this project— is correct, for the errors in predicting the natural frequencies are relatively low (less than 1 %) and in accordance to values reported in the related literature. Admittedly, discrepancies have been detected in some cases when visually comparing the deformed shapes computed with the classical modal analysis algorithm and the Craig-Bampton method. The exact reasons behind these discrepancies have not been, unfortunately, uncovered by our analyses. Future research should focus on trying to elucidate possible reasons behind these differences. As suggested in the first example, perhaps the causes may lie in the reconstruction process that allows one to plot the deformed shapes.

- All simulations carried out have been performed *automatically* by launching a single script in Matlab. For the shell problems (examples 2 and 3), this script calls a Python program, that in turns, calls the KRATOS software—which is programmed in C++. This software writes the finite element information into binary files that are then read by Matlab again to perform the required modal analysis. Finally, the deformed shapes are plotted by using

GID's postprocess facilities. In the substructuring version, this task is repeated for each partition.

- It should be noted that the simulation of the complete airplane model (design of the geometry, the partitioning, and the final meshing) has required an amount of work and time way higher than the devoted into coding and testing the substructuring algorithm. The geometry developed in this project has been extracted from a private airliner geometry, in which basic aeronautical structures have been added, such as spars, wings and even the torsion box. Even so, the quality of the geometry and meshing cannot be regarded as optimal from a engineering point of view. However, considering the amount of time assigned for this project, it can be considered an acceptable approximation in order to compare the results between the two algorithms studied in this project.

7 Future Lines of Research

This project can be the starting point for future research lines, namely

- As a first attempt to improve the performance of the program, it could be worthy to check the totality of the code and see if there are any bugs —mainly in the reconstruction process.
- Another possible research line would be to optimize the developed code. First of all, some MATLAB ™ operations are still programmed without vectorization —which is the recommended programming style in matlab. Further improvement can be achieved by parallelizing the modal vibration of all substructures. Another alternative could be simply to translate the entire problem into a general purpose language such as C++ or FORTRAN . This final solution would be, of course, a drastic one. It has to be also considered that, although a general purpose language tend to be faster than MATLAB ™, these kinds of languages lack the preimplemented functions that are commonly used in that platform, and this will suppose an extra amount of time in order to implement these auxiliar functionalities.
- Of course, it is also possible to simulate a more complex geometry than the ones treated in this study. This kind of models could be obtained by simply creating a finer mesh, as it can be seen in Figure 57, which would require more elements and so more computational effort. The other option is to develop, using CAD technologies a more complex geometry which would contain even more detailed structural components. If a better geometry is desired, it is possible to follow one of the suggested paths or even both, depending on the available resources for the researcher.

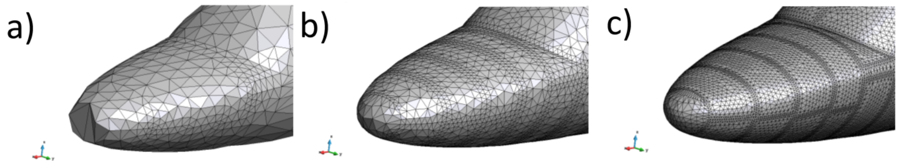


Figure 57: Mesh refinement: a) 17005 nodes with 36651 elements, b) 35599 nodes with 75739 elements, c) 128794 nodes with 268336 elements.

- Trying to increase the flexibility of the code is also an interesting research line. It would consist in changing the code structure at such a way that the geometry of each substructure may be exchangeable, allowing the simulation of models with different iterations for each substructure, as it is common to see in professional simulations, such the ones performed in the aeronautical industry.

References

[1] C Felippa. Introduction to finite element methods (asen 5007) course material: Lecture 10, 2013.

[2] Eugenio Oñate. *Structural analysis with the finite element method. Linear statics: volume 2: beams, plates and shells*. Springer Science & Business Media, 2013.

[3] C Felippa. Introduction to finite element methods (asen 5007) course material: Lecture 21, 2013.

[4] What is kratos? `http://kratos-wiki.cimne.upc.edu/index.php/What_is_Kratos`, note = Accessed: 2016-12-25.

[5] Cessna citation 2. `https://grabcad.com/library/cessna-citation-2-1`. Accessed: 2017-1-16.

[6] Cessna citation xls, brochure. `http://cessna.txtav.com/~/media/cessna/files/citation/xlsplus/xlsplus_brochure.ashx`, 2015. Accessed: 2016-9-10.

[7] Michel Géradin and Daniel J Rixen. *Mechanical vibrations: theory and application to structural dynamics*. John Wiley & Sons, 2014.

[8] Maurice Petyt. *Introduction to finite element vibration analysis*. Cambridge university press, 2010.

[9] Olgierd Cecil Zienkiewicz and Robert Leroy Taylor. *The finite element method: the basis*, volume 1. Butterworth-heinemann, 2000.

[10] Olgierd Cecil Zienkiewicz and Robert Leroy Taylor. *The finite element method: solid mechanics*, volume 2. Butterworth-heinemann, 2000.

[11] C Felippa. Introduction to finite element methods (asen 5007) course material: Lecture 16, 2013.

[12] C Felippa. Introduction to finite element methods (asen 5007) course material: Lecture 17, 2013.

[13] C Felippa. Introduction to finite element methods (asen 5007) course material: Lecture 18, 2013.

[14] Yair M Altman. *Accelerating MATLAB Performance: 1001 tips to speed up MATLAB programs*. CRC Press, 2014.

[15] Design breakdown of the p-51 mustang. `http://migrate.legendsintheirowntime.com/LiTOT/P51/P51_IA_4407_DA.html`. Accessed: 2016-12-29.

[16] The engineering toolbox. `http://www.engineeringtoolbox.com/poissons-ratio-d_1224.html`. Accessed: 2016-12-29.

[17] Michael Bauccio et al. *ASM metals reference book*. ASM international, 1993.

[18] C Esbrí. A-320: Structures, 2015.

[19] Eduardo Pinheiro, Ricardo Bianchini, Enrique V Carrera, and Taliver Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In *Workshop on compilers and operating systems for low power*, volume 180, pages 182–195. Barcelona, Spain, 2001.

A Costs

A.1 Workings Hours

To calculate the cost of time, an hypothetical salary of 36000 € has been considered. Using a working week of 40 hours, total hour-rate would be 18.75 €/h. Starting by this assumption and the hours listed below, the total amount corresponding to working hours can be seen in Table 11.

Table 11: Working hours costs

Concept	**Time**	**Cost**
Information research	90	1687.5
Development of the Project Charter	10	187.5
CAD software development	333	6243.75
GID file reverse engineering	15	281.25
Kratos files reverse engineering	20	375
Generation of the first problem geometry	10	187.5
Generation of the second problem geometry	24	450
Generation of the third problem geometry	264	4950
Data preparation and postprocess	40	750
Generate the mesh	5	93.75
Prepare external geometry to import	5	93.75
Modify Kratos source to extract matrices	30	562.5
Program implementation	214	4012.5
Implement vectorized modal analysis	20	375
Implement Craig-Bampton code	100	1875
Create Kratos interface	16	300
Implement beam element	20	375
Implement GiD geometry creator	48	900
Implement Kratos reader	10	187.5
Final data analysis and postprocess	15	281.25
Extract the results for the first problem	10	187.5
Compare the results of both problems	5	93.75
Documents and others	50	937.5
TOTAL	**752 hours**	**14100 €**

A.2 Software Licenses

For this project, the total cost related to software licenses has been computed. Most of the used software consist in open source projects. Also, most of them have important discounts for students.

Table 12: Software licenses costs

Concept	**Unit**	**Cost**
Cimne's kratos	1	open source
Cimne's gid	1	550*
Matlab	1	35.0*
Librecad	1	open source
Solidworks	1	135*
Gimp	1	open source
Latex	1	open source
TOTAL	**7**	**225 €**

A.3 Hardware

All the software development has been done using the same laptop machine, with a few peripherals, in order to improve work speed.

Table 13: Hardware costs

Concept	**Unit**	**Cost**
Laptop	1	840
Mouse	1	9.66
Keyboard	1	14.94
TOTAL	**3**	**864.6 €**

A.4 Total Budget

The total estimated budget cost of the project has been computed using the results of the previous sections.

Table 14: Working hours costs

Concept	**Cost**
Working hours costs	14100
Software licenses costs	225
Hardware costs	864.6
TOTAL	**15189.6 €**

B Environmental Impact

This project has no direct implication on that regard. All the same, clusters used for executing large FE simulations have high power consumption [19]. By reducing the computation time several orders of magnitude, the power need is also decreased.