L. Travaglini, S. Ricci, G. Bindolino
PyPAD: a Multidisciplinary Framework for Preliminary Airframe Design
Aircraft Engineering and Aerospace Technology, Vol. 88, N. 5, 2016, p. 649-664
doi:10.1108/AEAT-02-2015-0061

PyPAD: A Multidisciplinary Framework for Preliminary Airframe Design

Lorenzo Travaglini , Sergio Ricci , Giampiero Bindolino

Department of Aerospace Science and Technology

Politecnico di Milano, ITALY

Abstract

The preliminary design of an aircraft is a complex task involving a lot of different disciplines and it is becoming more challenging in the future due to the reduced budget and compressed time. At the same time new projects present new technical challenges due to the request for highly demanding performances, in order to reduce the fuel consumption (cost and environment impact) and to increase the payload. The preliminary design problem involves different disciplines, like: Aerodynamics, Controls, Regulations, Performances, Systems Definition, Design, Loads & Aeroelasticity, Structural Sizing, etc.. The present work focuses on the development of an integrated framework suitable for preliminary airframe design, called *PyPAD* (Python module for Preliminary Aircraft Design). The modules developed until now allow for the definition of multi fidelity aero-structural models starting from a CPACS input file and to compute static loads (trim) and flutter margin with the minimum effort by the user. Moreover *PyPAD* is able to compute the dynamic response under all the different load conditions, including discrete and continuous gust, nodal forces, command inputs. The tool is also able to export the state-space aeroelastic models tacking advantage of the modern state space model realization. In this way all the loads prescribed by the regulations can be computed in a fully automatic approach. A complete test case, starting from the CPACS input and ending with the definition of structural, aerodynamic and aeroelastic models and with the computation of different design loads is reported.

Keywords Preliminary Design, Aeroelasticity, Loads.

Paper type: Research paper

Introduction

The present work describes the development and an application of *PyPAD* (*Py*thon module for *P*reliminary *A*ircraft *D*esign), a framework suitable for preliminary airframe design. A special emphasis is devoted to the Design, Loads &

Aeroelasticity and Structural Sizing characteristics. All the other aspects are not neglected, but considered as inputs, constraints or output requests. Clearly, even considering a partial aspect of the project, the problem is still strongly multidisciplinary. The traditional approach iterates manually across models definition, loads computation, aeroleastic analysis and structural sizing. This could lead to a slow convergence to the best solution or, even worst, the solution found could not be the optimal one. Nowadays, thanks to computing power and engineering knowledge, the preliminary design process can be approached in a fully automatic way tacking advantage of the Multidisciplinary Design Optimization methods. Figure 1 depicts two logical flows, summarizing the traditional approach to preliminary design problem and the one here proposed. The main disadvantages of the first one are:

- All the steps are performed manually and by different departments inside a typical small, medium-size aircraft company. Engineers deal with repetitive works and the processes are error prone;
- A change in the conceptual design (due to aeroelastic behaviours or other problems) is very expensive. It is therefore hard to investigate different conceptual designs and the solution found could be not the best one.
- All the steps are performed manually and by different departments inside a typical small, medium-size aircraft company.

The proposed approach tries to solve all the highlighted problems:

- All the steps are performed using Multidisciplinary Optimization algorithms;
- Both structural and aerodynamic models (for loads computation and aeroelastic analysis) are defined in a fully automatic way. Therefore different conceptual designs can be investigated;
- Aeroelastic behaviour are considered in the sizing process.

To achieve these results a multidisciplinary framework called *PyPAD* and composed by different modules, is under development. The three main modules included in the full framework are:

- ***PyGFEM*** (a *Py*thon modules for the *G*eneration of *F*inite *E*lement *M*odels): It is an object-oriented tool developed under Abaqus-CAE environment able to define both structural and aerodynamic models starting from CPACS file input;
- ***PyAERO*** (a *Py*thon module for the *AERO*elastic analysis): It is a tool able to perform all the needed analysis needed to compute loads and aeroelastic responses.
- ***PySIZE*** (a *Py*thon module for the structural sizing): It will be a module dedicated to the sizing of the different components.

PyPAD runs embedded in the Abaqus-CAE environment so to take advantage of the large library of low level routines for FEM

One ckly gene

diffe adopted fo ork

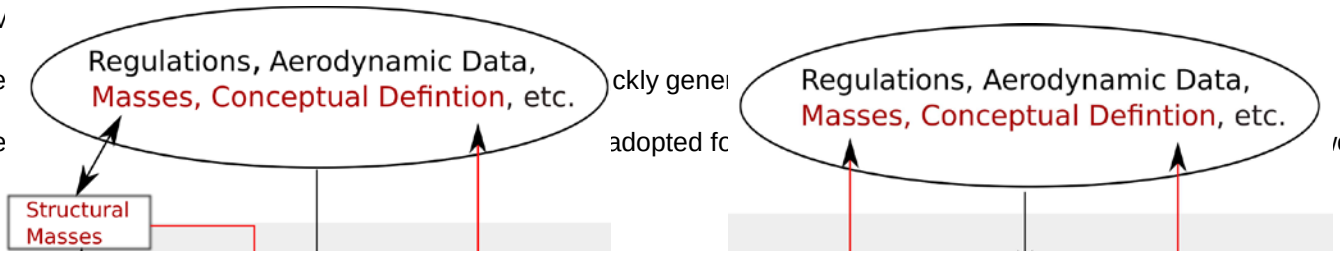
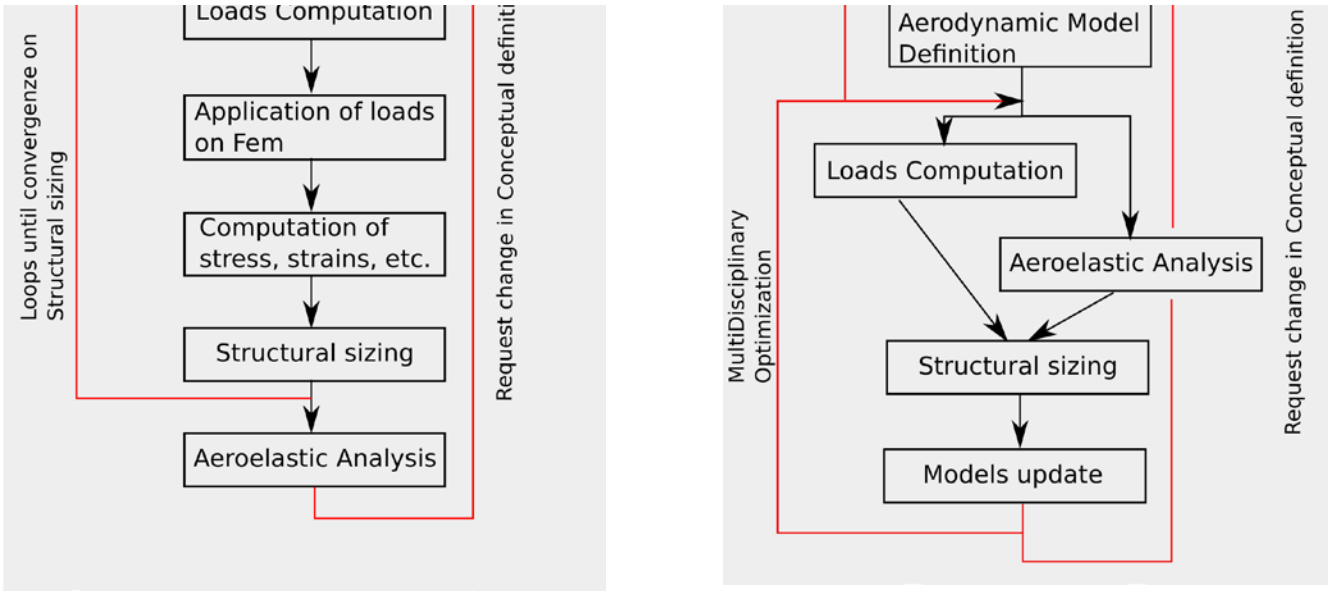


Figure 1 Preliminary Design: typical logic flow (left), proposed approach (right)



the parameterization called CPACS (The Common Parametric Aircraft Configuration Schema) (DLR, 2013) proposed by DLR is adopted. CPACS is a data definition for the air transport system based on a single XML file.

This data definition is becoming a standard due to its extendibility and completeness. A lot of data can be stored in a CPACS file; the most important ones for this work are the geometry definitions, both external lofts and structural configurations, as well as the related properties definition. In the following a quick overview of two of the main modules of *PyPAD* is reported, as well as some specific examples.

Aero-structural models generation: *PyGFEM* module

PyGFEM is the module dedicated to the automatic generation of the full finite element model of the complete aircraft. The capability to quickly generate a large spectrum of aerostructural models is a must due to the need to find the best

aircraft optimization by means of different iterations and mono-multi disciplinary optimization runs. Figure 2 depicts the main idea driving the development of this, where the main core can be divided in two branches:

1. The core classes providing the definition of basic objects, like edges, airfoils, planes, cuts, lofts, materials, properties, etc.;
2. The interface classes providing the coupling between CPACS input and core objects.

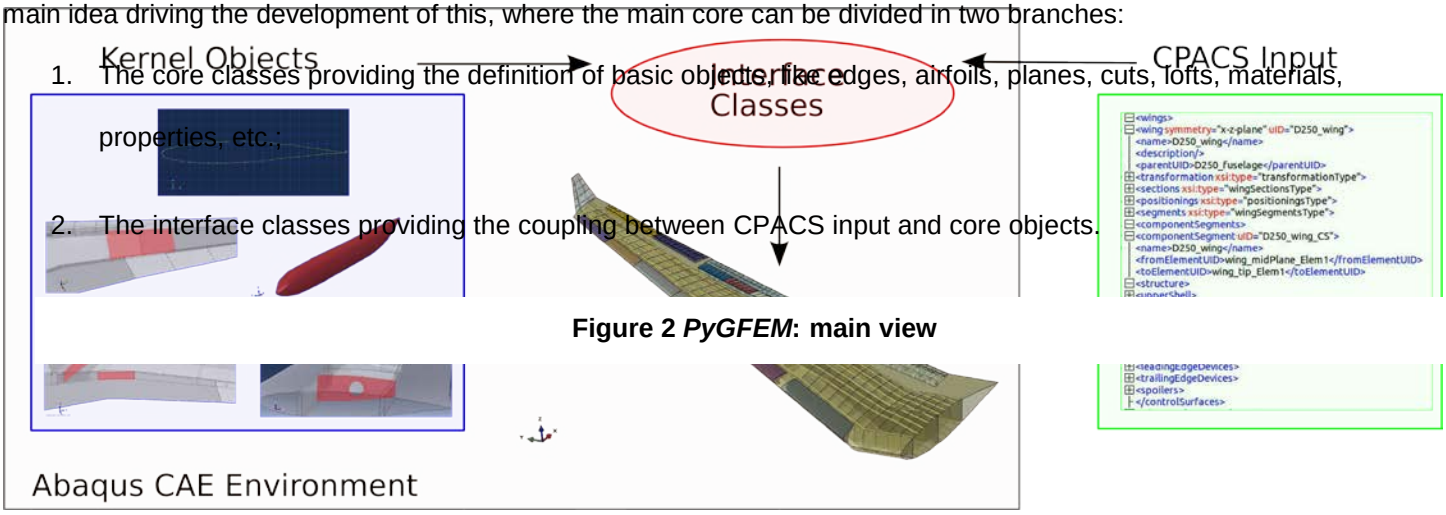


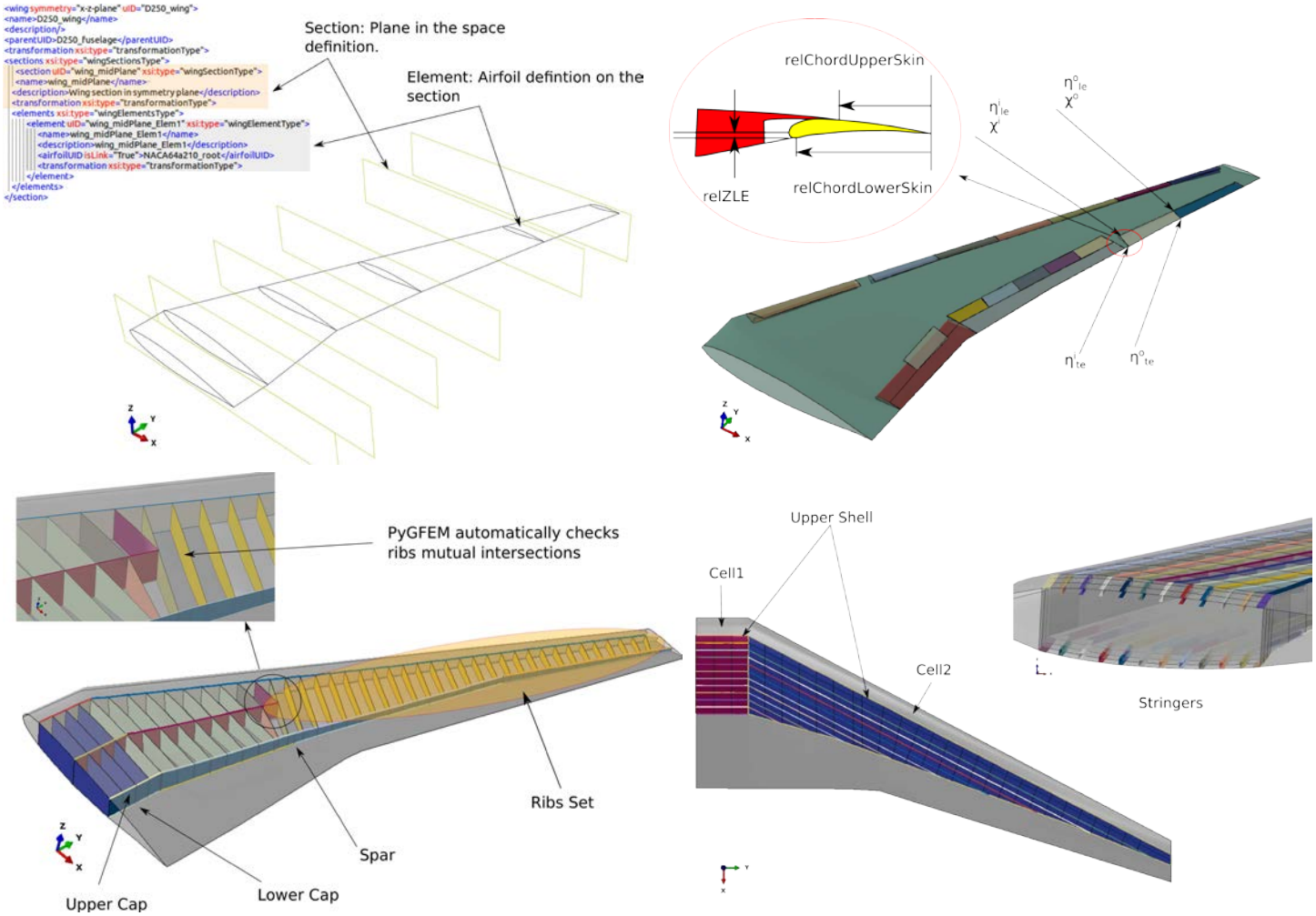
Figure 2 *PyGFEM*: main view

In such a way the core objects were developed without constraints by the inputs, providing generality and re-usability. The task to represent the final objects starting from CPACS file is carried out by the interface classes, the only ones depending on the input file. First of all *PyGFEM* reads from CPACS file all the information concerning airfoils (both for wings and fuselages), materials, beam profiles and properties definitions, common data needed for the definitions of all the others objects. Once all these data are stored in Abaqus-CAE database, *PyGFEM* can define wings and fuselages.

Wing

The loft definition (for both wings and fuselages) is described by sections and elements. The section defines a plane in the space and each section can hold different elements each of them defines an airfoil on its plane. To drive the loft the leading and trailing edges are defined and used as path for the loft extrusion (Figure 3 top left). Once the external loft is defined, the movable surfaces (leading edges and trailing edges devices and spoilers) can be defined (Figure 3 top right). The idea is to take advantage of the same classes used to describe the wing lofts. The user defines inboard and outboard positions (relative to the ‘parent‘ wing), while for the definition of the 2D geometry he can chose airfoils (among the ones defined) or can define them starting from the wing geometry. In the last case *PyGFEM* automatically computes the correct airfoils and defines the movable surfaces using the same approach used for the wing. At the end it modifies the parent loft removing the faces belonging to the new component. Also the leading edge devices are defined using this idea, but in this

case a new surface is added to the parent geometry, in order to redefine the structural leading edge. It is important to underline that the movable surfaces are objects derived by the same class of the wing one, therefore it is possible to



define other devices referring to a movable surface (like double slot flaps). Once all the movable surfaces are defined *PyGFEM* starts assembling the structural model of the different components (wings, movables surfaces and fuselages). For the wings, the principal structural component are: spars, ribs, skins and stringers.

Figure 3 Wings definition: Sections & Elements (top left), Movables surfaces (top right), Spars & Ribs (bottom left), Skins & Stringers (bottom right)

PyGFEM can define spars arbitrarily oriented in the space, with arbitrarily property partition. On the upper and lower edge of the spar, if specified, two stringers representing the lower and the upper cup are added. Ribs are collected in set, and if specified, each rib, in the ribs set, can have a different property.

As for spars, also the ribs can be defined arbitrarily in the space. The ribs can start from each characteristic geometry, like leading edge and spars, can end on trailing edge, spars and if specified can end when *PyGFEM* finds an intersection with another rib. Moreover it is possible to force the rib to be perpendicular to a reference geometry (leading edge, trailing edge, a spar, the local y axis) (Figure 3 bottom left). Knowing spars and ribs positions it is possible to define skins and stringers. CPACS provides a nested data structure for the description of the skins. The basic object of this data is the

simple skin that can be defined using spars, ribs, leading and trailing edges as border lines or by the definition of bounding parametric points on the loft. Inside this object the user can define (using relative parameters) other skins with different properties. Besides in each skin the user can define a stringers set specifying the number of stringers or the pitch between stringers or each relative position. Moreover the stringers can be rotated respect to the direction defined by the leading edge (Figure 3 bottom right).

Fuselage

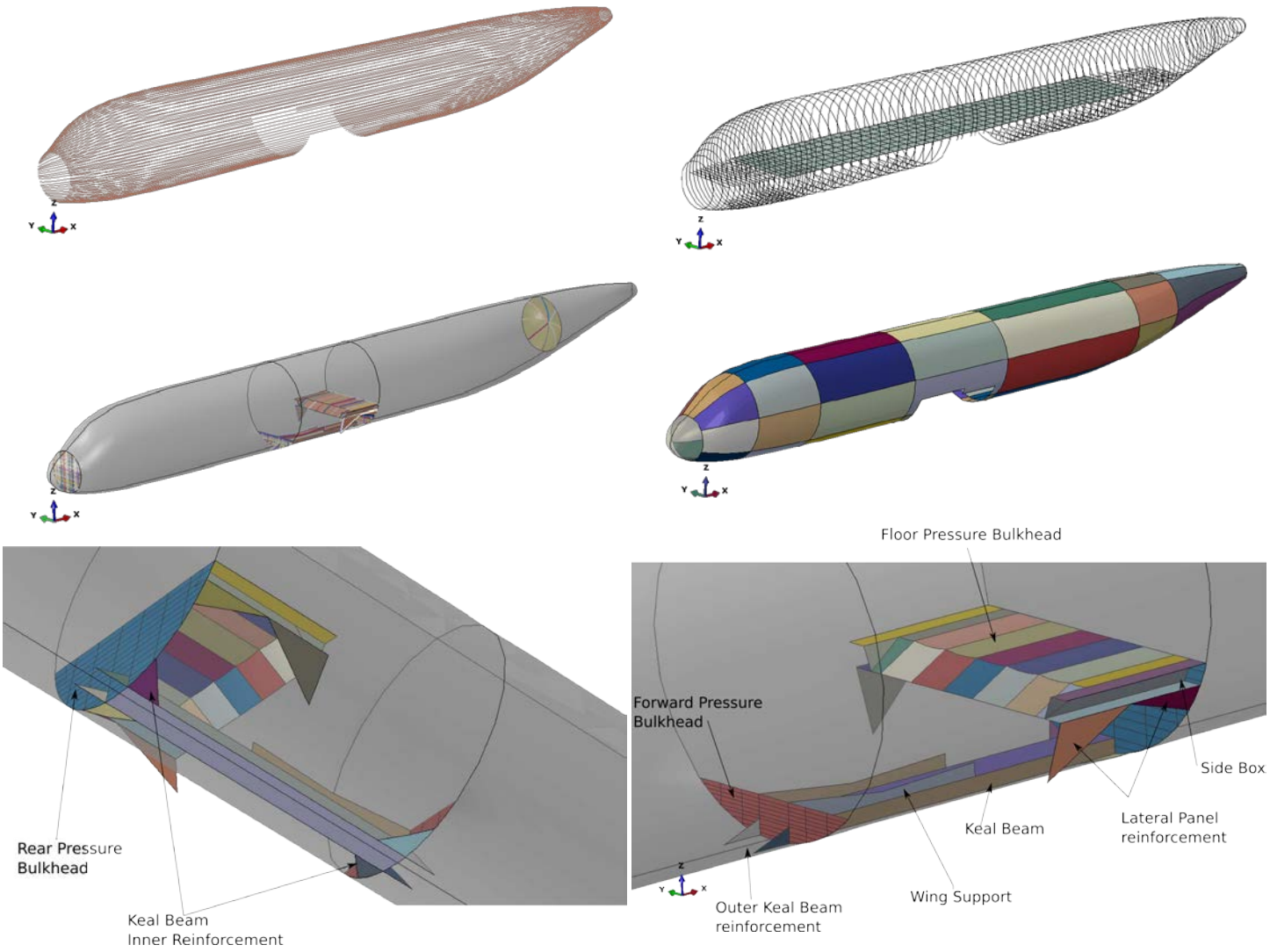
Regarding the structural definition of the fuselage a different approach was used. While for the wing all the structural objects were defined on the same part, sharing edges and points, for the fuselages, different parts are defined and then linked together using rigid links. Using this approach the time needed by the tool to define the model is substantially reduced. This because the whole time execution is driven by the stringers (and frames) definition, particularly by the cuts on the surfaces. The number of stringers on the wing could be very high, but it is possible to perform the cut of all the stringers at the same time, so saving a lot of computational time. The same approach is not feasible on the fuselage due to the curvature of the geometry and to the mutual cuts between stringers and frames. At the moment *PyGFEM* is able to define a complete FE model of a transport aircraft fuselage, and the classes are easily extendible to others type of fuselages. At the moment the limitation is on the input file. Figure 4 shows all the different parts and a detail of the central fuselage. The floor structure is modelled coupling beam and plate elements. Cross beams, strut beams and floor beams are described by beam elements, while the floor is described by plate elements (Figure top left). *PyGFE*M automatically defines the central fuselage, the part where the wing is linked to the fuselage (Figures 4 bottom) using different objects:

- The Keal Beam and its reinforcements defined to restore the bending stiffness decreased by the cut for the wing box;
- The forward and rear pressure bulkheads, defined to carry on the pressure load of the cabin;
- The Side Boxes, defined to restore the lateral bending stiffness;
- The Lateral Panel reinforcements, defined to stiffen the main landing gear bay;
- The Floor Pressure Bulkhead, defined to carry on the pressure load of the cabin.

Properties and Sets

The database defined in Abaqus CAE environment can be managed and accessible to the user using PyGFEM. The objects defined are able to handle all the properties of the model, moreover all the objects hold informations about their geometry and mesh using different set definitions. This will be very helpful during optimization study and will be very useful to run analysis on local objects like buckling using ad-hoc solutions.

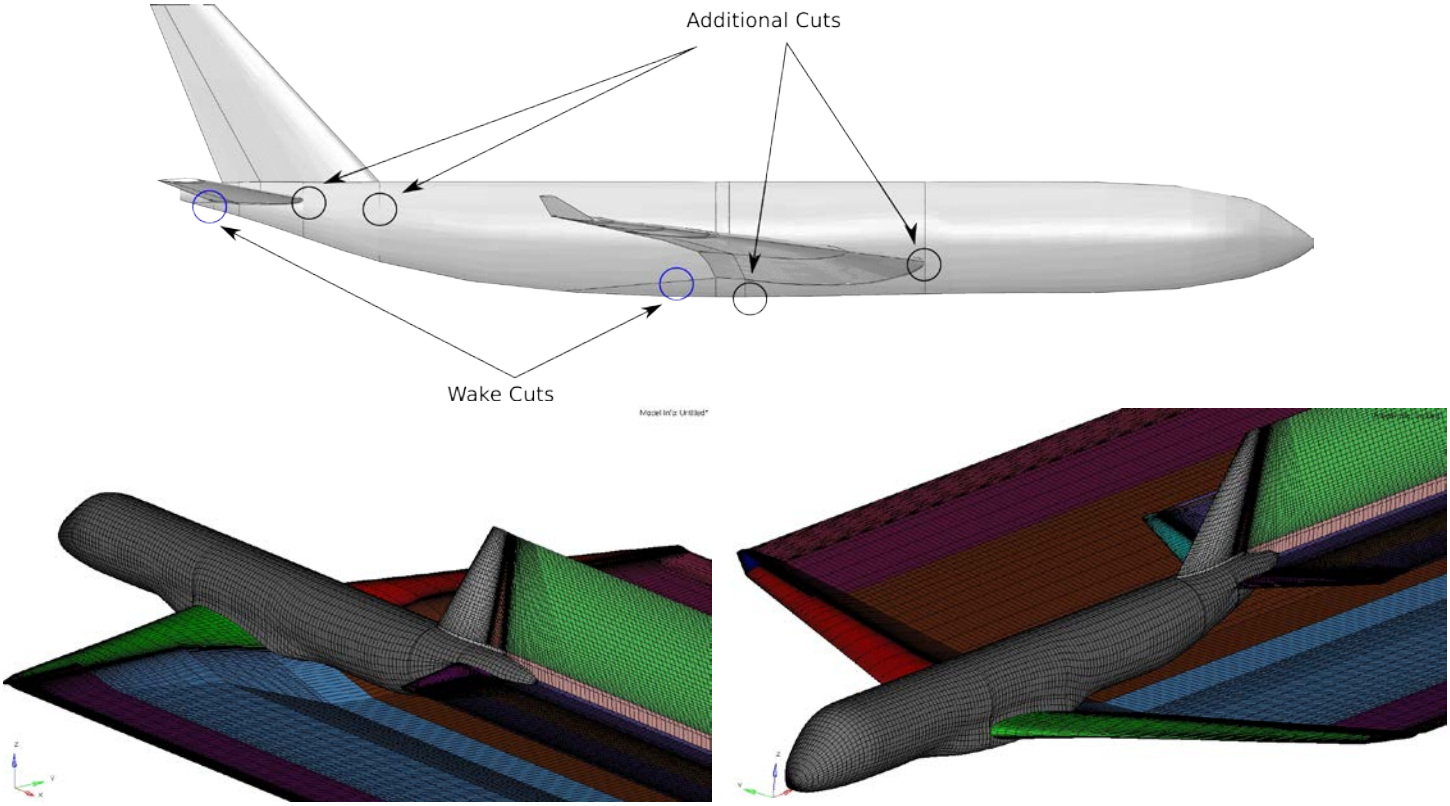
Figure 4 Fuselages definition



Aerodynamic models

PyGFEM is able to define a structured aerodynamic mesh to be used as input for ***SUnPaM*** (*S*ubsonic *Un*steady *P*anel *M*ethod), the aerodynamic solver available inside *PyPAD*, based on the Morino's method. Figure 5 (bottom) show different views of the results obtained on a test case. Starting from the different parts, the program merges all the components finding each intersection, then starts to check all the different couples of intersecting parts, adding construction cuts to simplify the geometry, or cuts to define the projection of wings wake on the other part involved in the intersection (Figure 5 top).

Figure 5 Aerodynamic Mesh



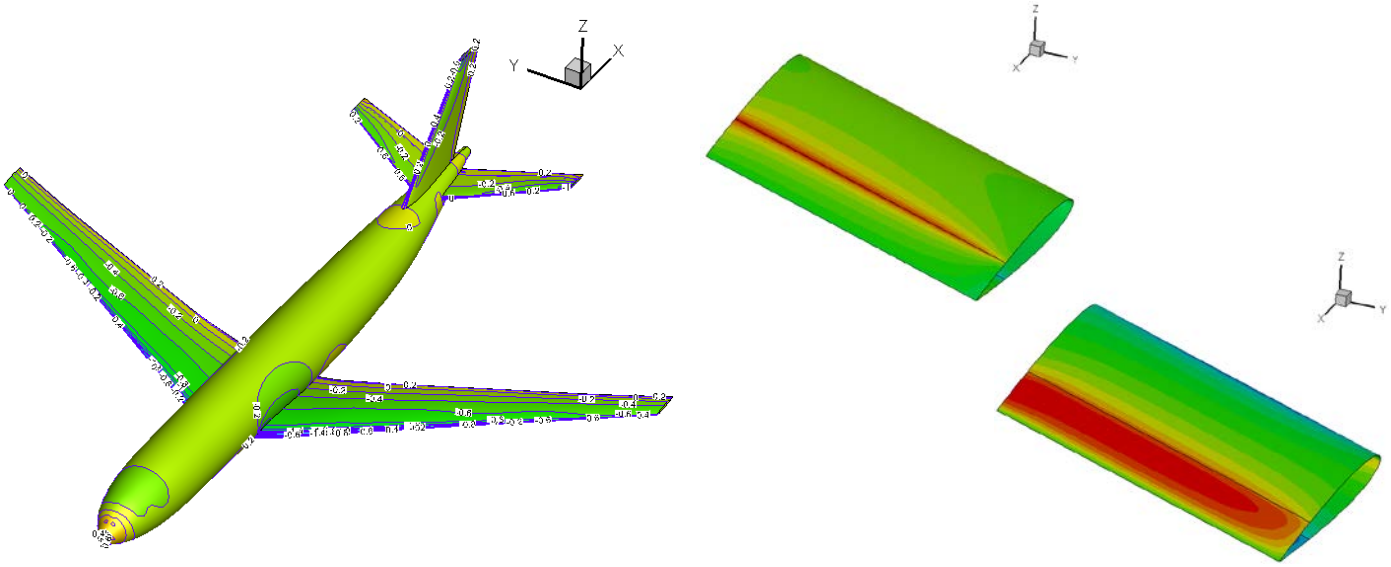
Loads and aeroelasticity: *PyAERO* module

PyAERO is a module developed to compute all the aeroelastic responses such as Trim, Flutter, Dynamic Analysis, in order to get all the loads and all the responses needed for the structural sizing. *PyAERO* uses the structural model defined by *PyGFEM* and the aerodynamic database computed using *SunPaM*. All the routines are written in Python, but to get better computational time performances, some core functions are written in FORTRAN, in order to take advantages of numerical libraries like Lapack, Optimized Blas and FFTW and to exploit the power of parallel computing using OpenMP.

SunPaM

SUnPaM (*S*ubsonic *Un*steady *Pa*nel *M*ethod) is a panel aerodynamic solver, based on the Morino method (Morino et al., 1975) & (Morino and Kuo, 1974) and developed in the Department of Aerospace Science and Technology of Politecnico di Milano. The tool solves linearized subsonic compressible flow both in steady and unsteady conditions. The program can exploit the power of modern multi-core CPU, both using external libraries for linear algebra (OpenBlas, Plasma (The University of Tennessee)) and OpenMP during the definition of the matrices. Figure 6 shows some test cases for the steady solution (full aircraft Figure 6 left) and on a single wing with oscillating flap (Figure 6 right).

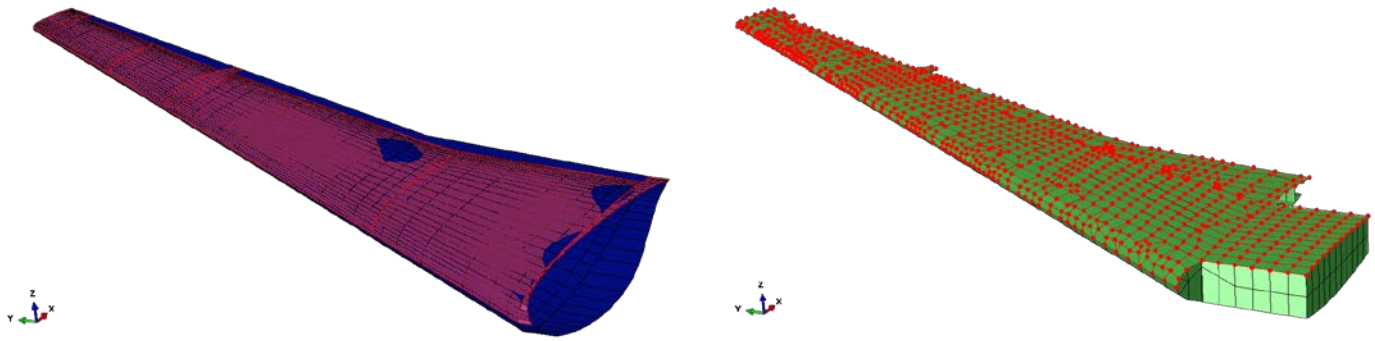
Figure 6 *SUnPaM* results: steady solution (left), unsteady solution (imaginary and real part, right)



Aeroelastic models

As previously introduced, PyAERO couples the structural and the aerodynamic models coming from *PyGFEM* by RBF (Radial Basis Functions) interpolations. Regarding the structural model, at the moment *PyAERO* works using as input an eigenvalue solution computed by Abaqus.

Figure 7 Aeroelastic coupling sets, aerodynamic (left) and structural (right)



It imports all the grids definition (used to couple the structural model with the aerodynamic one), the modal displacements, all the modal information and the nodal mass and stiffness matrices. The definition of the aeroelastic coupling between structural and aerodynamic models becomes very simple taking advantage of the automatic definition of both the models by *PyGFEM*. The structural model is divided in different parts each described by different sets and the same partition is reflected also on the aerodynamic mesh. In this way, to define an interface element between structural and aerodynamic mesh, the user must specify only the part name (of the structural model) the body name (of the aerodynamic model) and

the two set of the structural model describing the upper and lower surface. Using this information the solver is able to compute the interfaces matrices to get displacements and velocities on aerodynamic mesh and forces on the structural one.

Aeroelastic solvers

As introduced *PyAERO* provides different aeroelastic solvers in order to be able to perform all the analysis requested during the preliminary sizing of an aircraft.

Trim Solution

PyAERO solves the Trim problem using a classical approach, therefore assuming a steady solution on the deformable motion and considering a steady approximation of the aerodynamic. The trim problem can be written as:

$$\boldsymbol{M\ddot{u} + Ku = F_0 + q_\infty Q_{su} u + q_\infty Q_{sx} u_x}$$

where $\boldsymbol{M}$ and $\boldsymbol{K}$ are the mass and the stiffness matrices, $\boldsymbol{u}$ is the vector of nodal displacements, $\boldsymbol{F_0}$ is the vector of initial nodal forces, q_∞ is the dynamic pressure, $\boldsymbol{Q_{su}}$ is the steady approximation of aerodynamic forces due to nodal displacements and $\boldsymbol{Q_{sx}}$ is the steady approximation of aerodynamic forces due to mechanical flight degree of freedom $\boldsymbol{u_x}$ (like *α, β*, p, q, r, etc.). Given the complexity of the structural model the nodal displacements $\boldsymbol{u}$ are described using a modal bases:

$$\boldsymbol{u = Uq}$$

Where $\boldsymbol{U}$ is the modal base and $\boldsymbol{q}$ are the modal amplitudes. The modal bases is defined considering also the six rigid degrees of freedom, described in physical coordinates. Replacing the approximation of $\boldsymbol{u}$ in the trim Equation:

$$\boldsymbol{MU\ddot{q} + KUq = F_0 + q_\infty Q_{su} Uq} + q_\infty \boldsymbol{Q_{sx} u_x}$$

To improve computation performances *PyAERO* computes directly (using *SUnPAM*) the aerodynamic matrix $\boldsymbol{\tilde{Q}_{su} = Q_{su} U}$. Thanks to the orthogonality properties of eigenvector, scaling the last Equation by $\boldsymbol{U^T}$ it is possible to get:

$$\begin{bmatrix} \boldsymbol{M}^r & 0 \\ 0 & \boldsymbol{m}^l \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_r \\ \ddot{\boldsymbol{q}}_l \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{k}^l \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_r \\ \boldsymbol{q}_l \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_0^r \\ \boldsymbol{F}_0^l \end{bmatrix} + q_\infty \begin{bmatrix} \breve{\boldsymbol{Q}}_{su}^{rr} & \breve{\boldsymbol{Q}}_{su}^{rl} \\ \breve{\boldsymbol{Q}}_{su}^{lr} & \breve{\boldsymbol{Q}}_{su}^{ll} \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_r \\ \boldsymbol{q}_l \end{bmatrix} + q_\infty \begin{bmatrix} \breve{\boldsymbol{Q}}_{sx}^{r} \\ \breve{\boldsymbol{Q}}_{sx}^{l} \end{bmatrix} \boldsymbol{u}_x$$

Where $\boldsymbol{M}^r$ is the global barycentric mass matrix, $\boldsymbol{m}^l$ is the diagonal modal mass matrix associated to flexible modes, $\boldsymbol{k}^l$ is the diagonal stiffness matrix. Under the hypothesis of steady solution of the deformable part, the problem has $2n_r + n_l + n_x$ degrees of freedom with $n_r + n_l$ equations, where n_l, n_r and n_x are the number of modes, the number of rigid degrees of freedom (six) and the number of flight mechanics degrees of freedom. The six unknowns $\boldsymbol{q}_r$ can be neglected considering that the gravity center position and the rotation around *x* axis (speed direction) do not give aerodynamic loads, while the

pitch and yaw rotation are already described by α and β angle of the flight mechanics set. Therefore the number of unknowns is $n_r + n_l + n_x$ with $n_r + n_l$ equations. This means that to have a close problem the user must fix n_x unknowns. Once the trim problem is solved, using $\widetilde{\boldsymbol{Q}}_{su}$ and $\boldsymbol{Q}_{sx}$ it is possible to compute the nodal aerodynamic loads. Moreover, knowing the rigid accelerations and the nodal mass matrix, it is also possible to compute the inertia nodal loads. In this way for every trim conditions the user can export the corresponding nodal loads to compute the corresponding stress solution.

Flutter

PyAERO includes a dedicated well know (MSC.NASTRAN) & (Harder et al., 1979) p-k futter solver in order to get the best compromise between solution robustness and time efficiency. The utter problem can be written as:

$$\left(s^2\boldsymbol{M} + s\boldsymbol{C} + \boldsymbol{K} - q_\infty \boldsymbol{Q}_{qq}(Mach, p)\right)\boldsymbol{q} = 0$$

Where $\boldsymbol{M}$, $\boldsymbol{C}$ and $\boldsymbol{K}$ are the modal mass, damping and stiffness matrices and p is the complex reduced frequency:

$$p = \frac{sc}{2V_\infty} = g + k$$

Being c the reference cord and V_∞ the flight speed.

The aerodynamic matrix is computed at different pure harmonic reduced frequencies. The p-k method considers the $\boldsymbol{Q}_{qq}$ matrix as the sum of the real part $\boldsymbol{Q}_{qq}^R$ and the imaginary part $\boldsymbol{Q}_{qq}^I$. Considering the imaginary part it is possible to write:

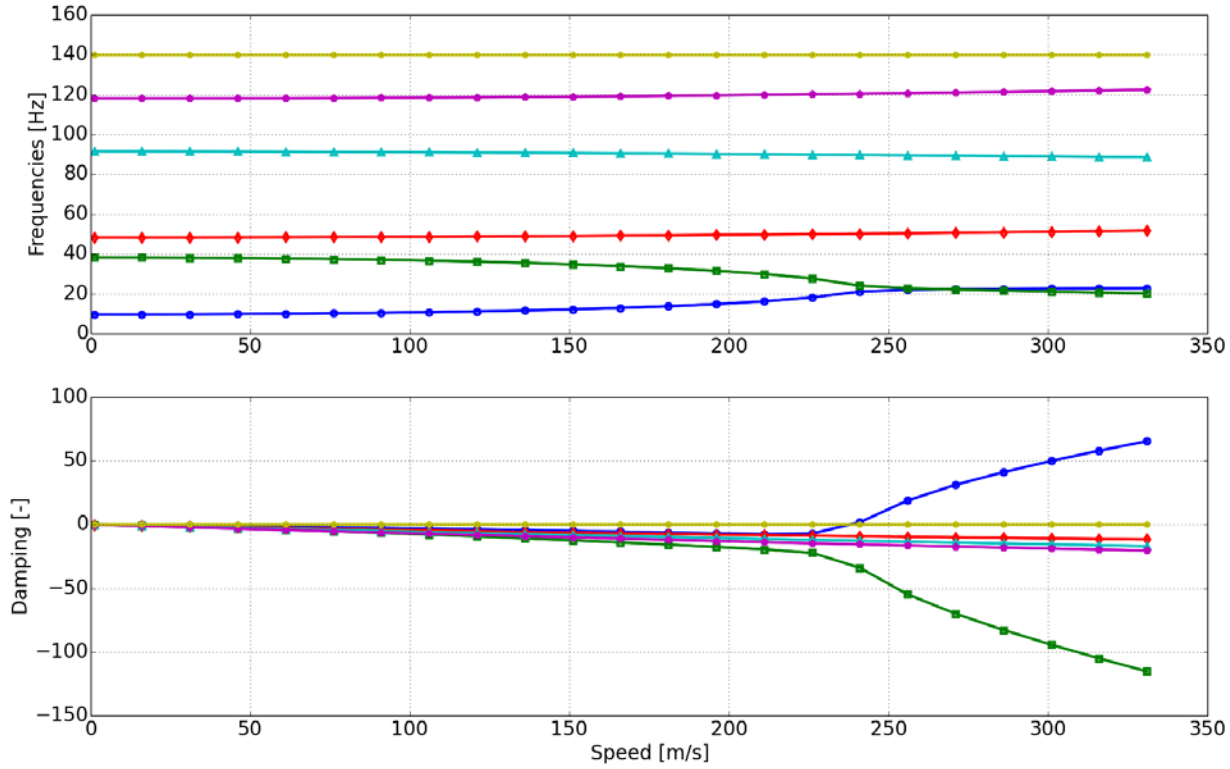
$$q_\infty \boldsymbol{Q}_{qq}^I = \frac{\rho}{2}V_\infty^2 \boldsymbol{Q}_{qq}^I \frac{k}{k} = \frac{\rho}{4}cV_\infty \frac{\boldsymbol{Q}_{qq}^I}{k}\,\omega \simeq \frac{\rho}{4}cV_\infty \frac{\boldsymbol{Q}_{qq}^I}{k}\,s$$

Using this approximation, the flutter equation can be approximated by:

$$\left(s^2\boldsymbol{M} + s\left(\boldsymbol{C} + \frac{\rho}{4}cV_\infty \frac{\boldsymbol{Q}_{qq}^I(Mack, k)}{k}\right) + \boldsymbol{K} - q_\infty \boldsymbol{Q}_{qq}^R\,(Mach, k)\right)\boldsymbol{q} = 0$$

The problem now is defined by pure real matrices and can be solved by normal eigenvalue algorithm. The solution must be iterate on the different eigenvalues to get the correct $\boldsymbol{Q}_{qq}$ approximation at the correct reduced frequencies k. Figure 8 shows a flutter diagram on the AGARD 445.6 wing test (Yates et al., 1963) & (Yates, 1987) performed for validation purpose.

Figure 8 Flutter: Agard 445.6 wing Test



Dynamic Response

The dynamic modal problem in frequency domain can be described by:

$$\left(\omega^2 \boldsymbol{M} + \mathrm{j}\omega \boldsymbol{C} + \boldsymbol{K} - q_\infty \boldsymbol{Q}_{qq}(Mach, k)\right) \boldsymbol{q} = \boldsymbol{F}(\omega)$$

Where $\boldsymbol{F}(\omega)$ can be both deterministic and stochastic vector of forces.

At the moment in *PyAERO* the user can simply define deterministic gusts, prescribed control surfaces motions and nodal forces. Once the reduced modal problem is solved, the software is able to compute nodal responses on the structural model both in term of displacements and forces. The user can ask as output displacements, velocities, accelerations, total force on specific set of nodes, total force on a part and nodal forces. Moreover the user, to save memory, can chose to get the wanted output only at a fixed time step. To write the problem in the time domain, a state space aerodynamic model must be computed starting from the frequencies database of $\boldsymbol{Q}_{qq}(Mach, k)$.

PyAERO takes advantages of the modern techniques developed in our department in the last years (Ripepi and Mategazza, 2013) & (Ripepi, 2014). The aerodynamic model in time domain can be described by:

$$\dot{\boldsymbol{x}}_A(t) = 2\,\frac{V_\infty}{c}\,\boldsymbol{A}^A \boldsymbol{x}_A(t) + 2\,\frac{V_\infty}{c}\,\boldsymbol{B}^A \boldsymbol{q}(t)$$

$$\boldsymbol{f}_A(t) = q_\infty \left(\boldsymbol{C}^A \boldsymbol{x}_A(t) + \boldsymbol{D}_0^A \boldsymbol{q}(t) + \frac{c}{2V_\infty} \boldsymbol{D}_1^A \dot{\boldsymbol{q}}(t) + \left(\frac{c}{2V_\infty}\right)^2 \boldsymbol{D}_2^A \ddot{\boldsymbol{q}}(t) \right)$$

Where $x_A(t)$ is the aerodynamic state, $\boldsymbol{A}^A, \boldsymbol{B}^A, \boldsymbol{C}^A, \boldsymbol{D}_0^A, \boldsymbol{D}_1^A$ & $\boldsymbol{D}_2^A$ are the matrices of the aerodynamic state space model, $\boldsymbol{f}_A(t)$ is the vector of the aerodynamic force acting on the structural modal coordinates.

Coupling the aerodynamic system and the structural one:

$$\boldsymbol{M}\ddot{\boldsymbol{q}}(t) + \boldsymbol{C}\dot{\boldsymbol{q}}(t) + \boldsymbol{K}\boldsymbol{q}(t) = \boldsymbol{f}_A(t) + \boldsymbol{f}_G(t)$$

Where $\boldsymbol{f}_G(t)$ is the vector of gust loads. The aeroelastic model becomes:

$$\begin{bmatrix} \dot{\boldsymbol{x}}_A \\ \dot{\boldsymbol{q}} \\ \boldsymbol{q} \end{bmatrix} = \begin{bmatrix} 2V_\infty/c\boldsymbol{A}^A & 2V_\infty/c\boldsymbol{B}^A & 0 \\ 0 & I & 0 \\ q_\infty \breve{\boldsymbol{M}}^{-1}\boldsymbol{C}^A & \breve{\boldsymbol{M}}^{-1}\breve{\boldsymbol{K}} & \breve{\boldsymbol{M}}^{-1}\breve{\boldsymbol{C}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \breve{\boldsymbol{M}}^{-1} \end{bmatrix} \boldsymbol{f}_G$$

Defining:

$$\breve{\boldsymbol{M}} = \boldsymbol{M} - q_\infty \left(\frac{c}{2V_\infty}\right)^2 \boldsymbol{D}_2^A$$
$$\breve{\boldsymbol{C}} = \boldsymbol{C} - q_\infty \frac{c}{2V_\infty} \boldsymbol{D}_1^A$$
$$\breve{\boldsymbol{K}} = \boldsymbol{K} - q_\infty \boldsymbol{D}_0^A$$

For what concerning the gust input $\boldsymbol{f}_G$, at the moment *PyAERO* uses a hybrid approach that combines the Fourier-transform and the time-domain approaches, presented by Karpel et al. (2005). The purpose of this approach is to obtain a time-domain state-space model without performing rational approximation to the spiral gust columns. The Fourier transform of the gust profile is computed, than the modal loads $\boldsymbol{f}_G(\omega)$ are defined in the frequency domain. At this point the term$\boldsymbol{f}_G(t)$ is computed by mean of inverse Fourier transform.

Application

A test case involving all the different capabilities of the framework just described is reported. The test case is based on a CPACS model of a transport aircraft courtesy of Dieter Kohlgrueber (DLR) used for the development of the models and analysis reported in (Scherer et al., 2013). Table 1 summarizes the principal data of the model.

Table 1 mass and geometry properties

M [kg]	I_{xx} [Kgm^2]	I_{yy} [Kgm^2]	I_{zz} [Kgm^2]	I_{xz} [Kgm^2]	S [m^2]	c [m]	b [m]
48475	961948.5	1895684.	2768967.	-133950.7	122.4	4.2	16

Structural and aerodynamic models

The structural model is composed by 21 parts, 29945 nodes and 40969 elements. Figure 9 depicts some views of the model and the detail of the wing fuselage interface. Figure 10 shows different views of the aerodynamic mesh used both for steady and unsteady analysis. The total number of panels is approximately 20000. The aeroelastic model is defined

using the first 25 normal modes of the aircraft. Figures 11 show the first three deformable normal modes of the model, comparing the shape on the structural mesh and on the aerodynamic one.

Figure 9 Structural Model

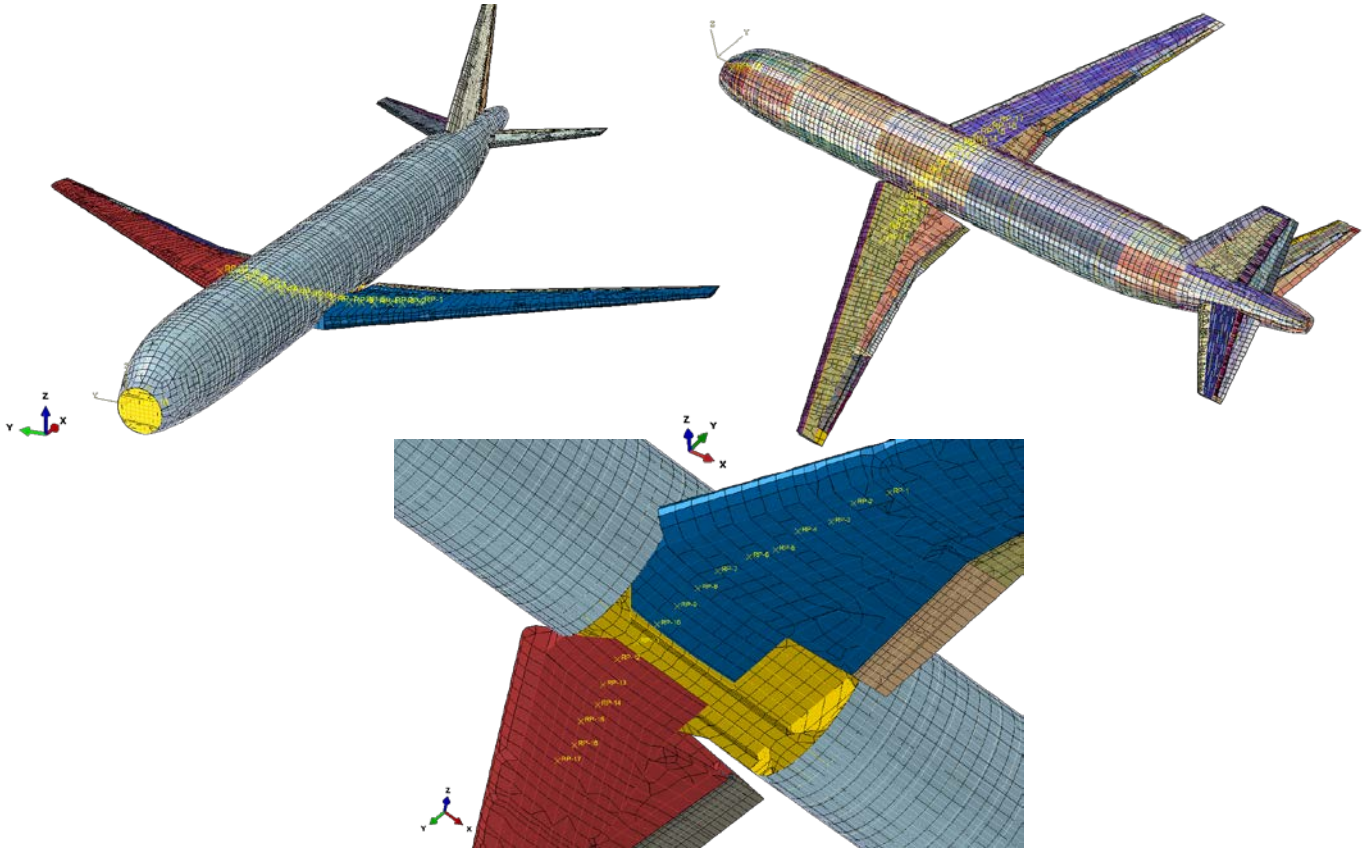


Figure 10 Aerodynamic model

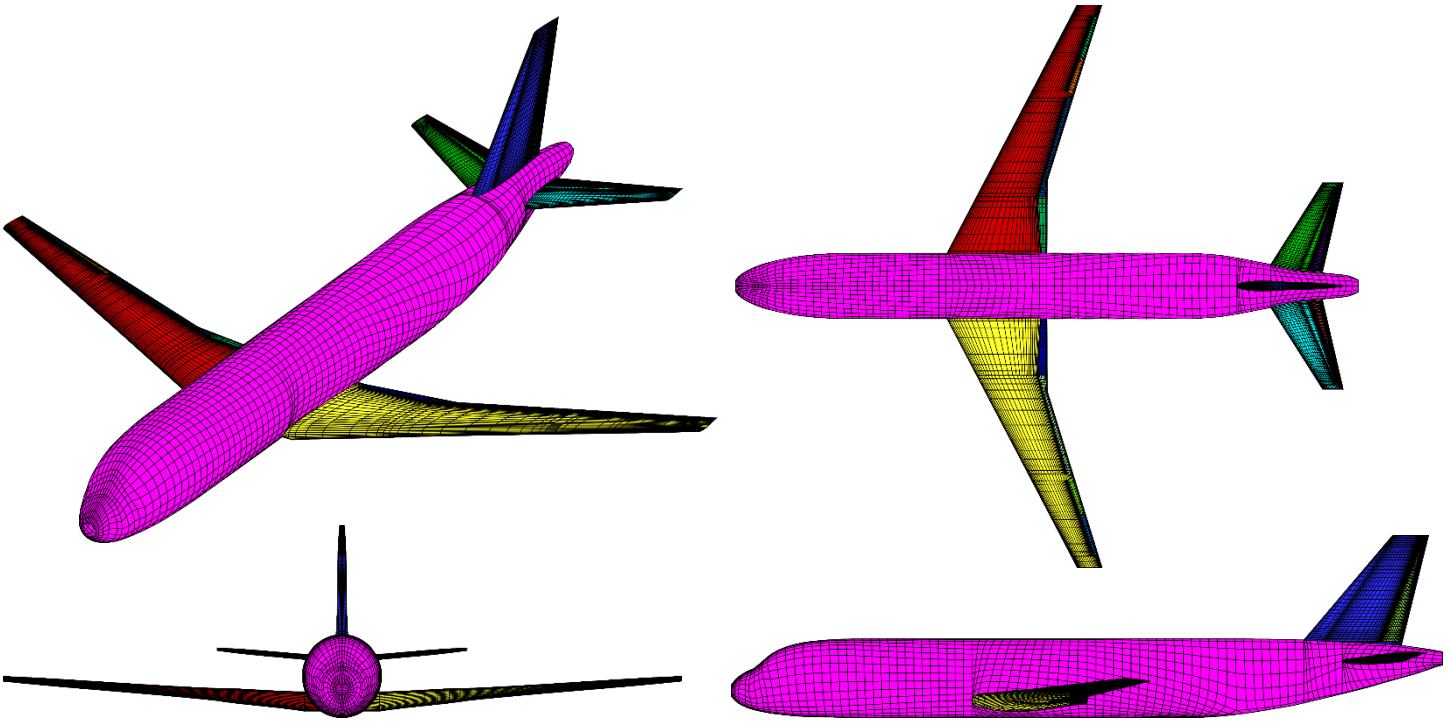
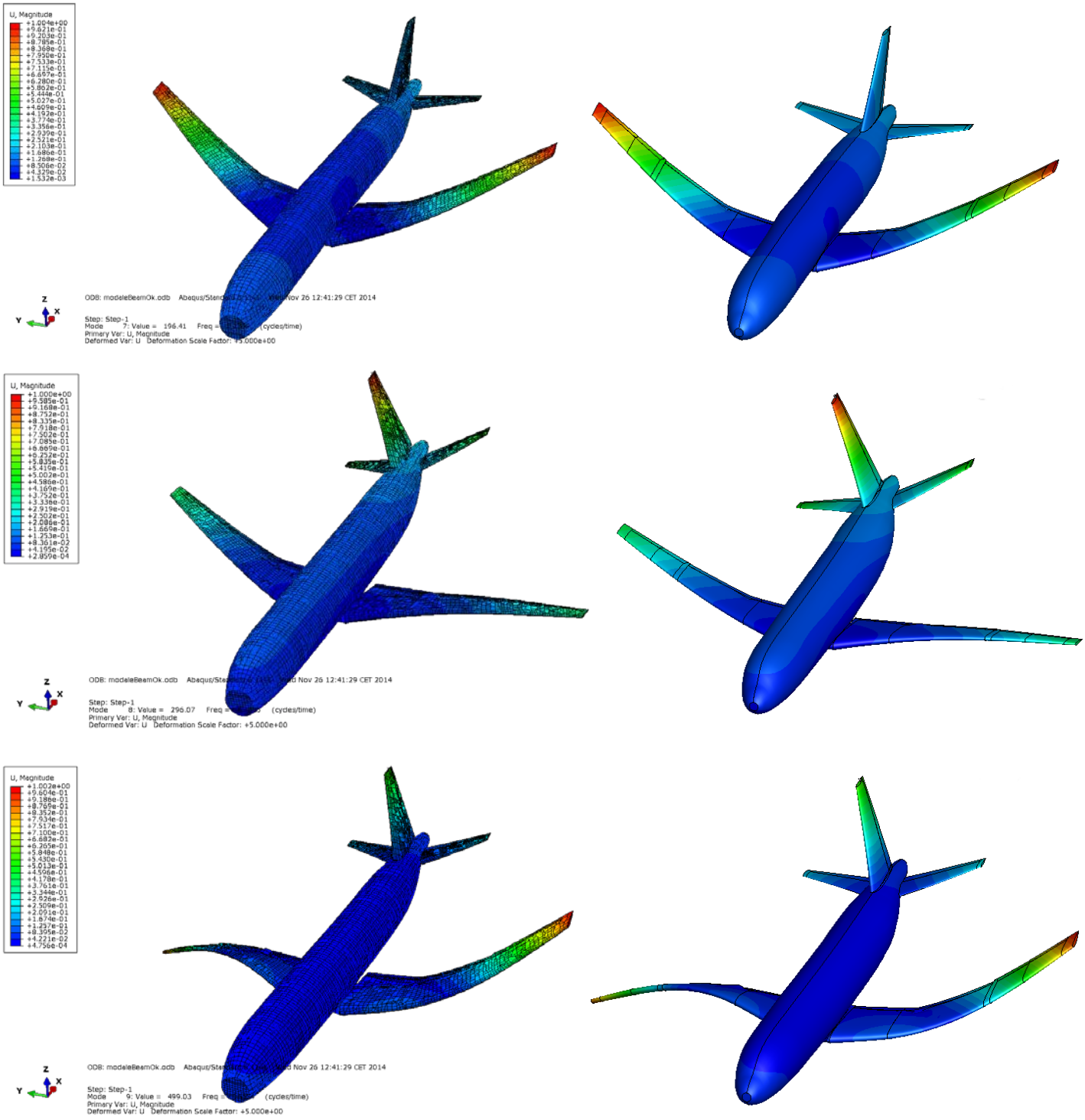


Figure 11 First three elastic modes, structural and aerodynamic meshes



Trim Analysis

Six different trim analysis (all at sea level) are presented:

- 2 Level flight conditions at Mach 0.3 and Mach 0.5,
- 2 Pull-up conditions (n_z = 2:5) at Mach 0.3 and Mach 0.5,

- 2 Sideslip conditions with b = 10deg at Mach 0.3 and Mach 0.5.

For all the conditions the nodal loads are computed and exported in Abaqus format. These loads are then used to perform a static analysis in Abaqus to get the stress solution. Tables 2 and 3 report the principal stability and control derivatives, for the rigid and for the elastic model, Table 4 reports the trim solutions (rigid and elastic model) for the level flight conditions, Table 5 reports the trim solutions for the pull-up conditions and Table 6 reports the solutions for the sideslip analysis (angles in degree and accelerations in [m/s^2]). Figure 12 shows the displacement solution and the Von-Mises stress distribution for the different load cases.

Table 2 stability and control derivatives, rigid model

Mach	$c_{l\alpha}$	$c_{M\alpha}$	$c_{l\delta_e}$	$c_{M\delta_e}$	$c_{y\beta}$	$c_{L\beta}$	$c_{N\beta}$	$c_{y\delta_r}$	$c_{L\delta_r}$	$c_{N\delta_r}$	$c_{L\delta_{a2}}$
0.3	5.65	-1.876	0.455	-1.895	-0.66	0.255	-0.259	-0.304	0.082	-0.336	0.167
0.5	5.896	-1.973	0.47	-1.976	-0.68	0.26	-0.274	-0.315	0.0849	-0.349	0.172

Table 3 stability and control derivatives, elastic model

Mach	$c_{l\alpha}$	$c_{M\alpha}$	$c_{l\delta_e}$	$c_{M\delta_e}$	$c_{y\beta}$	$c_{L\beta}$	$c_{N\beta}$	$c_{y\delta_r}$	$c_{L\delta_r}$	$c_{N\delta_r}$	$c_{L\delta_{a2}}$
0.3	5.67	-1.868	0.455	-1.895	-0.662	0.254	-0.26	-0.305	0.082	-0.337	0.167
0.5	4.74	-1.484	0.504	-1.636	-0.649	0.244	-0.243	-0.29	0.0956	-0.327	0.0999

Table 4 level flight trim solutions

Mach = 0.3	$\alpha\ [deg]$	$\delta_e\ [deg]$
Rigid	4.95	-5.694
Elastic	4.92	-5.64

Mach = 0.5	$\alpha\ [deg]$	$\delta_e\ [deg]$
Rigid	0.535	-1.20
Elastic	1.245	-2.016

Table 5 pull up trim solutions

Mach = 0.3	$\alpha\ [deg]$	$\delta_e\ [deg]$
Rigid	15.09	-16.13
Elastic	15.02	-16

Mach = 0.5	$\alpha\ [deg]$	$\delta_e\ [deg]$
Rigid	4.03	-4.84
Elastic	5.73	-6.30

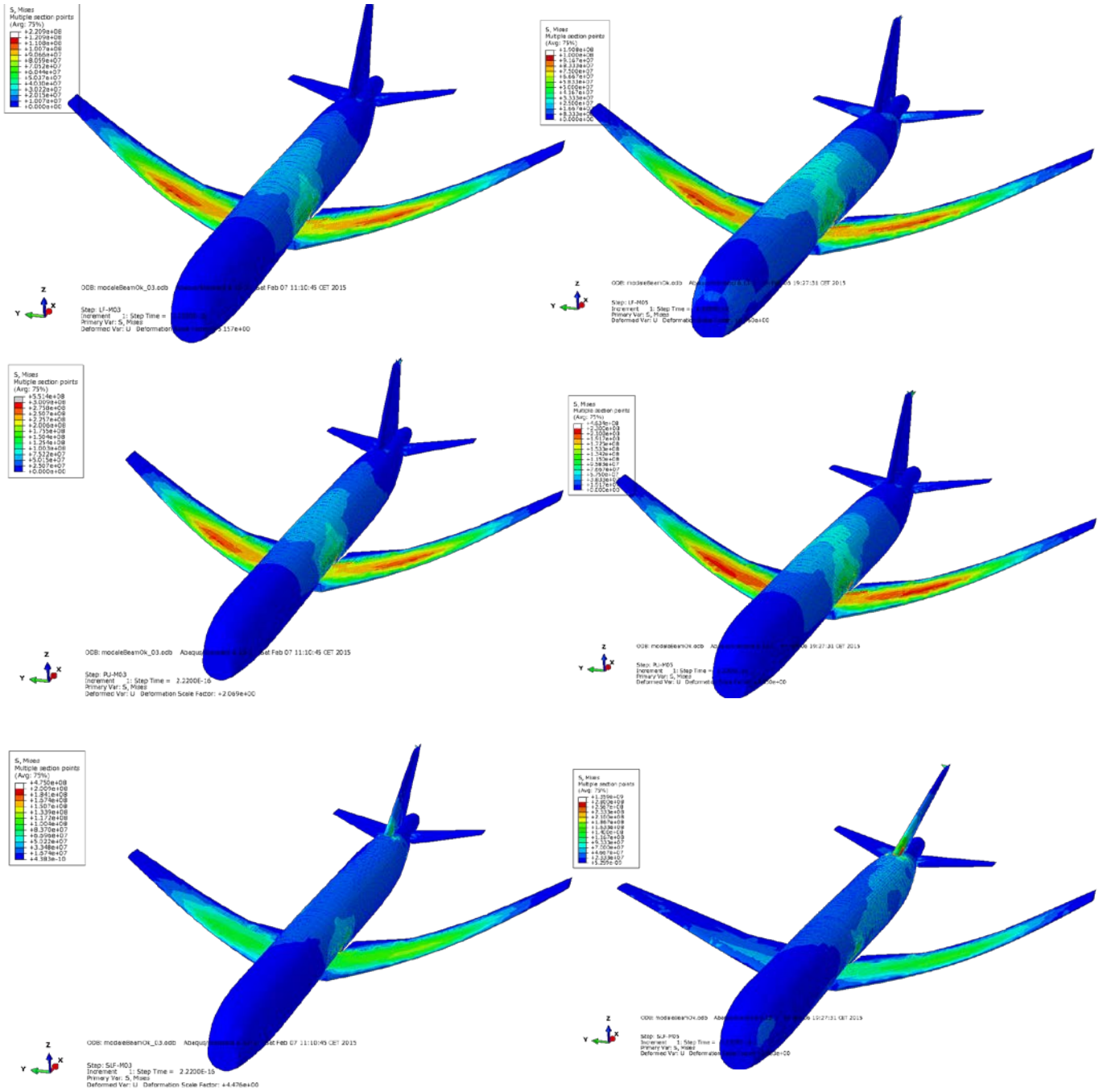
Table 6 Sideslip trim solutions

Mach = 0.3	α	δ_e	δ_{a2}	δ_r	$\ddot{y}$
Rigid	4.95	-5.70	-11.56	-7.59	-1.17
Elastic	4.95	--5.64	-11.517	-7.60	-1.17

Mach = 0.5	α	δ_e	δ_{a2}	δ_r	$\ddot{y}$
Rigid	0.535	-1.20	-11.31	-7.74	-3.30
Elastic	1.245	-2.013	-17.65	-7.059	-3.21

Figure 12 Trim deformations and Von-Mises stress distributions

Level flight, pull-up and sideslip for Mach 0.3 (left) and Mach 0.5 (right)

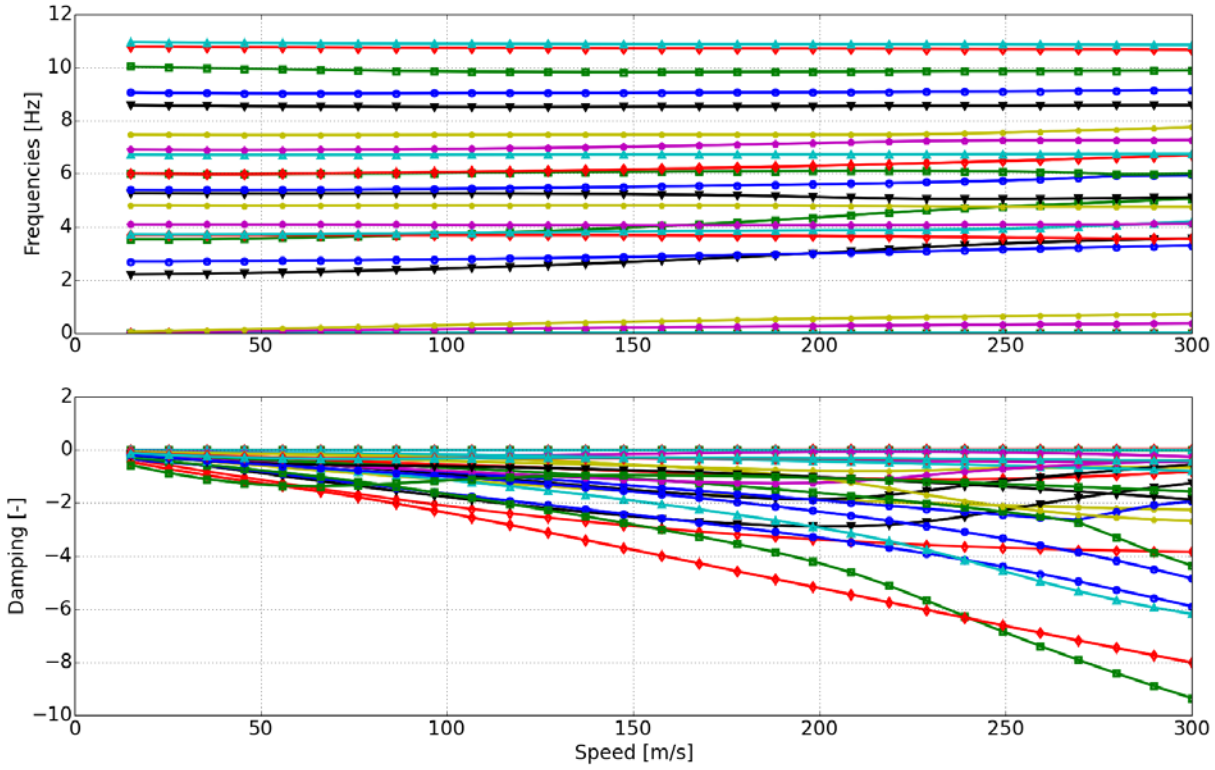


Dynamic Analysis

Before running the dynamic responses, the flutter analysis is computed at Mach 0.5. The analysis is performed with fixed movable surfaces. Figure 13 shows the flutter diagram. Once checked that the model has no instabilities, different dynamic analysis are performed, two of them are reported:

1. Discrete gust condition, $v_g = 15(1 - \cos(3\pi t))\ [m/s]$, Mach=0.5;
2. Elevator rotation, $\delta_e = 15\sin(2\pi t)\ \ [deg]$, Mach=0.5.

Figure 13 Flutter Diagram at Mach = 0.5



The output requested are:

- The wing root internal loads,
- The horizontal tail root internal loads,
- The internal loads of the fuselage section after the wing.

All the internal loads are computed trough the summation of forces technique.

For each solution, two time steps are selected; for the gust analisys the first one corresponds to the time when the load factor reaches the maximum value and the second one when the bending moment of the wing is maximum. For the elevetor manouver the first time step corresponds to the time when the banding moment on the horizontal tail is minimum and the second one when the bending moment of the wing is maximum. In these time steps the nodal loads, both aerodynamic and inertial ones (computed using the nodal mass matrix) of the whole aircraft are exported and an Abaqus steady solution is executed.

Figures 13 - 16 show the time history of the outputs for the gust analysis and Figure 17 shows the deformations and the Von-Mises stress obtained at the two selected time steps.

Regarding the elevator maneuver, Figure 18 - 21 report the time history of the outputs and Figure 22 shows the deformations and the Von-Mises stress obtained at the two selected time steps.

Figure 15 Gust Condition: CG responses

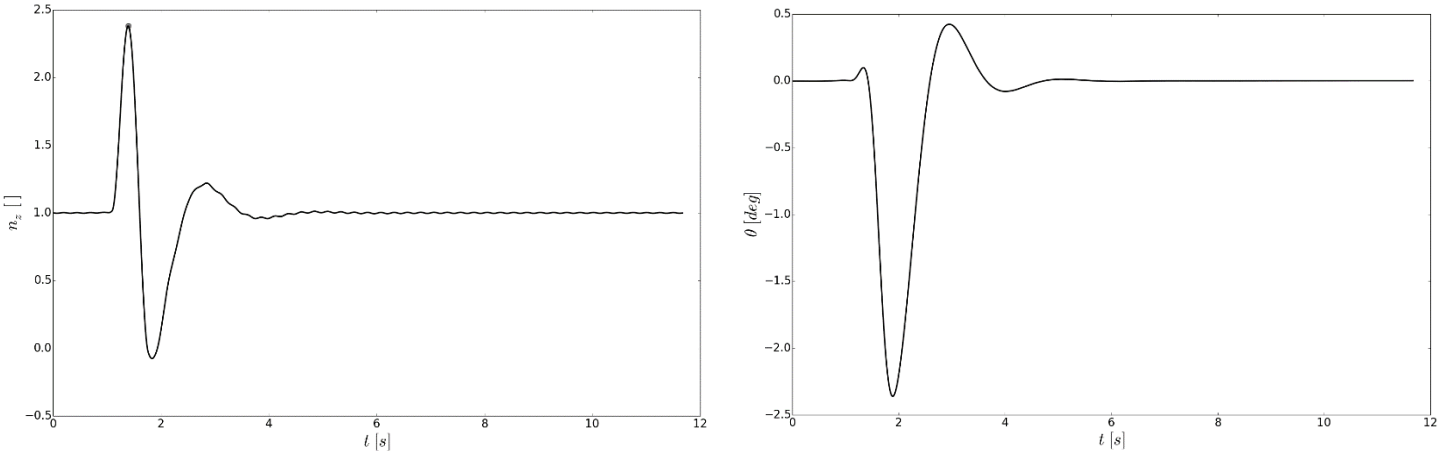


Figure 14 Gust Condition: Wing root responses

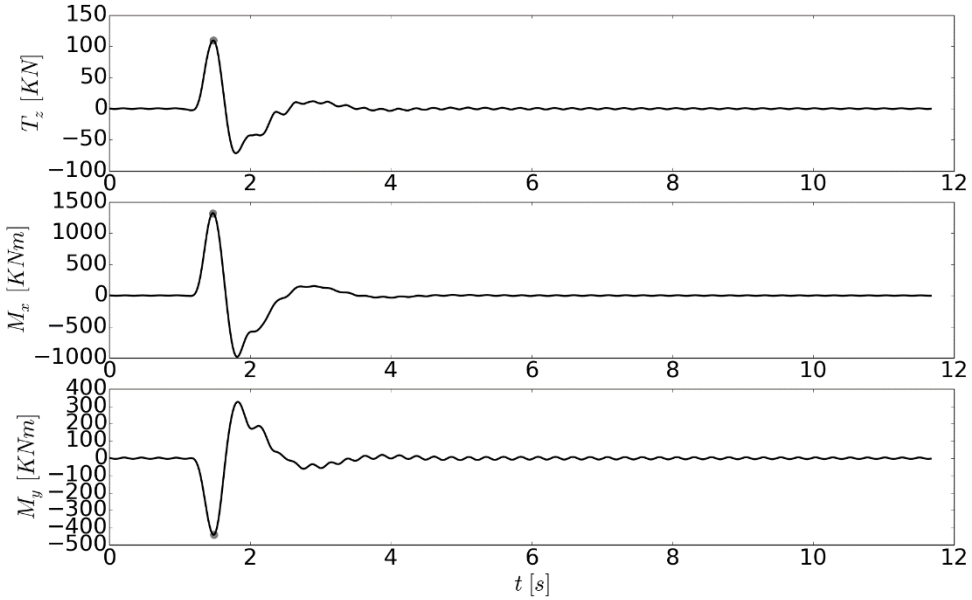


Figure 16 Gust conditions: Horizontal tail responses

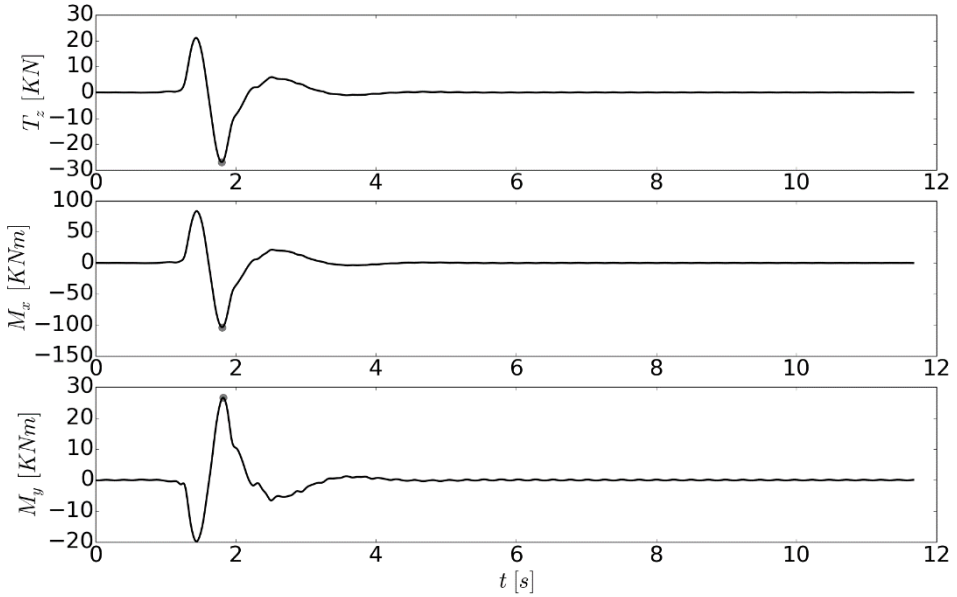


Figure 18 Gust conditions: Fuselage responses

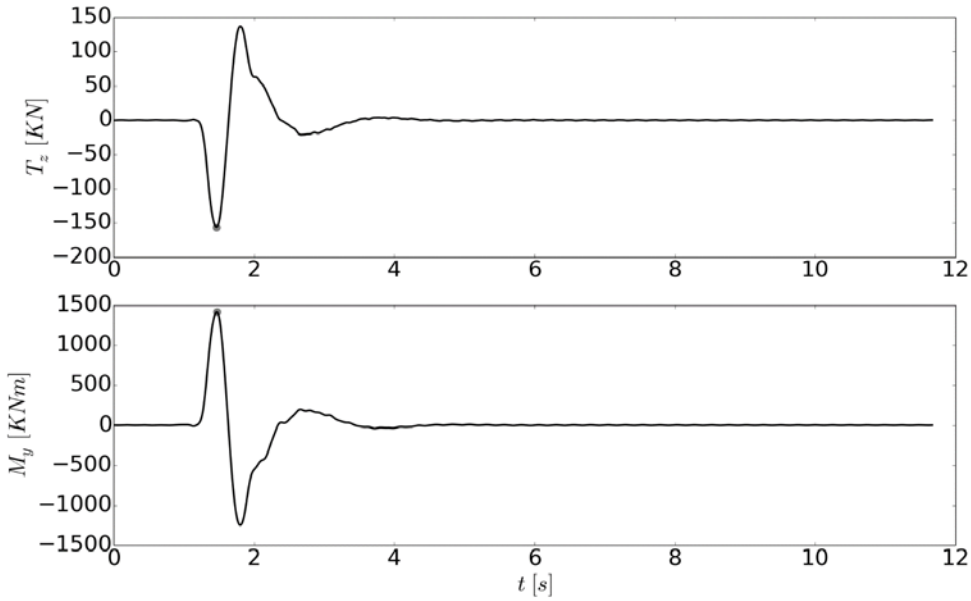


Figure 17 Gust condition: deformation and Von-Mises stress for maximum load factor (left) and maximum wing bending moment (right)

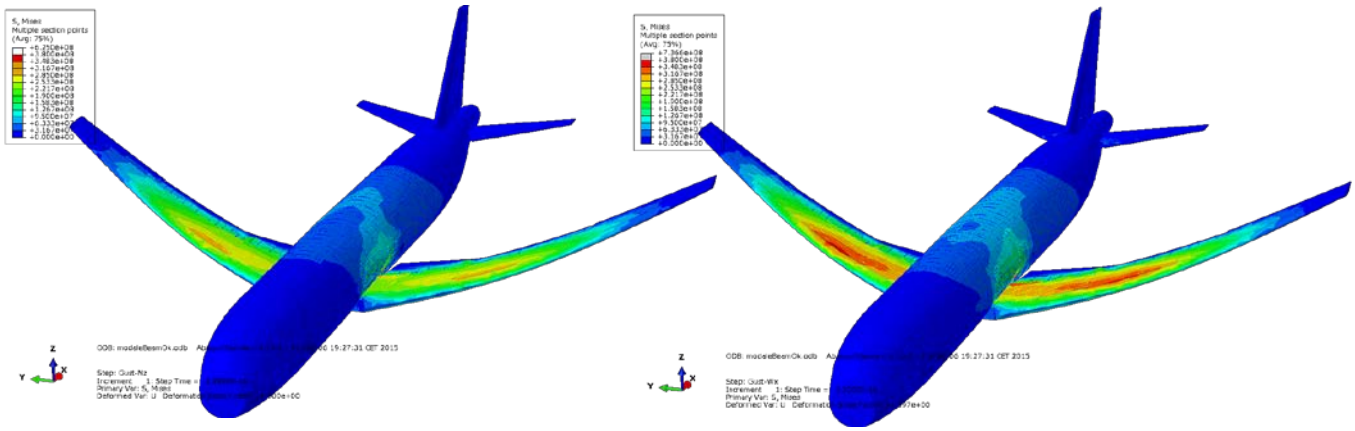


Figure 19 Elevator Manouver: CG Responses

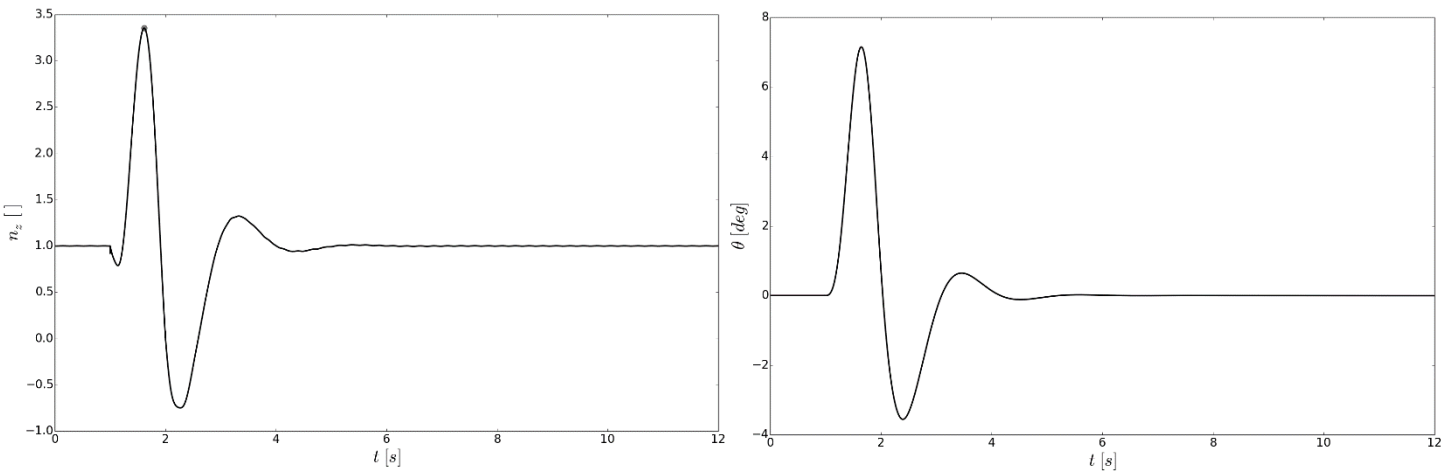


Figure 20 Elevator Manouver: Wing root responses

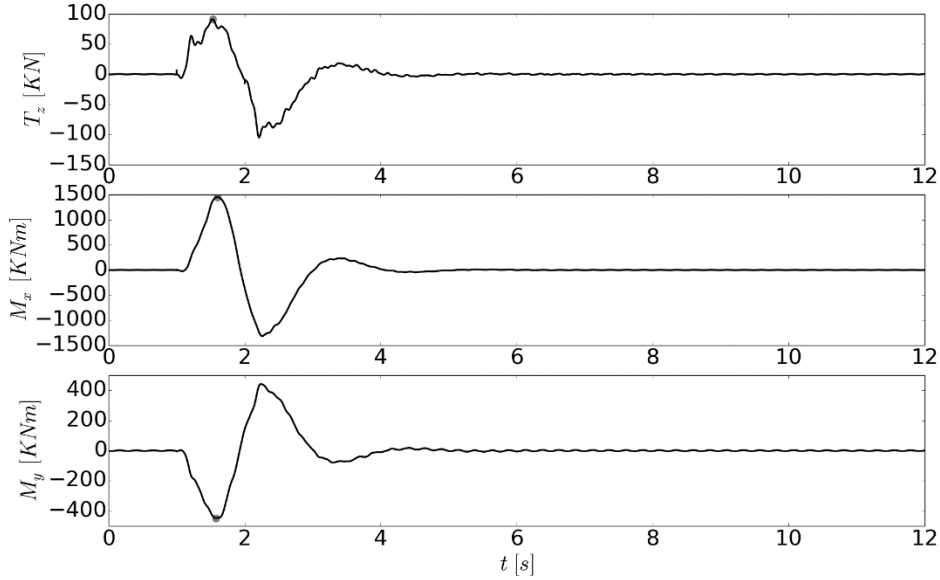


Figure 21 Elevator Manouver: Horizontal tail responses

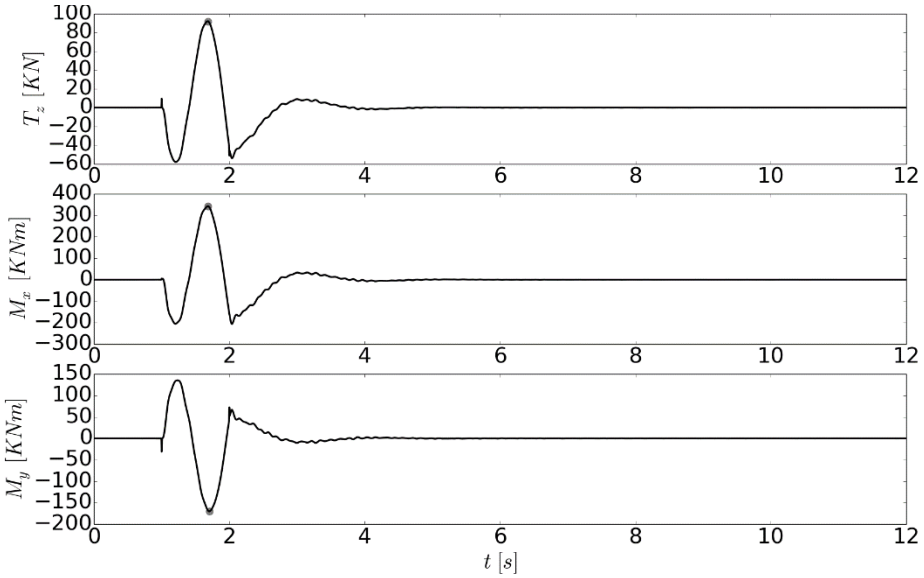


Figure 22 Elevator Manouvre: Fuselage responses

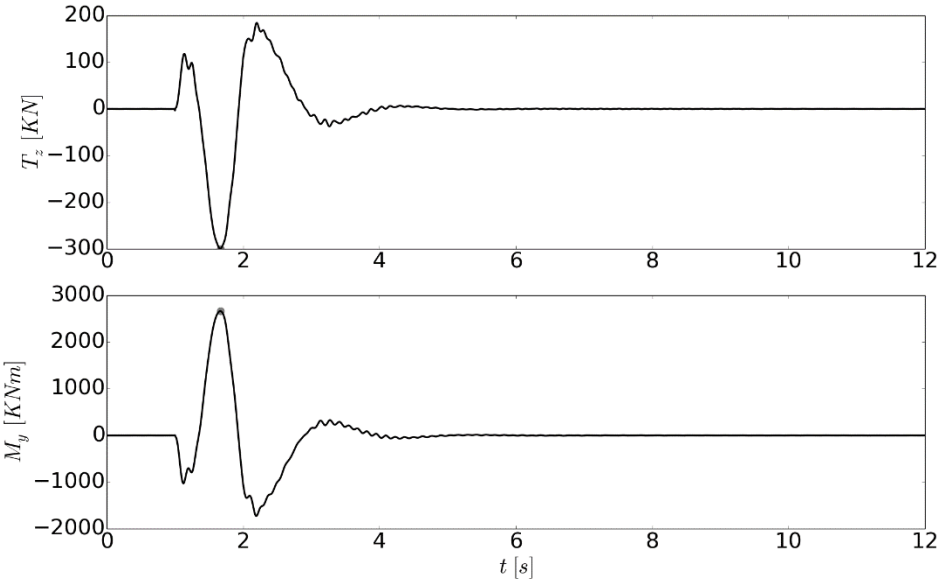
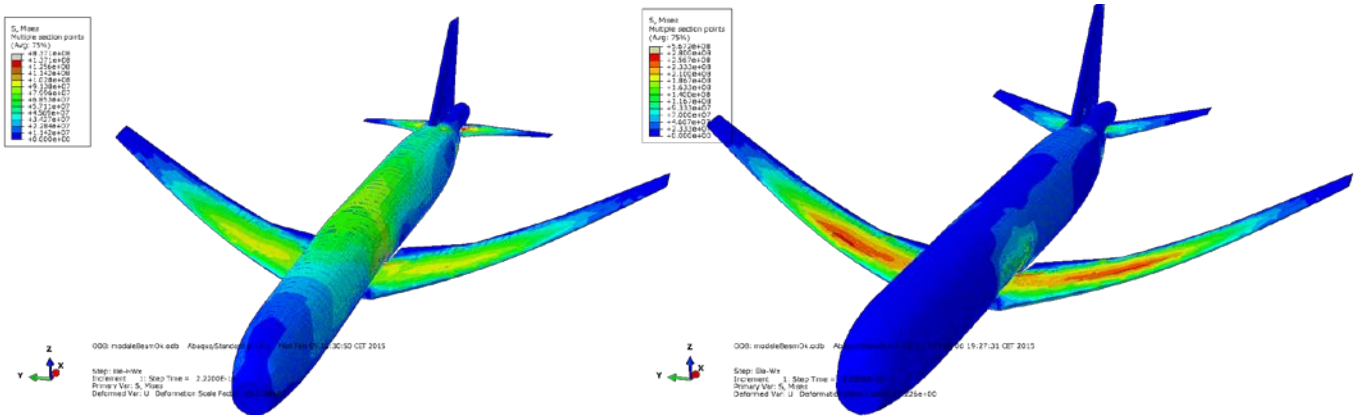


Figure 23 Elevator Manouver: deformation and Von-Mises stress for minimim bending moment on the horizontal tail (left) and maximum wing bending moment (right)



Conclusion

This work shows the definition of a new framework designed to be able to perform all the different analysis of the preliminary design managing quite big databases both for structural and for aerodynamic models. The tool is able to communicate with Abaqus in a very easily why, in order to take advantage of one of the most powerful CAD/CAE software. As shown by the test case, the tool at the moment is able to define complete and complex models and to perform different aeroelastic simulations to get responses and loads useful for the aircraft design. Different new features are programmed both for the mid-term and for the long-term time. Regarding the mid-term:

- At the moment is under development the coupling of the aeroelastic state space model with the multi-body solver MBDyn (Masarati, 2014), in order to be able to compute landing conditions and therefore landing loads on the whole aircraft;
- The second feature that will be added is a smart tool able to define all the load cases and analysis requested by the regulations (EASA-CS23 and EASA-CS25) starting from few inputs, like the aircraft class and the flight envelope.

Regarding the long-term development:

- A tool able to identify the design load cases for the different structures once all the load cases are defined and computed;
- Knowing the design load cases the sizing process can be defined trough an optimization problem. Different optimization tools are under study to identify the best solution and approach.

Acknowledgments

The authors are very grateful for the help given by Dieter Kohlgrueber (DLR) both for the CPACS model he

provided us and used for the test cases shown in the paper, and for the help given to understand the fuselage and the wing-fuselage interface CPACS parameterization.

References

Cavallaro,sh R. (2008). A code for surface modeling and grid generation coupled to a panel method for aerodynamic configuration design. Master's thesis, Universitá degli Studi di Pisa.

Daoud, F., Petersson, O., Deinert, S., and Bronny, P. (2012). Multidisciplinary airframe design process: Incorporation of steady and unsteady aeroelastic loads. In 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSM.

Deinert, S., Petersson, O., Daoud, F., and Baier, H. (2013). Aircraft loft optimization with respect to aeroelastic lift and induced drag loads. In 10th World Congress on Structural and Multidisciplinary Optimization.

DLR (2013). Cpacs – a common language for aircraft design: http://code.google.com/p/cpacs/.

Dorbath, F., Nagel, B., and Gollnick, V. (2011). A knowledges based approach for automated modelling of extended wing structures in preliminary aircraft design. In 60. Deutsche Luft- und Raumfahrtkongress.

Dorbath, F., Nagel, B., and Gollnick, V. (2012). Implementation of a tool chain for extended physics-based wing mass estimation in early design stages. In 71th Annual Conference of Allied Weight Engineers, Inc.

Harder, R. L., Rodden, P., and Bellinger, E. D. (1979). Aeroelastic addition to nastran. CR 3094,NASA.

Hurlimann, F. (2010). Mass Estimation of Transport Aircraft Wingbox Structures with a CAD/CAE-Based Multidisciplinary Process. PhD thesis, ETH Zurich.

Karpel, M., Moulin, B., and Chen, P. C. (2005). Dynamic response of aeroservoelastic systems to gust excitation. Journal of Aircraft, 42(No. 5).

Klimmek, T. (2013). Development of a structural model of the crm configuration for aeroelastic and loads analysis. In IFASD 2013, International Forum on Aeroelasticity and Structrual Dynamics.

Ledermann, C., Ermanni, P., and Kelm, R. (2006). Dynamic cad objects for structural optimization in preliminary aircraft design. Aerospace Science and Technology, Vol. 10.

Ledermann, C., Hanske, C., Wenzel, J., Ermanni, P., and Kelm, R. (2005). Associative parametric cae methods in the aircraft pre-design. Aerospace Science and Technology, Vol. 9.

Maierl, R., Petersson, O., and Daoud, F. (2013). Automated creation of aeroelastic optimization models from a parameterized geometry. In IFASD 2013, International Forum on Aeroelasticity and Structrual Dynamics.

Masarati, P. (2014). Mbdyn - free multibody dynamics simulation software: www.mbdyn.org.

Moerland, E., Zill, T., Nagel, B., Spangenberg, H., Schumann, H., and Zamov, P. (2012). Application of a distributed mdao framework to the design of a short- to medium-range aircraft. In Deutscher Luft-und Raumfahrtkongress.

Morino, L., Chen, L.-T., and Suciu, E. O. (1975). Steady and oscillatory subsonic and supersonic aerodynamics around complex configurations. AIIA Journal, Vol. 13(No. 3).

Morino, L. and Kuo, C.-C. (1974). Subsonic potential aerodynamics for complex configurations: A general theory. AIIA Journal, Vol. 12(No. 2).

MSC.NASTRAN. Aeroelastic analysis user's guide.

Osterheld, C. M., Heinze, W., and Horst, P. (2000). Influence of aeroelastic effects on preliminary aircraft design. In ICAS.

Parrinello, A. and Mantegazza, P. (2010). Independent two-fields solution for full-potential unsteady transonic flows. AIAA Journal, Vol. 48.

Parrinello, A. and Mantegazza, P. (2012). Improvements and extensions to a full-potential formulation based on independent fields. AIAA Journal, Vol. 50.

Ricci, S. (2013). Neocass suite homepage: http://www.neocass.org/.

Ripepi, M. (2014). Model Order Reduction for Computational Aeroelasticity. PhD thesis, Politecnico di Milano.

Ripepi, M. and Mategazza, P. (2013). Improved matrix fraction approximation of aerodynamic transfer matrices. AIAA Journal, 51(No. 5).

Rocca, G. L. (2011). Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization. PhD thesis, Technische Universiteit Delft.

Scherer, J., Kohlgruber, D., Dorbath, F., and Sorour, M. (2013). A finite element based tool chain for structural sizing of transport aircraft in preliminary aircraft design. In CEAS 2013.

Tarkian, M. and Tessier, F. J. Z. (2007). Aircraft parametric 3d modelling and panel code analysis for conceptual design. Master's thesis, Division of Machine Design, Linkoping University Institute of Technology.

The University Of Tennessee. Plasma homepage: http://icl.cs.utk.edu/plasma/.

Vescovini, R. (2011). Analytical formulation for buckling and post-buckling analysis and optimization of composite stiffened panels. PhD thesis, Politecnico di Milano.

Werner-Westphal, C., Heinze, W., and Horst, P. (2008). Multidisciplinary integrated preliminary design applied to unconventional aircraft configurations. Journal of Aircraft, Vol. 45(No. 2).
Yates, E. C. J. (1987). Agard standard aeroelastic confgurations for dynamic response i—wing 445.6. NASA, TM 100492.
Yates, E. C. J., Land, N. S., and Foughner, J. T. (1963). Measured and calculated subsonic and transonic flutter characteristics of a 45deg swepteback wing planform in air and in freon-12 in the langley transonic dynamics tunnel. NASA, TN D-1616.