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

Department de Resistencia de Materials

Grau en Enginyeria en Tecnologies Aeroespacials

REPORT
Study of the model-order reduction of the aerolastic behavior of a wing

Final Degree Thesis of:
Rodeja Ferrer, Pep

Director:
Prof. Joaquin Hernández Ortega

Co-director:
Prof. Juan Carlos Cante

June 2016

[This page is intentionally left blank]

Abstract

The main objective of this paper is to apply the model-order reduction technique to an airplane's wing in order to speed up development of aircrafts or to get real-time results of a plane structural state. However, this case is especially complex since the wings are an aeroelastic problem where both fluid and structure must be computed in order to get realistic results.

In order to improve the overall airplane design speed -in addition to the usage of MOR techniques- a complementary software has been developed. This is a parametric software capable of quickly generating a geometry and exporting it to simulate both the fluid and the structure with a FE software like Kratos. This software will be open sourced.

The usage of the custom software helps to generate geometries that differ only on a single design parameter (the angle of attack in this paper). These different geometries are then processed with Kratos to obtain the high-fidelity result from each one of them.

Once the high-fidelity snapshots have been obtained (five are used in this paper), the reduced order models are generated using a discrete version of the Proper Orthogonal Decomposition (POD) called Single Value Decomposition (SVD). Finally, using the discrete empirical interpolation method (DEIM), it is possible to interpolate between the simulations and obtain the results of any intermediate state in less than a second without having to perform the full simulation.

No physical model has been constructed to compute the fluid and only statistical methods are employed for that part.

The results turned out to be very precise regarding the structure ROM; all the same, the only statistical approach to the fluid proved to be not ideal and the accuracy error remained around 15% for this part.

Acknowledgments

Thanks to everyone who has helped with this project and has supported me, including:

Joaquín A. Hernández Ortega

Riccardo Rossi

Eduardo Soudah

Stefano Zaghi

Ilaria Iaconeta

Users of the Kratos forum

My parents and sibilings

Contents

List of Figures

List of Tables

1 List of abbreviations

Ad	Addresses matrix (for assembling the K and M matrices)
CN	Connectivities matrix
$COOR$	Coordinates matrix
d_l	Vector of free nodal displacements
M_{d_l}	Matrix of free nodal displacements
DOF	Degrees of freedom
$DOFi$	Degree of freedom i of the nodeI-nodeJ combination
$DOFj$	Degree of freedom j of the nodeI-nodeJ combination
F_{ll}	Nodal force vector of the degrees of freedom (right hand side)
F_e	Elemental nodal force vector (right hand side)
F	Nodal force vector (right hand side)
FE	Finite elements
M_F	Matrix of nodal forces
K_{ll}	Stiffness matrix of the degrees of freedom
K_e	Elemental stiffness matrix
K	Stiffness matrix

M_{ll}	Mass matrix of the degrees of freedom
M_e	Elemental mass matrix
M	Mass matrix
MOR	Model order reduction
$nDOF$	Number of degrees of freedom
$nNodes$	Number of nodes
$NodeI$	Node i
$NodeJ$	Node j
P	Pressures vector
M_P	Matrix of pressures
POD	Proper orthogonal descomposition
ROM	Reduced order model
SVD	Singular value descomposition

2 Introduction

2.1 Justification of the need

The first phases of a plane design are crucial. During these phases, a fast iteration among different models is critical and; usually, different simulations must be carried out changing a design variable to determine its optimal value.

There are two significant improvements over a typical CAD-and-simulation approach (where a new geometry needs to be generated and then simulated each time independently) that can be made.

1. First of all, the usage of a parametric design reduces time spent on the CAD.

2. Secondly, a reduced order problem can be formulated to diminish the number of high-fidelity simulations. The usage of reduced order models would speed up significantly the design process and produce more accurate results than using simple linear interpolations between high fidelity simulations.

Apart from this use case, there is also a need for high-speed simulations on boards planes [3]. This usage fits in an increasingly important paradigm of a dynamic data-driven application system (DDDAS) where planes are

designed to ingest and process as much information as possible to automatically make informed decisions about its own operation. This is gaining interest in a world where drones are increasingly important and autopilots need to improve to increase the overall industry security.

2.2 Aim

The aim of the paper is to implement custom software to tackle each improvement mentioned in the justification section. The software used to solve the first problem will be open-sourced and made available so it can be used for free. All the same, the paper is more focused on the MOR implementation, and its benefits and applications.

Finally, after the ROM is constructed, the project aims to extract useful information from it and analyze the reliability of the data. Then, it must be determined if the MOR technique is a great fit for airplane development and onboard plane systems.

2.3 Scope

The project scope is limited by the amount of time available. The main focus of attention is the construction of a great structural ROM followed by a great implementation of the CAD software. These are the two key parts that would improve the speed of a plane's design.

To do that some HF simulations are generated and a statistical ROM of the fluid has been generated. Those parts of the project are necessary in order to achieve the main goal; nevertheless, an accurate study of those topics is out of scope.

2.4 Requirements

This project is required to:

1. build an accurate model of the wing structure of a plane with typical aerodynamic loads;
2. implement a flexible parametric wing design software;
3. and determine if these techniques are a great fit for airplane development and onboard plane systems.

build an accurate model of the wing structure of a plane with typical aerodynamic loads.

2.5 Introduction to model order reduction

Generally speaking, model order reduction refers to any endeavor aimed at constructing a simpler model from a more complex one. The simpler model is usually referred to as the reduced order model (ROM), while the more complex one is termed the full-order or high-fidelity model. This full-order model may be, for instance, —as is the case here— a finite element (FE) model. These two distinct models are computed at the different stages called offline for the high-fidelity model and online phase for the ROM.

Given the simplicity of the model used in the ROM, the solutions obtained by this process are only approximations; however, the simulation time is reduced by several orders of magnitude. As stated, this speed bump can be used for a faster design or -since the computational requirements are not too high- for real-time simulation onboard a real plane. This would

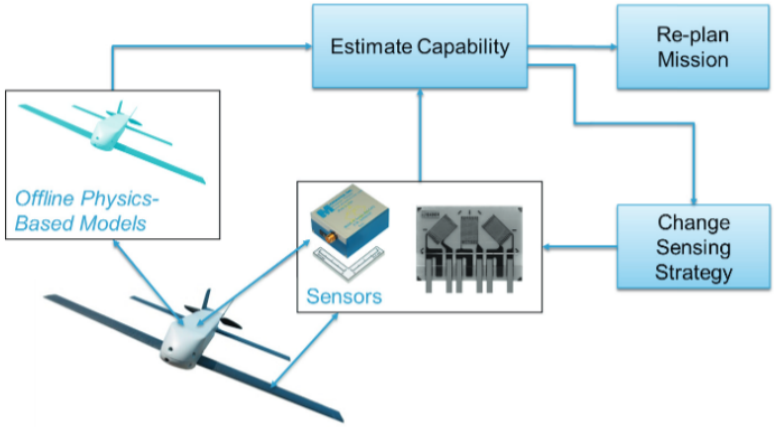


Figure 1: Application of ROM in a self-aware plane capable of modifying it is behaviorbased on the online simulation results given by the ROM. Image from [3]

enable the planes to be self-aware and dynamically adapt its behavior. In Figure 1 an example of the implementation of this technology on a drone is outlined. In that case, the results from the online simulations are used to estimate the abilities of the plane in real-time and let it re-plan the current task if necessary.

In this paper, ten HF simulations are conducted to generate the ROMs (half fluid simulations and half structural simulations). All these simulations are conducted using the multi-physics and open-source software Kratos[1]. These simulations correspond to the following angles of attack: -3º, 0º, 3º, 6º, and 9º. Additionally, four HF simulations are used to check the accuracy of the results obtained with the ROMs. These additional simulations correspond to the angles of attack of -1.5º, 1.5º, 4.5º and 7.5º.

The available MOR techniques are described in the state of the art chapter.

[1]Kratos is developed by CIMNE: http://www.cimne.com/kratos/

3 State of the art

In this section the most recent and common techniques used in the science community are explained. This section also gives some background and explains the methods used in the following parts of the paper.

3.1 Finite elements

To follow the paper, some basic knowledge of the finite elements method (FE) is required. In the following section, the basic concepts are outlined.

The FE method is a numerical technique for finding approximate solutions to boundary value problems. The method subdivides a large problem into a set number smaller parts called finite elements. Each single element is modeled by a simple set of equations that are later assembled into a larger system of equation that models the entire problem. The FE method uses variation methods to approximate a solution by minimizing an associated error function.

Given a differential equation f such that $f \in \Omega$ and $\Omega \in \mathbb{R}$, two sets of functions must be defined.A space of trial function in contained in $\bar{\Omega}$ and whose derivate is piecewise continuous and bounded on $\bar{\Omega}$. And a space of variations -or test functions. The trial functions are required to satisfy a boundary condition (f.e. $u(1) = g$) while the variations are required to satisfy the homogeneous counterpart of the boundary condition (f.e. $v(1) = 0$).

The weak formulation of the boundary problem is obtained by taking the product of the differential equation to solve with the test function and integrating over the domain. This results in the variational equation, in mechanics, this equation is referred to as the equation of virtual work. This equation is still exact to the initial problem since it has yet to be discretized.

To discretize the problem, a finite-dimensional approximation of both spaces (trial and test) must be constructed using the Galerkin's method. These spaces are defined in such way that $S \subset S^h$ where S is the space where the exact solution is and S^h is the subspace of the discretized solution.

Using a set of shape functions that are specific to the element shapes is then possible to approximate the discretized problem to the exact solution. This method will generate a set of equations that can be written in the following form:

$$Kd = F \tag{3.1}$$

where d is the vector of nodal solutions, K is specific to the problem and F is usually a function of the boundary conditions.

In order to calculate the matrix K, a set of integrals over the domain of each element must be computed. The implementations of the FE method usually compute the K_e matrix per element instead of the total K matrix. For this reason, there is an additional operation to be computed called the assembly; this operation is represented by the following equation:

$$K = A_{e=1}^{n_e l} K_e = \sum_{e=1}^{n_e l} L^{eT} K^e L^e \tag{3.2}$$

where A is the so-called assembly operator and L^e is a $2 \times (n+1)$ matrix constituted of 0 and 1s that relates element nodal variables d^e with the global nodal vector d.

In the particular case of structural analysis, the direct stiffness matrix is commonly used to solve complex problems. This method is also the one used in this paper to address the structural part, additionally, parts of this theory will later be used in the reduced order model. In this method, the previously mentioned matrix K is called the stiffness matrix, and the solutions are the nodal displacements.

As explained, the method discretizes the domain into smaller pieces called elements; however, subdivisions itself must be introduced by the user using what's referred to as a mesh. This mesh consist of a series of nodes -points in the domain- and elements that result of the union of several nodes.

The mesh is typically stored in two matrices, one of which is the coordinates matrix (COOR) that stores each coordinate of each node. For a problem defined in $\mathbb{R}^n$ and with $nNodes$, the dimensions of the COOR matrix would be $n \times nNodes$. The other matrix is the connectivities matrix (CN) that stores the nodes of each element, one element per row. So, if the mesh has triangular elements (three nodes per element) and n elements, the CN matrix would have a size of $n \times 3$.

3.2 Model order reduction

Model reduction strategies may range from purely physical insight-based approaches, that rely on experimental observations and analytical simplifications and, hence, are highly contingent upon the physical intuition, depth

of insight, and knowledge of the modeler; to black-box methods, that allow to construct simpler models in a generic and systematic manner (by artificial neural networks, for instance), but are somehow "agnostic" to the underlying physics. The focus of the project is in a class of model reduction techniques that lies somehow between these two extremes, combining advantageous features of both: the projection -(or reduced basis)- based methods. As stated, projection-based methods employ a previously computed —in an offline stage— set of state solutions (snapshots) to generate a relatively low-dimensional basis whose corresponding subspace intends to approximate, in a certain sense, the full-order solution space; then, the governing equations are projected onto this reduced-order subspace, resulting in a model with a significantly reduced number of degrees of freedom —the reduced-order model(ROM)— that is solved in the online stage. The method that combines the proper orthogonal decomposition (POD) and the Galerkin projection is, arguably, the most popular model reduction technique in the computational mechanics' community.

The SVD implementation in MATLAB returns three matrices, U, S and V; called left singular vectors (or modes matrix), singular values, and right singular vectors respectively. It is possible to reconstruct the exact solution using the formula $A = U * S * V^T$. All the same, in this paper only the modes and the singular values will be used. The modes are the representations of the different states that the system takes, and the singular values represent how important each mode is.

In this section, to explain the different methods, examples will be given on how to compute F, the vector of FE nodal internal forces. However, the same concepts can be -and will be- applied to the pressures in the boundary conditions (P) and the vector of FE nodal displacements at the unrestricted nodes (d_l).

Approximation of nonlinear terms

In the general case of governing equations featuring terms that bear a nonaffine relationship with both the state variable and input parameters, the construction of an inexpensive low-dimensional model entails two sequential stages [9], namely: 1) projection onto the reduced basis, and 2) approximation of the nonlinear term. Once a basis matrix for the state variable is available, the projection stage is a standard operation consisting in introducing the approximation of the state variables in the governing equation, and then in posing the resulting equation in the space spanned by the basis vectors. This operation naturally leads to a significant reduction in the number of unknowns and hence diminishes considerably the equation solving effort. However, in a general nonlinear case, the computational cost of evaluating the residual still depends on the size of the underlying finite element mesh —hence the need for a second reduction stage.

In contrast to the first reduction stage, which is more or less standard, the second stage of dimensionality reduction —Ryckelynck [25] coined the term hyper-reduction to refer to it— is far more challenging and remains an issue of discussion in the model reduction community. In the following, we examine the various approaches encountered in the related literature to deal with this additional dimensionality reduction stage.

Classification of "hyper-reduction" methods

Let $F^h \in \mathbb{R}^N$ denote the full-order term bearing a general, nonaffine relationship with both the input variable and the state variable. The corresponding projection onto the reduced order space will be represented by $F \in \mathbb{R}^n (n << N)$, the connection between these two variables being the matrix of basis vectors $\Phi \in \mathbb{R}^{N \times n} (F = \Phi^T F^h)$. Existing approaches for

dealing with the approximation of F can be broadly classified as *nodal vector approaches and integral approaches*.

Nodal vector approximation approaches ("gappy" data)

In this type of approaches, the approximation is carried out by replacing the finite element vector F^h by a low-dimensional interpolant $F^h \approx R_F F_z^h, R_F \in \mathbb{R}^{N \times m}$ being the interpolation matrix, and F_z^h the entries of F^h corresponding to the degrees of freedom ($z \subset \{1, 2...N\}$) at which the interpolation takes place. The interpolation matrix is obtained following the common procedure of computing a basis matrix for F^h, and then determine a set of indices so that the error is minimized over a set of representative snapshots of F^h. This set of interpolation indices can be determined offline using procedures such as the Empirical Interpolation Method (EIM) [7], the Best Points Interpolation Method (BPIM)[21] or the Discrete BPIM[6]. The idea behind this approximated vector approach has its roots in the landmark work of Everson and Sirovich [12] for reconstruction of "gappy" data, and was historically the first proposal for dealing with nonlinear terms in model order reduction; it has been adopted by, among others, [9][10][11]. Alternatively, [25] proposes to bypass the construction of the low-dimensional interpolant and simply solve the balance equations at appropriately selected degrees of freedom (collocation).

Integral approximation approaches

In a finite element context, F can be regarded, not only as a projection of a large vector into a reduced-order space ($F = \Phi^T F^h$), but also as the result of integrating over the concerned domain $\Omega \subset R^d$ (d = 2 or 3) the corresponding reduced-order variable $f = \Phi^T f^h (f^h : \Omega \rightarrow R^N)$, i.e.:

$$F = \Phi^T \int_\Omega f^h d\Omega = \int_\Omega f d\Omega \tag{3.3}$$

Accordingly, the problem at hand can also be viewed as that of approximation of an integral, rather than an approximation of a vector. In turn, this problem can be addressed by:

1. Seeking a low-dimensional approximation of the integrand. This type of approaches follows, in essence, the same procedure described for approximation vector approaches; the difference lies in that, rather than constructing an interpolant for the integral, in this case, it is the integrand that is subjected to approximation via interpolation. This approach followed by [17][5][18][1].

2. Approximating the integral itself as a weighted sum of the integrand evaluated at optimal sampling points, named *Cubature methods*. The first scheme of this type was proposed by [4] in the context of computer graphics applications, and was recently introduced in computational mechanics circles by Farhat and co-workers [15] [14]. Following the classical recipe of Gaussian quadrature of polynomial functions. The method approximates the integral as a finite sum of positive scalar weights $\{\omega_g\}_{g=1}^m$ times the integrand evaluated at appropriately chosen sampling points:

$$F \approx \sum_{g=1}^{m} \omega_g f(x_g) \tag{3.4}$$

3.3 Parametric wing software

When it comes to the state of the art of wing geometry generating software, there is no publicly available software built to accomplish this. There are some general programs like GID that can be used to preprocess data before ingesting into Kratos, but they do not support the creation of parametric geometries like a wing.

On the other side, there are applications like XFLR5 that are specific to planes but, among other limitations, they do not allow the export of the geometry to be simulated using software other than XFLR5 itself.

4 Parametric CAD software

As explained earlier, parametric plane design software exists (like the open source XFLR5). This software, however, has some limitations. For starters, it is open-source, but it is difficult to modify to include non-common design elements because the geometry is coupled with their simulation software and any change would require a reimplementation of the simulation as well. This leads to a second limitation: the impossibility to use third-party simulation software. Finally, it provides no methods to design the structure accurately.

The parametric CAD software was not originally intended to be developed for this project; it has been a solution to a problem encountered during the making of the project. The problem was that using a standard CAD software like AutoCad, SolidWorks or Catia would require a lot of work when generating different geometries during the iteration process of the design. (For example, changing the airfoil of the wing would require changing the external structure and the internal parts shapes). It is true that some of the mentioned software is parametric and that the model can be built so that some changes are automatically performed when modifying a variable. All the same, as the design becomes more complicated, it becomes harder to keep introducing the new parameters as variables.

For that reason, new software was to be implemented. This new program needed to be specific to the airplane case so that it would be powerful for this project.

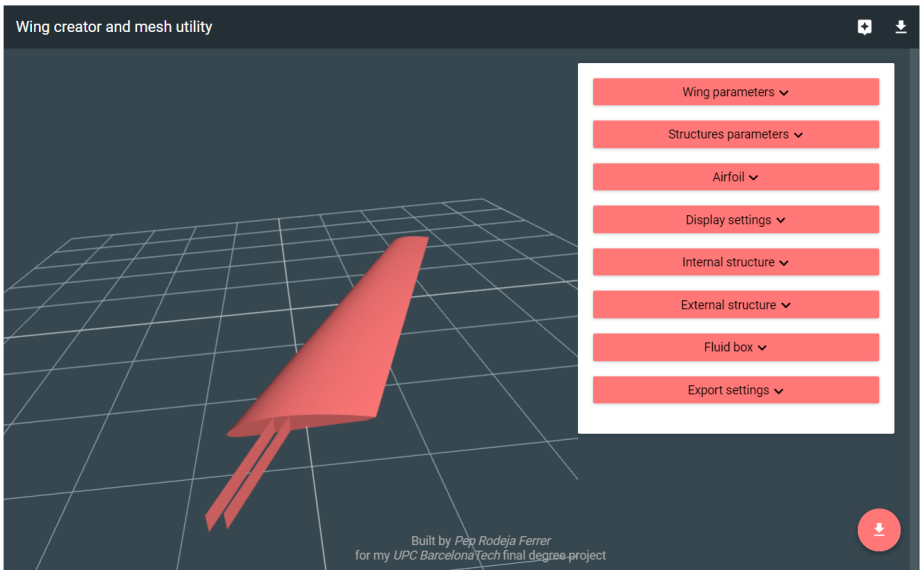


Figure 2: CAD software, main page

4.1 Technologies

The software is implemented using web technologies. This approach has been taken so that the program can be multiplatform easily and does not require an install.

The server is a NodeJs [22] server that serves static content through Express [13]. If the server is run locally, it keeps scanning the airfoils folder so new airfoils can be added without the need to restart the server or client.

The client is implemented using the state of the art framework from Facebook: React [24]. This framework is based on functional programming and handles the rendering of the different components. To manage the data, the new Redux [2] (based on Flux [16]) architecture has been used.

4.2 Software architecture

One of the primary goals of the implementation was that it should be easy to modify; meaning that new parameters should easily be included without modifying the entire software. To accomplish that, the program has been architected with different modules:

- The main core that generates and stores the different elements (vertex, segments, faces and volumes)
- A set of functions used by the core to generate the elements from the parameters (f.e. A function that generates an airfoil with the input data, a different function that changes how the points are distributed along that airfoil, etc)
- A live preview module that takes the data on the core and displays a 3D model on the application.
- An export module that takes the data on the main core and generates a GID project.
- A UI module that allows the user to change the input parameters.

4.3 Software exports files

The export files generated by this software is a GID[1] project with the desired geometry. It is also possible to export the project with a Kratos problem type already defined and some boundary conditions already set.

[1]GiD is a universal pre and post processor for numerical simulations in science and engineering

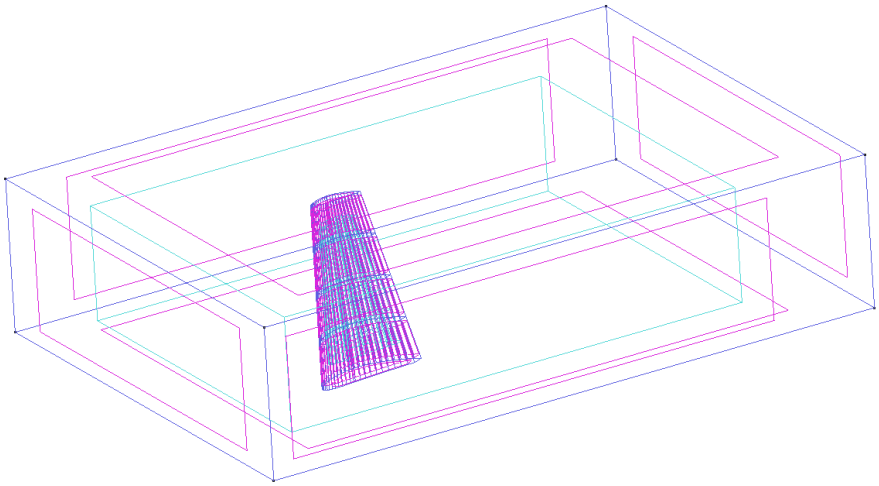


Figure 3: GID main panel showing the wing external geometry and the fluid box

This approach was chosen to enable full flexibility. Exporting to GID means that the geometry can then be meshed using a wide variety of techniques, and it uses the powerful GID mesh algorithms. Also, GID can be used to execute the simulations not only with Kratos but with any major simulation software.

The geometry exported can be selected from the UI. Three groups of geometries can be chosen: internal structure, external structure, and fluid box. This option is provided so that no unnecessary data is generated. For example, if a fluid analysis is to be performed, the internal structure is probably not necessary; in the same way, if a structural analysis is needed, the fluid box would not be required. The Figure 4 shows the internal structure of the wing without the external part nor the fluid box; the Figure 3 shows the fluid box and the wing in GID.

It is important to notice that each geometry group is exported on its own layer. Also, different structures are grouped together, for example,

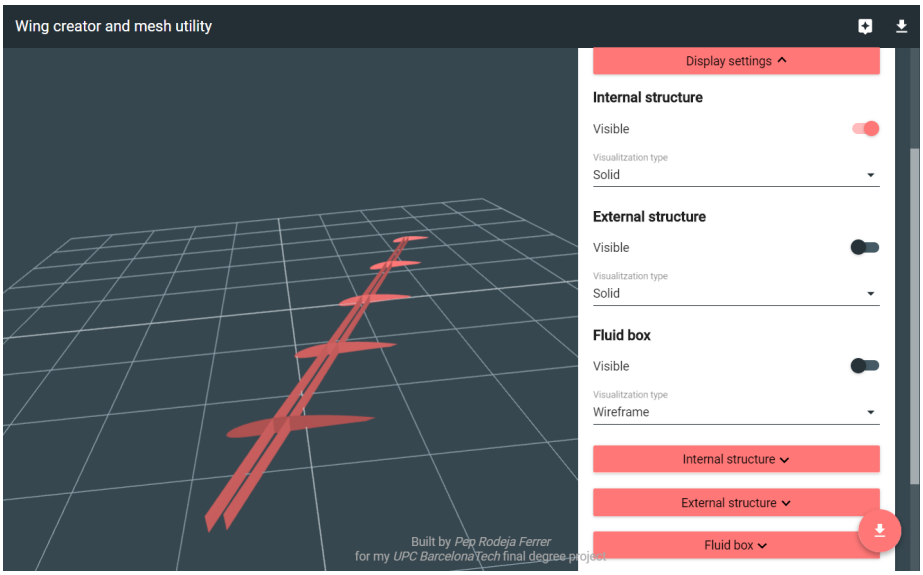


Figure 4: CAD software, external part hidden

each element of a rib (each vertex, segment, and face) is grouped together so users can easily manipulate the geometry from GID.

Types of exports

There are three different types of exports depending on what is intended to do with them.

The first export type is the ***General export*** and it generates a GID project without any problem type or boundary conditions attached. The user will then be able to apply its own conditions and problem type to make any type of computation with any available software.

The second type of export is ***Kratos fluid***. This export attaches a *Kratos fluid* problem type and sets the interior of the fluid box to air. It also sets the wing surface to *no-slip* condition; the laterals of the fluid box

to *is-slip* condition; and the end of the fluid box to *fixed pressure* with 0 bars. The user can then set the inlet velocity from GID and start the fluid simulation.

The third and last type of export is ***Kratos structural***. This export sets the problem type to *Kratos 3D shell* and establishes a *fixed displacement* condition to the root of the wing.

Additionally, the results from the *Kratos fluid* can be imported. If this is done, the software will also apply *fixed pressure* conditions to the exterior of the wing with the results from the fluid simulation.

Finally, if the wing with the same structure and mesh is required but with different pressure conditions (different fluid velocity or angle of attack for example), a mesh can also be imported from GID. This will ensure that the mesh is equal for each structure simulations, freeing GID from this computation. A single mesh across simulations is a requirement to perform a great model-order-reduction analysis.

4.4 CAD software conclusions

All this makes this software unique since it is easy to modify and to incorporate new elements.

An excellent example of this would be how to add a new beam near the back of the wing. In order to do that, it would be necessary to determine the required parameters. In this case, the only parameter is a position variable that will determine where the beam is in a percentage of airfoil chord. No fixed longitudes are used because this would cause problems with wings that change the chord along the span.

After this is determined, the parameter should be added to the UI module. Once done, in the core module, new vertices, segments, and faces must be created on the correspondent positions. Finally, this geometry must be added to the *Internal structure* object.

The view module and the export module will take care of the live render and the export to GID.

5 High-fidelity simulations

As explained, the high-fidelity simulations -using FE in this case- are run by Kratos and the preprocess is done by both custom software and GID. In this paper the design variable selected is the angle of attack, for that reason, multiple simulations with different angles of attack have been performed. For each angle, two simulations are needed, a fluid simulation and a structural one.

The purpose of this paper, however, is not to get exact simulation results that match reality but to demonstrate the speed improvement gained by the usage MOR techniques. As a consequence, more effort has been put into getting the best possible ROM than into getting the best possible FE simulations; therefore, the simulations use simple techniques without taking into account complex considerations.

5.1 Fluid simulation

The simulation of the fluid has been carried out in a fluid box of size $17 \times 12 \times 4$ meters. The root of the wing is coplanar with one side of the box; it is centered vertically; and the border of attack is at 2 meters of the start of the box.

The mesh is composed of tetrahedras and the size of the elements is variable: its smaller near the wing and bigger far away. This can easily be seen on Figure 5 that shows the boundaries of the mesh. The total number

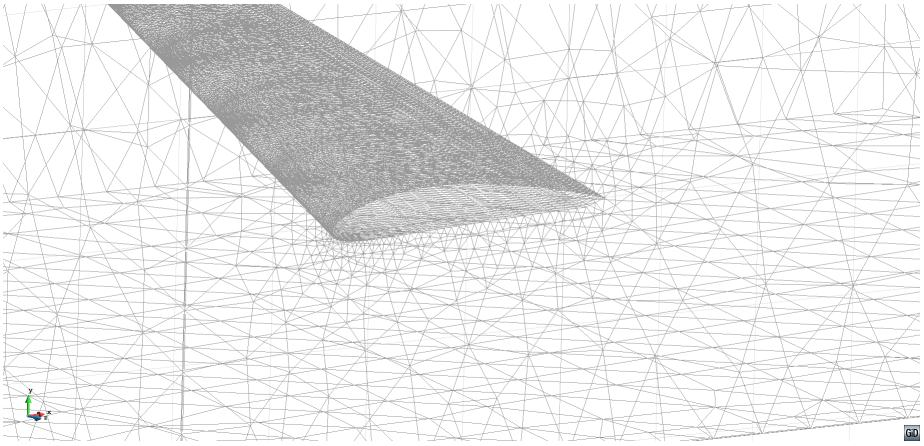


Figure 5: Detail of the boundary mesh around the root of the wing; where it connects with the fluid box.

of elements vary from 143 356 to 95 536 depending on the angle of attack.

The fluid has been simulated using computational fluid dynamics with an eulerian approach. The fluid is Newtonian, incompressible and the properties are set to match air with a viscosity of $1.5 \cdot 10^{-5} m^2/s$.

The boundary conditions are: fixed inlet velocity, outlet pressure to 0, is-slip condition to the fluid box walls, and no-slip condition to the wing surface.

5.2 Structural simulation

Thin shell element types have been selected because their flexibility when designing. Since the goal was to have a parametric design, the shell elements parametric and independent of the geometry width where a perfect match for the project.

The structural mesh uses triangular elements and has 22287 elements

and 11073 nodes. The same mesh is used in every structural simulation.

The material used is an isotropic Aluminum 7021 with a Young Modulus of $72 \cdot 10^9 GPa$, a Poisson of 0.33 and a density of $2780 kg/m^3$.

The boundary conditions are: no displacements on the root nodes and the corresponding pressures to the exterior wing surfaces.

6 Implementation of the model-order reduction

Several multiple implementations have been made in order to obtain different results based on the available data. In that regard, a ROM of the pressures has been implemented in order to calculate the aerodynamic forces around the wing. A different ROM model of the displacements has been assembled to calculate the deformations at any given angle of attack. Finally, a ROM model of the nodal forces (right-hand side) is also required to calculate such deformations at any angle.

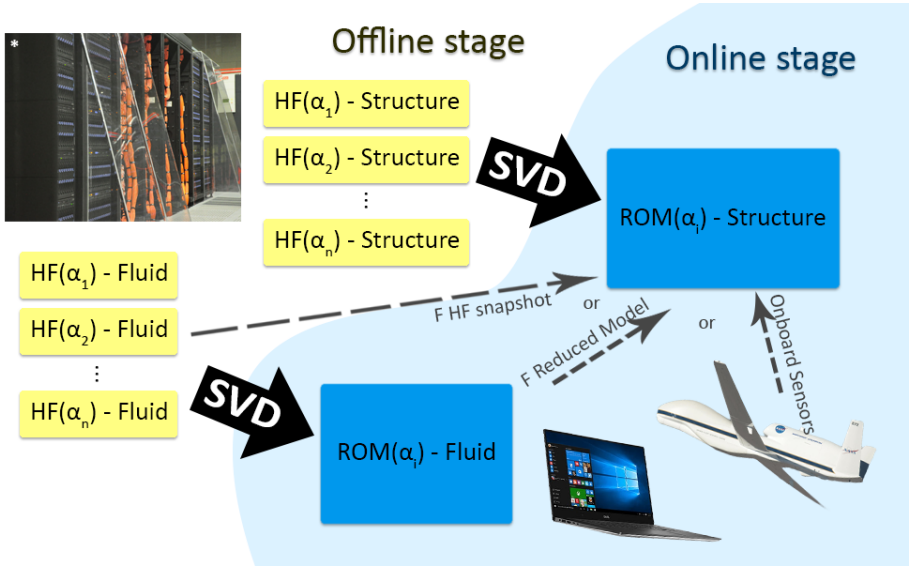


Figure 6: Structure of the ROM implementation. *Cluster image share by the wikipedia user Vcarceler under CreativeCommons license

6.1 ROM of displacements

As explained earlier, to calculate the displacements a combination of statistical methods and physical laws will be employed.

First, it will be necessary to obtain the vectors of displacements from all the snapshots performed at the offline stage. A M_{d_l} matrix will be constructed using one vector per column like $M_{d_l} = [d_{l1}...d_{li}...d_{ln}]$ for n snapshots.

This assembled matrix is used to calculate the svd such that:

$$M_{d_l} = U \times S \times V^T \tag{6.1}$$

Such U matrix -referred to as modes matrix- will have dimensions equal to $nDOF \times nSnapshots$. This matrix is then used to project the stiffness matrix (K) onto the reduced space as seen on the equation 6.2.

$$K^h = U^T \times (K \times U) \tag{6.2}$$

Where K^h is in the reduced space.

Afterwards, the vector of forces F for the required angle of attack is needed. As seen in the Figure 6, this vector can be obtained either by simulating the fluid and assembling it or by constructing a ROM of the fluid and using it to calculate the F vector. In this case, both approaches have been taken to compare the results.

The obtained vector of nodal forces F_c will be projected onto the reduced space as such:

$$F_i^h = U^T \times F_i \tag{6.3}$$

where i is an specific value of the design variable. Note that F is dependant on i but the modes matrix is note.

Taking this projected nodal forces, the projected displacements will be calculated by solving the reduced system as in Equation 6.5

$$d_l^h = (K^h)^{-1} \times F^h \tag{6.4}$$

Finally, the displacements are projected back to the solution space, as seen in the equation **??**

$$d_l = U \times d_l^h \tag{6.5}$$

6.2 ROM of the pressures and Forces

Unlike the displacements, no physical law can be used to approximate the forces or pressures in a simple way. Since no physical law restricts the results, both P and F have been calculated using purely statistical methods. This section will refer to P to indicate the vector of pressures. However, the vector F is calculated in an equivalent way.

First, it will be necessary to obtain the vectors of pressures -or forces- from all the snapshots performed on the offline stage. A Pm matrix will be constructed using one vector per column like $M_P = [P_1...P_i...P_n]$ for n snapshots.

This assembled matrix is used to calculate the svd such that:

$$M_P = U \times S \times V^T \tag{6.6}$$

The returned U matrix -or modes matrix- will be an array with dimensions equal to $size(P) \times nSnapshots$. Therefore, the matrix is still large and the computations may still take significant computation time.

To solve that, a technique called Discrete Empirical Interpolation Method (DEIM) [20] will be used to reduce the model size further. This technique will generate a so-called hyper-reduced model.

The DEIM technique selects as many elements from the P vector as snapshots have been employed; these elements are called z. Afterward, the rows selected will be extracted from the matrix U resulting in a square matrix (U_z) with size equal to the number of snapshots used. Thanks to the use of DEIM, this new matrix is guaranteed not only to be the best possible representation of the higher-order matrix but also to be invertible. This last condition is necessary to be able to calculate the final P vector.

The calculate the matrix U_z, it must be understood that each row of U corresponds to the pressure values of one element such that:

$$U = \begin{pmatrix} P_1^1 & P_2^1 & \dots & P_{n_{modes}}^1 \\ P_1^2 & P_2^2 & \dots & P_{n_{modes}}^2 \\ & & \vdots & \\ P_1^{n_{nodes}} & P_2^{n_{nodes}} & \dots & P_{n_{modes}}^{n_{nodes}} \end{pmatrix} \in \mathbb{R}^{n_{nodes} \times n_{modes}} \tag{6.7}$$

Then, U_z is defined such that:

$$U_z = U_{ij} \ \forall \ i \in z; \ and \ j = 1, \ldots, n_{modes} \tag{6.8}$$

To calculate the P vector at the desired angle of attack, it is necessary to obtain the pressures at the DEIM selected coordinates. If the hyper-reduced model is employed to reconstruct the state of real structures, sensors can be utilized at the determined positions. However, if the hyper-reduced model is being used to speed-up the development, the value of these pressures will be unknown.

This implementation uses simple interpolation between the two nearest points to determine the value of the specified DOFs in the requested angle of attack. This has turned out to be the best solution after some experimentation shown in the annex. Figure 7 represents the real values and the interpolated ones.

After the pressure -or force- values are obtained for the selected points, the rest of the values will be calculated using the hyper-reduced model as seen in the equation 6.9.

$$P_c = U \times (U_z^{-1} \times P_{zc}) \tag{6.9}$$

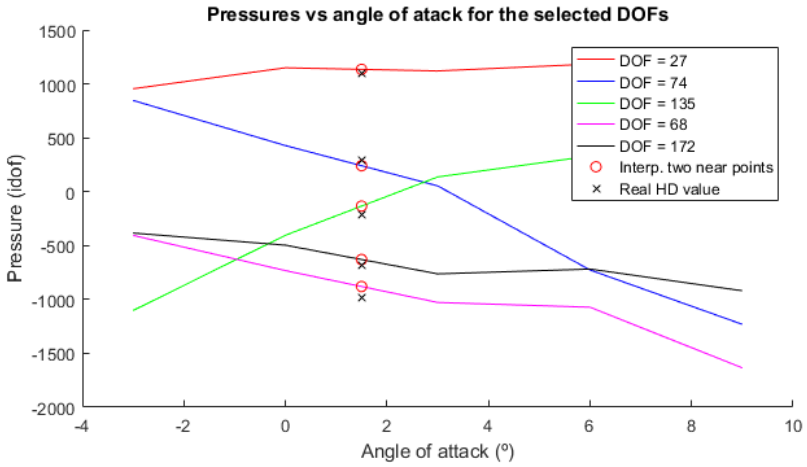


Figure 7: Plot of the selected DOF and their computed approximations for the 1.5º angle

7 Results

Since the aim of this project is to obtain correct reduced models of the wing structure, no comment will be made on the results of the FE analysis because they are considered inputs for the ROM.

There are different aspects of the ROM that can be studied, in this paper the results will be divided in: ROM quality, the accuracy of the results and computation time.

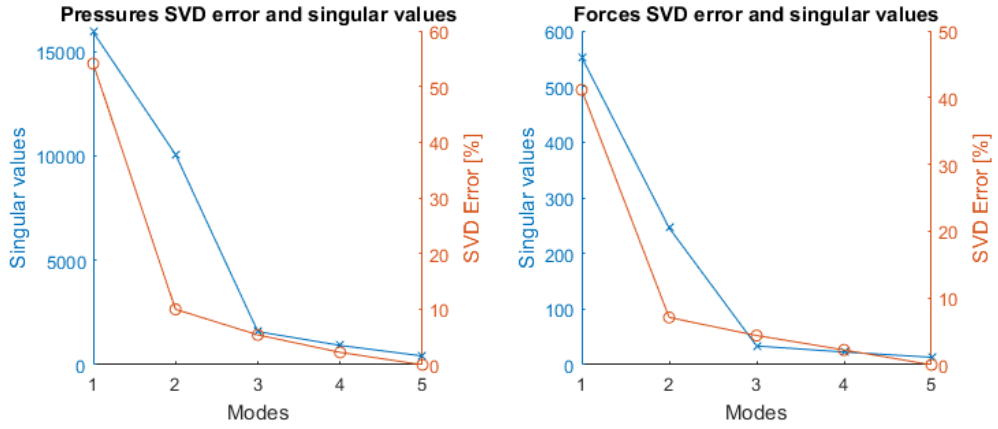
Unless otherwise indicated, the following ROM models have been constructed using the HF snapshots from the angles of attack -3º, 0º, 3º, 6º & 9º; therefore, the models will contain five different modes.

7.1 ROM quality

The analysis of the ROM quality will determine how the model can represent back the snapshots used to create ROM. To do so, the modes generated by the SVD and their weights will be studied.

Pressures and Forces

In Figure 8 the singular values and the SVD error from the pressures and forces ROMs are represented. As explained, the final results will be a combination of the 5 (or less) modes obtained. Those modes can be seen on Figure 9.



(a) Pressures error, singular values **(b)** Forces error and singular values

Figure 8: Chart of the forces and pressures SVD error and singular values per mode

The *SVD error* represents the percentage of error that will be induced if the entire matrix of pressures (M_P) or forces (M_F) was to be generated using only the first mode, the two first modes, etc.

The *Singular values* represent the weight of each mode to regenerate the M_P matrix. If the first modes have significantly more weight than others, it means that the last modes are less important or that are lineal combinations of the previous ones. In either case, the *SVD error* is likely to decrease with the singular values because if a mode is less relevant, the error generated if it is missing will be less significant.

Since the forces from the structural simulation are calculated from the pressures of the fluid, the ROMs are significantly similar. For this reason, they will be analyzed together.

From the *singular values*, it is possible to see that the most important modes are the two first ones; all the same, the error is notable even using only three modes -around 5% of error- or when using 4 modes -around 3%

of error. For this reason, it will be reasonable to use the five modes to get the lowest error possible.

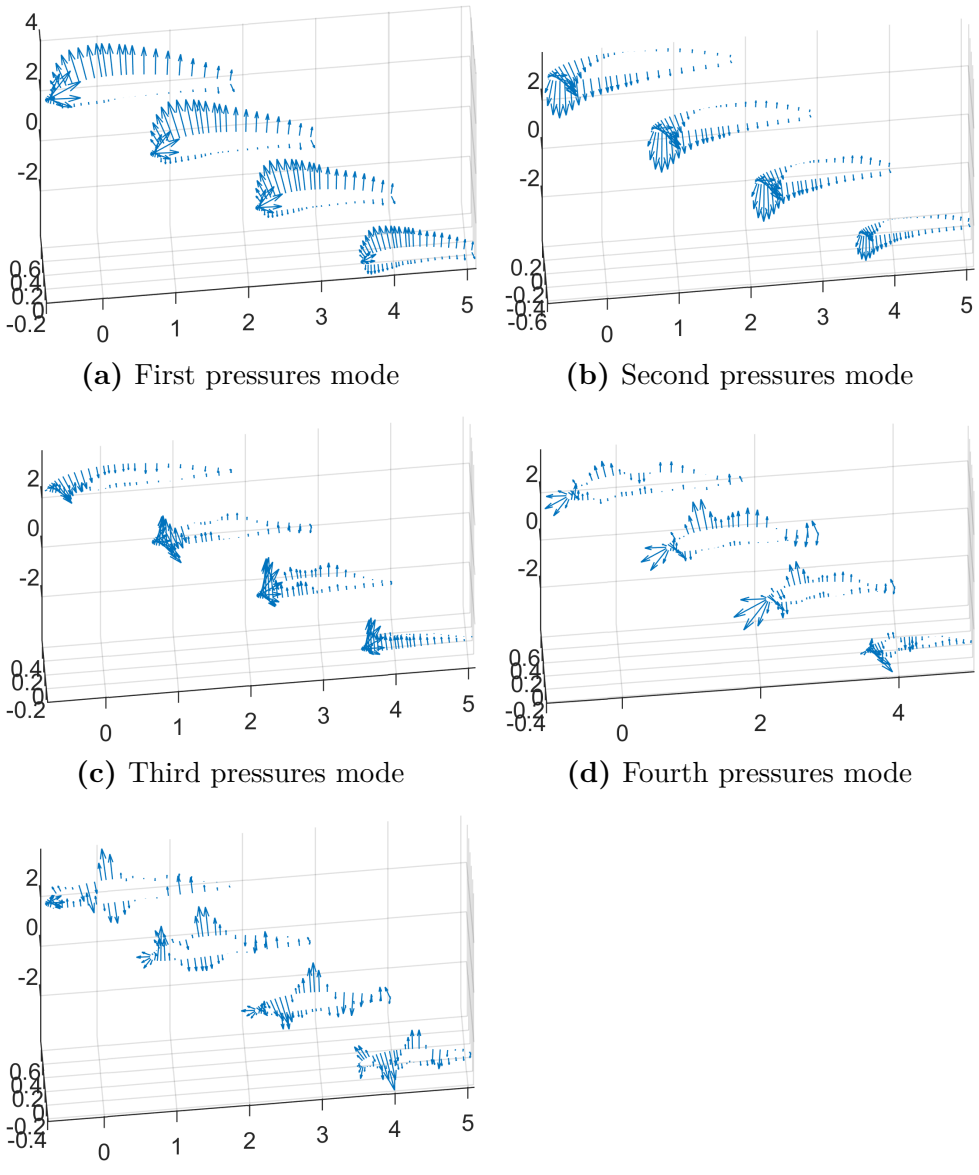
It is important to realize that the error displayed on the chart is only the one due to the ROM; when trying to interpolate, the final error will be higher if the interpolations are not correct.

Displacements

Contrary to the pressures and forces, the error on the displacement is really small. Even when using only the first mode, the error is lower than 0.5%. This is also reflected on the *singular values* and it is possible to see that the first mode weight is extremely bigger than the rest.

With the representation of the modes of Figure 11 it is possible to understand why the first mode is the most important. The displacements of the wing on every angle of attack have probably similar shapes but different intensities. Since these modes are statistical and not physically based modes, the system automatically groups deflection and torsion into the first mode and the rest of the modes are local variations. To see the difference, the vibration modes have been plotted at Figure 12. The frequencies of the vibration modes are 5.87Hz, 65.8Hz, 72.0Hz, 85.4Hz, and 96.4Hz.

In the local modes, it is possible to appreciate how the pressures modify the surface and how the exterior shape is best kept where the beam or ribs are. It is also significant that the second mode is very influenced by these deformations of the skin (note that the large deformations of this mode are from the surface under the wing, not from the top ones). This is possibly the biggest difference between the different angles of attack since on the -3º angle, for example, the wind will not directly impact the lower surface of the wing, and thus, the deformation of the lower surface will be less



(a) First pressures mode

(b) Second pressures mode

(c) Third pressures mode

(d) Fourth pressures mode

(e) Fifth pressures mode

Figure 9: Vectorial representation of the pressure modes

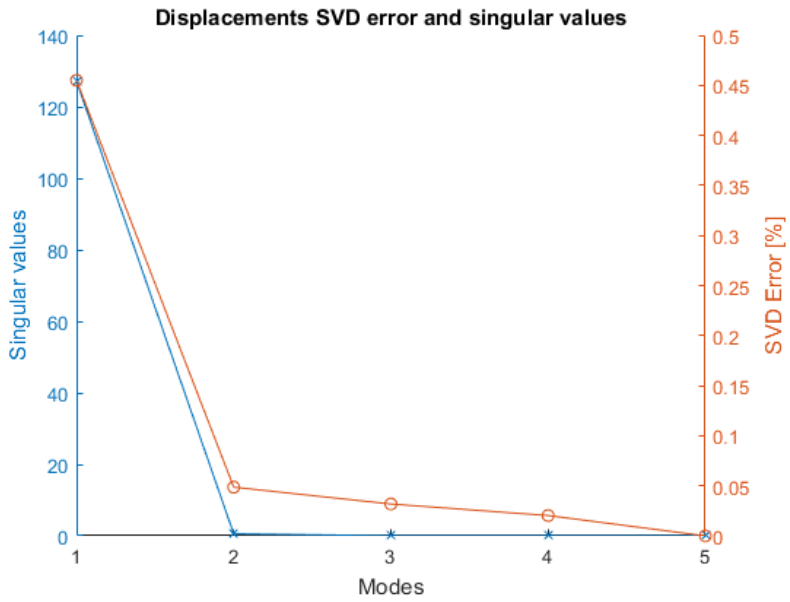


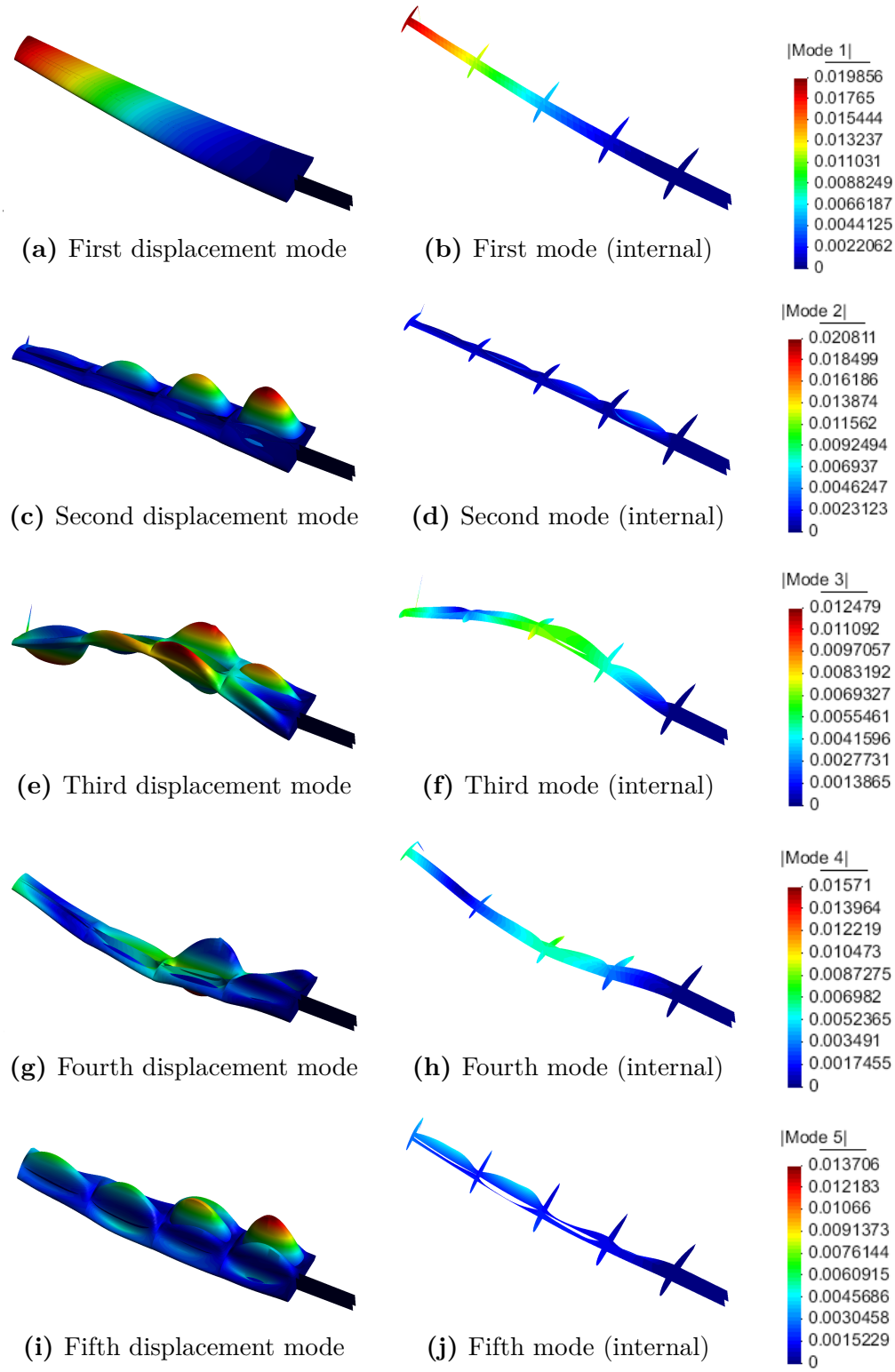
Figure 10: Chart of the displacement SVD error and singular values per mode

important. On the 9º angle, however, the air will directly collide with the surface under the wing increasing its deformation and making the second mode more relevant.

7.2 Results Accuracy

After the analysis of the ROMs now the actual values obtained will be analyzed. To calculate the error induced with this method, additional HF simulations have been carried out. The error shown in this section is the difference in percentage between the ROM results and these HF simulations. The simulations for the angles -1.5º, 1.5º, 4.5º and 7.5º, however, have not been used in any form while generating the ROM.

Since the earlier results show that two modes are enough to successfully represent the displacements, two different ROMs have been employed: one



(a) First displacement mode **(b)** First mode (internal)

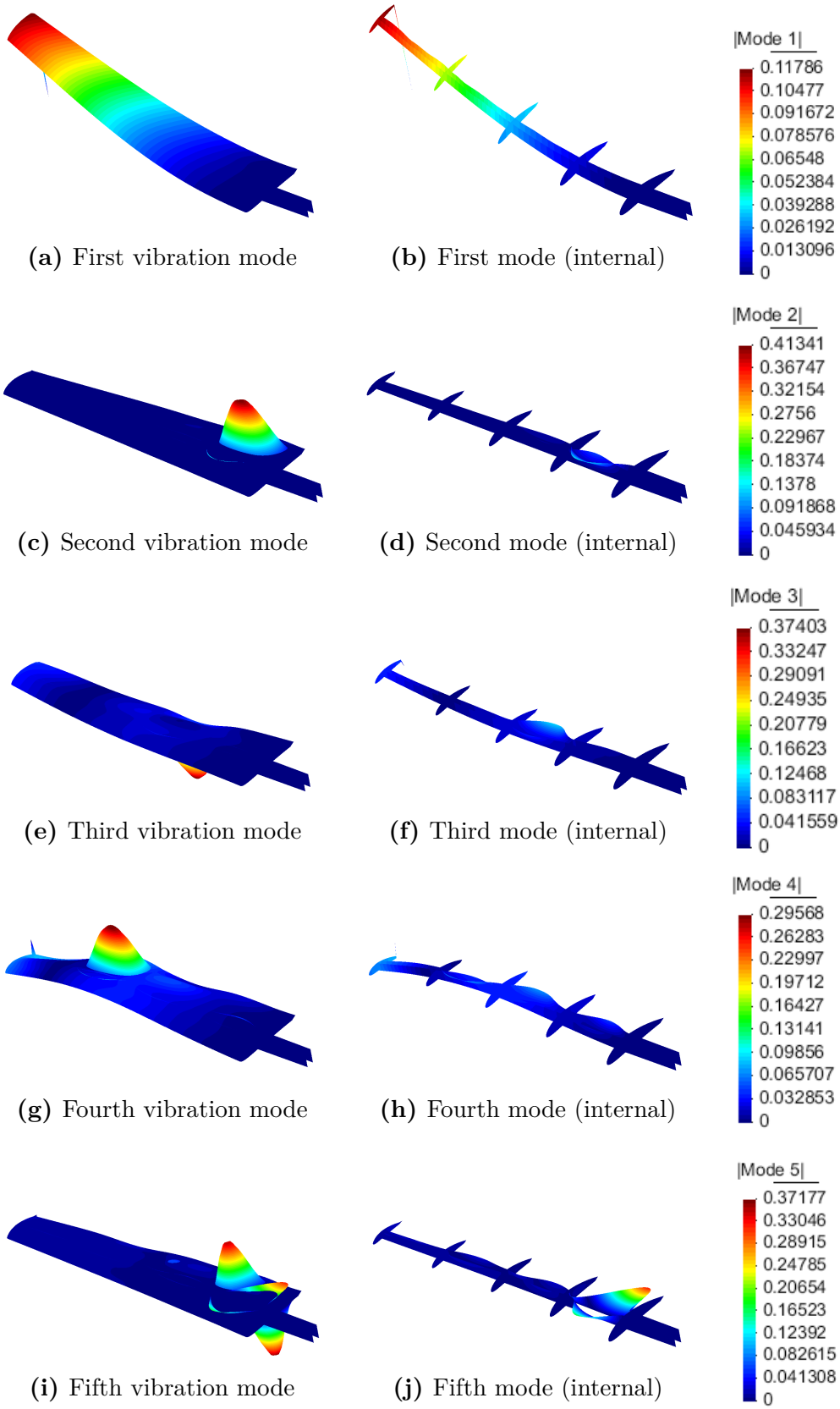
(c) Second displacement mode **(d)** Second mode (internal)

(e) Third displacement mode **(f)** Third mode (internal)

(g) Fourth displacement mode **(h)** Fourth mode (internal)

(i) Fifth displacement mode **(j)** Fifth mode (internal)

Figure 11: Representation of the displacement modes



(a) First vibration mode **(b)** First mode (internal)

(c) Second vibration mode **(d)** Second mode (internal)

(e) Third vibration mode **(f)** Third mode (internal)

(g) Fourth vibration mode **(h)** Fourth mode (internal)

(i) Fifth vibration mode **(j)** Fifth mode (internal)

Figure 12: Representation of the vibration modes

	Pressures	Forces
Two modes	14.59%	13.36%
All modes	14.59%	13.36%

Table 1: Pressures and forces accuracy errors using different ROMs

using only the data from the two nearest angles (and resulting in a ROM with two modes) and a different one using all the data (and resulting in a ROM with five modes).

Pressures and Forces

On Table 1 the mean accuracy error comparing the computed values with the ROM and the HF values for the four angles previously mentioned.

The results are exactly equal because the SVD method minimizes the quadratic error and, if the input variables are interpolated, so will be the results. Since it has been established (see Annex) that the interpolation between the two near angles is the best method to obtain the DEIM selected pressures, the results are interpolated too. Of course, this interpolation would not be necessary if a modeling of the fluid is done or if the application of the ROM is one that already provides this values (like self-aware vehicles).

To demonstrate that, the accuracy of the approximated pressures has been tested again using different methods to obtain the input pressure values. As seen in Table 2, the interpolation between the two nearest values is the best method to obtain the DEIM selected pressures. Also, it is possible to observe that the values are more precise when using all the modes as the study of the ROM quality suggested. Finally, when the input values are not interpolated, the results differ between the two modes and all the

	Interpolation	Algorithm	Regression
Two modes	14.59%	15.19%	16.67%
All modes	14.59%	14.83%	15.28%

Table 2: Pressures accuracy errors using different methods to interpolate the input pressures

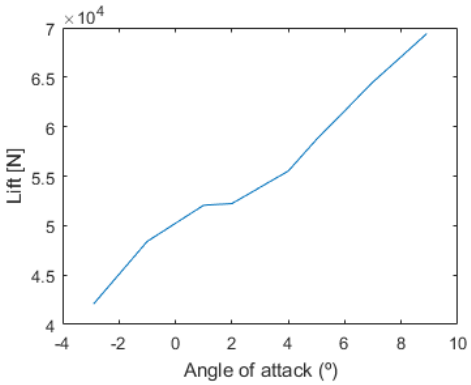


Figure 13: Lift per angle of attack

nodes.

Additionally, since the full distribution of pressures are obtained for any angle of attack between -3 and 9 it is possible to integrate the pressures and obtain the total lift generated by the semiwing. The results of doing this can be seen on Figure 13.

Displacements

The mean accuracy error comparing the ROM computed values with the HF values is shown on Table 3. In the table, there are two sets of values, using the F obtained from the previous ROM and using the high-fidelity F computed with the FE method.

	ROM F	HF F
Two modes	29.15%	0.4053%
All modes	29.15%	0.3252%

Table 3: Displacements accuracy errors using different ROMs and F vectors obtained with the ROM or with the HF technique.

As with the previous results, if the values are interpolated, the results will be a direct interpolation too. As seen, direct interpolation in the displacements would produce results with significant error, adding up to almost 30%. It must be noted that this error is due to the F data having already 15% of error and not because the ROM is unusable.

This is proven by the HF values that produce an extremely precise result even when using only two modes. As predicted in the previous section, the elevated ROM quality pays off and returns very accurate displacements when the input data is correct.

7.3 Computation Time

The computation time of both the HF and ROM models have been measured on the same machine executing only the necessary software. It must be noted that the implementation of the ROM is done in MATLAB while the HF simulations run on a combination of C++ and Python. Several studies show that MATLAB can be up to an order of magnitude slower than C++ [19] [8].

The times for the HF analysis have been measured by Kratos while simulating the cases that have been used for HF in this paper.

	Fluid [s]	Structure [s]
	~~1191.4~~	~~246.36~~
	1702.6	251.60
	1799.1	258.94
	1705.1	257.70
	1376.5	~~389.16~~
	1682.7	258.06
	~~2519.7~~	388.93
	1836.5	260.33
	1475.2	367.22
Mean	**1654.0**	**291.70**

Table 4: Duration in seconds of the HF simulations

Matlab has measured the times for the MOR. When using the two modes approach, the svd had to be recalculated per different angles since the data is different, using the five modes however, the svd was calculated only one time. The individual simulation time of Matlab has been calculated by simulating 11 angles (from -2 to 8) and then dividing. The test has been repeated 5 times.

In tables 5 and 4 the measured times are listed. The lowest and highest value of each type have been discarded and the mean of the remaining values is calculated.

The results from the ROM where significantly faster, from 45578 to 161160 times faster when calculating the displacements and from 264240 to 870526 times faster when calculating the fluid.

[s]	Fluid (two)	Fluid (all)	Structure (two)	Structure (all)
	0.00612	0.00197	~~0.00681~~	~~0.00204~~
	~~0.00642~~	0.00189	0.00641	0.00184
	~~0.00603~~	~~0.00186~~	0.00619	0.00180
	0.00624	~~0.00202~~	~~0.00612~~	0.00179
	0.00638	0.00186	0.00660	~~0.00161~~
Mean	**0.00625**	**0.00190**	**0.00640**	**0.00181**

Table 5: Duration in seconds of the ROM simulations using two modes or all modes

8 Conclusions

The results discussed in the foregoing sections show that the structural predictions (for displacements) obtained with the proposed reduced-order model are startlingly accurate, with error levels around 0.5%(even when just using two modes). This demonstrates the suitability of a typical wing structure to be represented using this method.

By contrast, the purely data-driven approach employed for the fluid furnishes pressure distributions that are not sufficiently accurate for practical purposes.

8.1 Future

These results look very promising not only for aircraft development but for the future of aviation in general. With this degree of accuracy and the speed of this online simulations -under 0.002 seconds- it is completely viable to run real-time simulations with an onboard computer. Moreover, in real planes, it is not necessary to use ROM models from the fluid since the information is obtained from sensors.

A possible application of this technique allow onboard aircraft sensors to identify damage, fatigue and loss of aircraft capability, with the aim of reducing maintenance costs and increasing reliability. Another application is a Model Predicted Control that automatically controls the aircrafts with this accurate information. This integrated health management system

would combine the ROM (and thus the offline simulations information) and the onboard sensors data with prognosis techniques to reliably measure this information.

Future research work, should be devoted to develop a reduced-order model for the fluid itself and to study how to appropriately couple this fluid ROM with the structural ROM proposed in the present work.

Doing so the workflow for the airplane analysis would be completely optimized for speed. Regarding the onboard plane system, future research might focus new applications for the ROM and to further study the implication of this method in an autonomous control system, especially on delicate operations like landing.

8.2 Environmental implications

This project has no direct implication on that regard. All the same, clusters used for executing large FE simulations have high power consumption [23]. By reducing the computation time several orders of magnitude, the power need is also decreased.

Regarding the onboard plane system, autonomous planes can be programed to use the available information to operate more efficiently. A better understanding of the plane state will further this power reduction capabilities. This is extreamly important on planes because they use limited fuel and produce elevated amounts of contaminants directly into the atmosphere [26].

Bibliography

[1] T. Aanonsen. *Empirical interpolation with application to reduced basis approximations*. 2009.

[2] Dan Abramov. *Redux webpage*. URL: `http://redux.js.org/`.

[3] D. Allaire et al. "Multifidelity DDDAS Methods with Application to aSelf-Aware Aerospace Vehicle". In: *Procedia Computer Science* 29.14 (2014), pp. 1182–1192.

[4] S. An, T. Kim, and D. James. "Optimizing cubature for efficient integration of subspace deformations". In: *ACM transactions on graphics* 5.27 (2009), p. 165.

[5] H. Antil et al. "Two-step greedy algorithm for reduced order quadratures". In: *Journal of Scientific Computing* 3.57 (2013), pp. 604–637.

[6] J. Baiges, R. Codina, and S. Idelsohn. "Explicit reduced-order models for the stabilized finite element approximation of the incompressible navier–stokes equations". In: *International Journal for Numerical Methods in Fluids* 12.72 (2013), pp. 1219–1243.

[7] M. Barrault et al. "An empirical interpolation'method: application to efficient reduced-basis discretization of partial differential equations". In: *Comptes Rendus Mathematique* 9.339 (2004), pp. 667–672.

[8] E. Cabot and P. Rodeja. "Problema dels tres cossos restringit". unpublished. June 2016.

[9] K. Carlberg, C. Bou-Mosleh, and C. Farhat. "Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations". In: *International Journal for Numerical Methods in Engineering* 2.86 (2011), pp. 155–181.

[10] K. Carlberg et al. "The gnat nonlinear model reduction method and its application to fluid dynamics problems". In: vol. 2730. 6th AIAA Theoretical Fluid Mechanics Conference, Honolulu, Hawaii, June. 2011, pp. 2011–3112.

[11] S. Chaturantabut and D. C. Sorensen. "Application of pod and deim on dimension reduction of non-linear miscible viscous fingering in porous media". In: *Mathematical and Computer Modelling of Dynamical Systems* 4.17 (2011), pp. 337–353.

[12] R. Everson and L. Sirovich. "Karhunen–Loeve procedure for gappy data". In: *Journal of the Optical Society of America* 8.12 (1995), pp. 1657–1664.

[13] *Express framework website*. URL: `http://expressjs.com/`.

[14] C. Farhat, T. Chapman, and P. Avery. "Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models". In: *International Journal for Numerical Methods in Engineering* (2015).

[15] C. Farhat et al. "Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency". In: *International Journal for Numerical Methods in Engineering* 9.98 (2014), pp. 625–662.

[16] *Flux webpage*. URL: `https://facebook.github.io/flux/`.

[17] M. Grepl et al. "Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations". In: *Mathematical Modelling and Numerical* 3.41 (2007), pp. 575–605.

[18] J. A. Hernández et al. "High-performance model reduction techniques in computational multiscale homogenization". In: *Computer Methods in Applied Mechanics and Engineering* 276 (2014), pp. 149–189.

[19] *High-Performance JIT Compiler*. URL: `http://julialang.org/`.

[20] SIAM J. "Nonlinear Model Reduction via Discrete Empirical Interpolation". In: *Procedia Computer Science* 5.32 (2010), pp. 2737–2764.

[21] N. Nguyen, A. Patera, and J. Peraire. "A best points interpolation method for efficient approximation of parametrized functions". In: *International Journal for Numerical Methods in Engineering* 1.73 (2008), pp. 521–543.

[22] *NodeJs website*. URL: `https://nodejs.org/en/`.

[23] E. Pinheiro et al. *Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems*. Tech. rep. Rutgers University, 2001.

[24] *React webpage*. URL: `https://facebook.github.io/react/`.

[25] D. Ryckelynck. "Hyper-reduction of mechanical models involving internal variables". In: *International Journal for Numerical Methods in Engineering* 1.77 (2009), pp. 75–89.

[26] O. A. Søvde et al. "Aircraft pollution – a futuristic view". In: *Atmospheric Chemistry and Physics* 7.13 (2007), pp. 3621–3632.