Master of Science in Aerospace Engineering

S2. Project Report

Multifidelity aeroelastic optimization with application to a BWB

Author :
M. Gilberto RUIZ JIMÉNEZ

Tutors :
M. Joseph MORLIER
M. Joan MAS COLOMER

Starting date of the project : February, 1st 2019. Duration : 14 months
Due date of the report: July 1st, 2019
Actual submission date : June 27, 2019

Contents

Declaration of Authenticity

This assignment is entirely my own work. Quotations from literature are properly indicated with appropriated references in the text. All literature used in this piece of work is indicated in the bibliography placed at the end. I confirm that no sources have been used other than those stated.

I understand that plagiarism (copy without mentioning the reference) is a serious examinations offence that may result in disciplinary action being taken.

June 27, 2019 Signature: Gilberto Ruiz.

Gilberto RUIZ JIMÉNEZ, B.Eng.

M.Sc. in Aerospace Engineering Student

Abstract

A multifidelity approach for the solution of aeroelastic optimization problems is developed using open-source tools. The Multidisciplinary Optimization problem is managed via OpenMDAO and a dedicated python library. High fidelity aerodynamics are handled with ADFlow, whilst PANAIR solves the low fidelity calls. Structural displacements are computed via NASTRAN and transferred to the aerodynamic mesh with radial basis functions (RBF). The program starts solving the coupled problem at a low fidelity level, and once some predefined convergence tolerance is reached, the high fidelity solver is called. The MDA solution is then treated by the outer-loop optimizer, which modifies the geometry in order to comply with design constraints and minimize drag. Low fidelity implementation results of the NASA CRM wing section are presented as a reference for future solutions in multifidelity mode. The final goal of this project is to apply the methodology to the optimization of a Blended Wing Body (BWB) concept aircraft.

Keywords— Multifidelity, aeroelasticity, BWB, MDO

Nomenclature

BWB	Blended Wing Body
COBYLA	Constrained Optimization By Linear Approximation
IDF	Individual Discipline Feasible
MDA	MultiDisciplinary Analysis
MDF	MultiDisciplinary Feasible
MDO	MultiDisciplinary Optimization
MLE	Maximum Likelihood Estimate
RANS	Reynolds-Averaged Navier–Stokes equations
RBF	Radial Basis Functions
TRMM	Trust Region Model Management

1 Introduction

Modern engineering design problems rely increasingly on the interdisciplinary interactions between subsystems. As a consequence, multidisciplinary design strategies have been developed, these allow to manipulate design variables from various disciplines simultaneously. Moreover, the incorporation of optimization tools within these methodologies leads to what is referred collectively as Multidisciplinary Design Optimization (MDO). The present work deals with the interaction of aerodynamics and structures (so called Fluid-Structure Interaction, FSI), which is a key feature in aircraft design. Coupled aeroelastic analyses allow a better prediction of the aerodynamic forces and structural displacements that the aircraft experiences in real flight, hence, an MDO platform is implemented in order to study these interactions.

Each discipline inside the MDO problem requires a solver that computes its effects on the global problem, with the aerodynamic solution being the main source of computational cost. High fidelity CFD analyses offer accurate results, but at the expense of high computational time. On the other hand, potential flow theory, and its computational implementation, the panel codes, offer reasonable approximations with low computational demands. The main goal of this project is to efficiently use low-fidelity analyses, combined to high-fidelity ones, to conduct aeroelastic optimizations, with an application to a Blended Wing Body (BWB) configuration.

The second section of this report starts by establishing the context of the problem, the issues that arise from the solution strategy and that must be solved so as to get a working method. Then, a state of the art review is presented, with a special focus on two multifidelity methods that are particularly relevant nowadays: Co-kriging and Trust Region Model Management. From there, the degree of novelty for the current method is discussed, as well as the aims and objectives of the project at its current state. The third section deals with the investigation methods, the main tools and their interactions are described. A special focus is put on the multidisciplinary platform (OpenMDAO) and both structural and fluid solvers. Next, section four shows the preliminary results of the method and discusses its implementation. Plots of the stress distribution, C_p distribution and evolution of the iterative solutions are among the main outcomes thus far. Finally, section five outlines the conclusions from the present work and the future work for the next period in order to fulfill the objectives of the project.

2 Semester 2 section

The development of a multifidelity method for aeroelastic analysis requires a coordinated implementation for the high-fidelity and low-fidelity models. In order to achieve this, a management strategy must be implemented. Commonly used alternatives include: Trust Region Model Management (TRMM), Bayesian Regression, Co-kriging [14]. A detailed review of these strategies is given in subsec. 2.2. Another important part of this step is to define the criteria that triggers the alternation between fidelities.

Once a management strategy has been chosen, the next step is to implement said model in the OpenMDAO platform. The resulting code has to be applicable to general cases and not just the particular one discussed in this work. Finally, the multifidelity method will be used for the aeroelastic optimization of a Blended Wing Body. The main issue at this point is the validation of the results because there are few well documented cases of BWB concepts [16, 1] that can be used as reference for the present work.

2.1 Context and key issues

Coupled FSI simulations are the most accurate way to represent aeroelastic phenomena. However, their computational cost remains too expensive for engineering design and optimization. Low-fidelity models are used instead, such as the linearized potential flow approximation. These models are considerably cheaper than RANS solvers and perform well in the low subsonic regime [12]. Nevertheless, there are important non-linear phenomena that are not adequately represented by linearized potential flow models. For this reason, there is a great interest in developing strategies that ease the computing load of the solution while preserving the accuracy of the final solutions.

The present method leverages information from low-fidelity aeroelastic models to optimize a BWB (or any other aerodynamic structure) at a reasonable cost. The multifidelity method has three main parts:

- A multidisciplinary platform that couples the aerodynamic effects to the structural displacements.
- A set of solvers for aerodynamic and structural problems.
- A management strategy that assigns the high or low fidelity software according to given criteria.

At the beginning of this period, the main task was to define the scope of the problem, the objectives and the schedule for the core activities. Next, a state of the art review was needed so that the current strategies and methodologies carried out by the researchers of the field around the world could be taken into account for the present work. Then, as this project relies heavily on the use of open-source tools and python libraries developed previously, a key issue was to get the whole set of programs working together properly. The task took significantly longer than expected due to compatibility problems between recently released versions and cross-platform consistency. Once this was solved, the project started to advance faster. Specifically, the ADFlow implementation was not possible during this period because of integration issues when working on the university server. It is expected that over the course of the next period, there will be enough time to overcome the delays of the present one.

2.2 State of the art

The starting point for the bibliography research is a Survey of Multifidelity Methods by Peherstorfer [14], where a variety of alternatives are listed and classified. Surrogate modelling is a global optimization strategy. It uses co-kriging (see subsec. 2.2.1 for details) as a regression technique in order to link High-Fidelity sources with one or many Low-Fidelity functions. This correlated model can be used to find optimal solutions faster [5]. On the other hand, simpler approaches such as a linear regression are found to offer a good balance between accuracy, cost and simplicity [19]. Other advantages include: ease to combine many low-fidelity sources and robustness for High-Fidelity samples with the presence of noise [19].

Newton methods and their variations have been explored as well. Jovanov and Breuker propose to solve the High-Fidelity problem by adding an aerodynamic correction load to the solution of the Low-Fidelity equation. The defect-correction method accelerates the convergence compared to the Quasi-Newton method [9]. This last case does not consider the methods as black-boxes, but rather exploits the fact that the algorithms of both fidelity levels are known and can be modified at any stage to accommodate the necessary corrections between them. Scholz presents an Aggressive Space Mapping methodology, it solves for the low fidelity fluid–structure interaction solution and then feeds that information to a Quasi-Newton algorithm. The final results are obtained from a mapping function between both fidelity levels [17].

When it comes to the application of mutifidelity models to aerospace design, there are several publications on the subject. A Bayesian-enhanced Low-Fidelity correction proved the ability to maintain high-fidelity accuracy while reducing computational cost associated with the optimization of an airfoil shape [4]. An aeroelastic optimization of a BWB was carried out by Bryson [1] using a new TRMM (Trust Region Model Management) approach (see subsec. 2.2.2). Its main difference is that it adds hybrid additive-multiplicative corrections (or bridge functions) to the low-fidelity analysis [2]. Another approach to the aeroelastic optimization of a BWB is presented by Marques [12]. The flutter boundary problem is solved with a contour location approach (i.e. the zero contour of the aeroelastic damping coefficient). It also incorporates an active learning strategy, where the model evaluations are selected iteratively based on how much they are estimated to improve the predictions.

2.2.1 Kriging and co-kriging

Kriging is a surrogate-based method that approximates a function f using a single set of sample data computed along the domain. The kriging prediction of f is built from a mean base term, $\hat{\mu}^2$ (the circumflex denotes a maximum likelihood estimate, MLE) plus a stationary Gaussian process, $Z(x)$, representing the local features of f around the n sample points, $X = \left\{x^{(1)}, ..., x^{(n)}\right\}^T$, $x \in \mathbf{R}^k$. $Z(x)$ has zero mean and covariance [5].

$$cov[Z(x^{(n+1)}), Z(x^{(i)})] = \sigma^2 \psi^{(i)} \tag{1}$$

Where $\psi^{(i)}$ are correlations between a random variable $Y(x)$ at the point to be predicted ($x^{(n+1)}$) and at the sample data points.

$$\psi^{(i)} = corr[Y(x^{(n+1)}), Y(x^{(i)})] = \exp\left(-\sum_{j=1}^{k} \hat{\theta}_j ||x_j^{(n+1)} - x_j^{(i)}||^{\hat{p}_j}\right) \tag{2}$$

The hyper-parameter, $\hat{p}_j$, can be thought of as determining the smoothness of the function approximation. In many situations, we can assume that there will not be any discontinuities and use $\hat{p}_j = 2$ rather than using an MLE [5]. $\hat{\theta}$ therefore can be thought of as determining how quickly the function changes as $x^{(n+1)}$ moves away from $x^{(i)}$. The kriging prediction is found as the value at $x^{(n+1)}$ that maximizes the likelihood given the sample data and MLEs of the hyper-parameters, and is given by:

$$\hat{y}\left(x^{(n+1)}\right) = \hat{\mu} + \sum_{i=1}^{n} b^{(i)}\psi^{(i)}\left(x^{(n+1)}, x^{(i)}\right) \tag{3}$$

The constants $b^{(i)}$ are given by the column vector $b = \psi^{-1}(y - 1\hat{\mu})$, where ψ is a $n \times n$ symmetric matrix of correlations between the sample data; y is a column vector of responses, $\left\{y(x^{(1)}), ..., y(x^{(n)})\right\}^T$;1 is a $n \times 1$ column vector of ones; and the MLE $\hat{\mu} = 1^T\psi^{-1}y/1^T\psi^{-1}1$ [5].

Co-kriging is the multifidelity implementation of the previous method, it follows the same principle, but uses two or more sets of data to approximate the function f. For the purposes of this document, the sample data will be limited to two data sets evaluated at X_e and X_c, representing the expensive function data and the cheap function data, respectively. Each of the evaluation points is associated to an evaluation of the expensive or cheap function. Both sets are concatenated, resulting in:

$$X = \begin{pmatrix} X_c \\ X_e \end{pmatrix} \quad Y = \begin{pmatrix} Y_c\left(X_c\right) \\ Y_e\left(X_e\right) \end{pmatrix}$$

It is then assumed that $cov\{Y_e(x^{(i)}), Y_c(x)|Y_c(x^{(i)})\} = 0 \quad \forall x \neq x^{(i)}$, in other words, the expensive simulation is correct and any inaccuracies lie wholly in the cheaper simulation [10]. Essentially, the expensive code will be approximated as the cheap code multiplied by a scaling factor, ρ, plus a Gaussian process $Z_d(x)$ that represents the difference between $Z_e(x)$ and $Z_c(x)$:

$$Z_e(x) = \rho Z_c(x) + Z_d(x) \tag{4}$$

In order to obtain the hyperparameters of the Gaussian properties, the process described in the kriging part is repeated, with the main difference being that the covariance is now a more complex matrix of covariances between the different fidelity levels. A complete derivation of the co-kriging method is given in [5], with the final approximation given as:

$$\hat{y}_e(x^{(n_e+1)}) = \hat{\mu} + c^T C^{-1}(y - 1\hat{\mu}) \tag{5}$$

where

$$c = \begin{pmatrix} \hat{\rho}\hat{\sigma}_c^2\psi_c\left(X_c, x^{(n+1)}\right) \\ \hat{\rho}\hat{\sigma}_c^2\psi_c(X_e, x^{(n+1)}) + \hat{\sigma}_d^2\psi_d\left(X_e, x^{(n+1)}\right) \end{pmatrix}$$

2.2.2 Trust Region Model Management

This framework consists of an iterative process on which a surrogate of the desired model is optimized during each iteration, rather than the full model.The concept behind the model management scheme is the adaptive nature in which the size of the trust region is adjusted based upon the accuracy of the surrogate model [4]. For every outer loop iteration, the trust region is re-centered about the current design, and the sizes of the move limits or subproblem bounds are determined via a heuristic. The decision to expand or contract the trust region is based on the ratio of the actual improvement to the expected improvement [2]:

$$\rho^{(k)} = \frac{f(x_c^{(k)}) - f(x_*^{(k)})}{\hat{f}(x_c^{(k)}) - \hat{f}(x_*^{(k)})} \tag{6}$$

where k is the outer iteration count, f and $\hat{f}$ represent high-fidelity and surrogate model evaluations, respectively, and x_* and x_c are the location of the sub-optimum and trust region center, respectively. When the value of ρ is close to unity, it means that the low-fidelity approximation is very good and thus the step should be increased as well as the trust region. The opposite is true for ρ far from unity. There are different criteria for the change of trust and step sizes required for different values of ρ [4, 2]. The following parameters are shown for reference [2]:

$$\Delta^{(k+1)} = \begin{cases} 0.5\Delta^{(k)} & \text{if } \rho^{(k)} \leq 0.25 \\ \Delta^{(k)} & \text{if } 0.25 \leq \rho^{(k)} \leq 0.75 \text{ or } 1.25 \leq \rho^{(k)} \\ \gamma\Delta^{(k)} & \text{otherwise,} \end{cases}$$

$$\gamma = \begin{cases} 2 & \text{if } \|x_*^{(k)} - x_c^{(k)}\|_\infty = \Delta^{(k)} \\ 1 & \text{if } \|x_*^{(k)} - x_c^{(k)}\|_\infty < \Delta^{(k)}. \end{cases}$$

The design optimization using trust regions begins with high- and low-fidelity function and gradient evaluations at the initial design point. An initial trust region is formed using a prescribed initial size. The approximate problem is formed, and optimized using an optimization algorithm of choice. Once the sub-optimization has converged, the trust region is re-centered about the approximate optimum, and the high-fidelity objective and gradient are evaluated. Convergence of the outer loop optimization is checked. If the outer optimization has not converged, the ratio of actual versus predicted improvement is determined and the trust region expanded or contracted accordingly. Using the data at the new design point, an updated approximate problem is constructed, and the process is repeated until the termination criteria are reached [2].

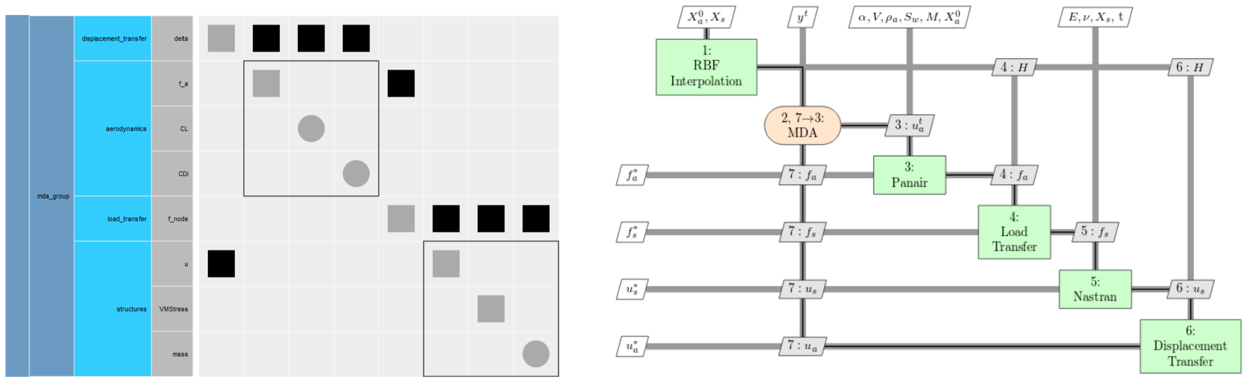
2.3 Justification of the potential degree of novelty

Most of the aforementioned research focuses on models that consider the inner disciplines of the MDO as black-box solvers. The outputs of both fidelities are parsed through statistical processes in order to create surrogate models that are then evaluated at each iteration of the optimizer. In many cases, the low fidelity model is actually a surrogate tailored to the behaviour of the high fidelity solver. On the other hand, the current strategy uses two indepedent ways to model the same phenomena. In addition, this approach aims to leverage the advantages of open-source tools at every level (MDO architecture, structural solver, fluid solvers, post-processing) so that the creation of a surrogate is not needed and the exchange of information between fidelities occurs seamlessly. Furthermore, the present work would introduce a multifidelity strategy to the existing aerostructures package developed by Mas-Colomer [13], a python library developed specifically to set up aerospace problems within the OpenMDAO platform.

2.4 Aims and objectives

From the early stages of the project, a series of objectives were defined for the current period, as follows:

- Complete review of the State of the Art: Literature review required to define strategies, methods, tools, etc.



(a) N^2 (b) XDSM

Figure 1: MDA diagrams of the coupled problem

- Single fidelity MDO implementation: Solution of the sample problem with both fidelity levels (separately), which will later be used as bulding blocks for the multifidelity code.

- Define multifidelity model management: Propose a way to coordinate both fidelities, their interconection and the criteria that triggers a change from one level to the next.

Although the original schedule did not mention it, an initial multifidelity would have been very welcome at the end of this period, however, it was not possible to have it ready before the submission date for this report and thus it is expected to be ready by mid-September. Future work is outlined in sec. 5.

3 Investigation Methods

A series of tools will be used together in order to create a complete multifidelity optimization solver. This section presents each of them as well as their coordinated implementation.

3.1 OpenMDAO

First of all, the Multidisciplinary Design Analysis has to be established. The current work uses the OpenMDAO platform [8], an open source python library that allows the user to create, solve and optimize a wide variety of coupled problems. As a complement to this tool, the aerostructures package [13] assists the user in the creation of aerospace-related problems within the platform. Two different diagram formats of the MDA problem to be solved iteratively using a Gauss-Seidel method are shown in fig. 1.

The MDAO architecture is MultiDisciplinary Feasible or MDF, this means that the coupled problem is solved completely before the output is sent to the outer-loop optimizer. One of the advantages being that the optimization process can be interrupted before converging and the result at any point will satisfy consistency constraints (but not necessarily optimization constrains) [7]. There are many different MDAO architectures, in this case, the MDF was chosen because of the relatively simple problem (only 2 disciplines involved) and the ease of formulation. Another widely used architecture is Individual Discipline Feasible (IDF). It decouples the MDA, adding consistency constraints, and giving the optimizer control of the coupling variables. This strategy generates a problem that is potentially much larger than MDF, depending on the number of coupling variables [7]. A wide study of different MDAO architectures is out of the scope of the current problem.

3.2 Structural Solver

NASTRAN simulations are treated using input files filled with instructions. Two files are needed to run the program, the first one, `'nastran_static_template.inp'` looks mostly like a normal NASTRAN file but instead of the classical Nastran `FORCE` cards, there are modified cards that include a dictionary key for each force component [13], for example:

```
FORCE,201,33566,0,1.0,{Fx33566},{Fy33566},{Fz33566}
```

Another difference with respect to a normal NASTRAN file is that the `GRID` cards, which contain the node coordinates, are also modified to include a dictionary key for each component of the nodal coordinates [13], for example:

```
GRID,33566,,{x33566},{y33566},{z33566}
```

The dictionary keys are used to create input files for NASTRAN in an automated way. As the nodal forces vary during the MDA, the input force values are updated by substituting the dictionary keys in the template by the contents of the dictionary. The same operation is performed for the GRID cards. Even if in the case of an MDA the nodal coordinates remain constant, this feature is useful for shape optimization, in which the nodal coordinates will change with each optimization iteration [13].

The last difference with respect to an ordinary NASTRAN file is that, at the end of the file, there is a list of node IDs. The nodes in this list represent the nodes used for displacement interpolation. Typically, this list is a subset of the total nodes in the structural model, since not all nodes are required to perform the displacement interpolation. Each node ID is precede by a '\$' sign, so they appear as comments to NASTRAN [13]. An example:

```
$List of nodes belonging to the outer skin
$33566
```

The file `'nastran_input_geometry.inp'` is a typical NASTRAN input file containing all the `GRID` points of the jig shape of the structural model (i.e., the shape of the wing structure when no forces are applied onto it) [13]. For example, for one GRID point, this looks like:

```
GRID,33566,,33.5157,0.0010402,2.554365
```

The structural displacements are treated by a special script and interpolated using RBF to the aerodynamic mesh in order to update its shape. This module is treated as if it was a discipline in the OpenMDAO architecture.

3.3 Fluid solvers

High and low fidelity solvers were established to obtain the aerodynamic loads from the wing shape and the fluid characteristics. ADFlow is an open-source CFD code developed by the MDOLab at the University of Michigan. It solves the compressible Euler, laminar Navier-Stokes and Reynolds-Averaged Navier-Stokes equations [11]. On the other hand, PANAIR [3] was chosen as the low fidelity model. It is an open-source implementation of the potential flow theory via panel methods. The exchange between fidelities will take place at this stage: the idea is that for every call of the MDA, PANAIR will initially compute the solutions that are fed to the Gauss-Seidel algorithm. Next, when the norm of the unknowns reaches a given tolerance, ADFlow will start from the previoulsy computed solution. As a consequence, the MDA computation will be more accurate than the one calculated in low fidelity and faster than the one calculated in high fidelity.

In a similar way to the structural solver, the aerodynamic loads obtained from either one of the solvers are interpolated using the load transfer script of the aerostructures package. The result is sent to the structural discipline to re-start the cycle until convergence is reached.

3.4 Optimizer & post-processing

The solution of the MDA is then sent to the outer-loop optimizer, a COBYLA algorithm readily available in the OpenMDAO framework. The main characteristic of COBYLA is that it is a numerical optimization method for constrained problems where the derivative of the objective function is not known [15]. Finally, post-processing of the data is achieved through python scripts that transform outputs from NASTRAN and PANAIR to mesh files (.msh) that can be opened in the open source software Gmsh [6].

4 Results and analysis

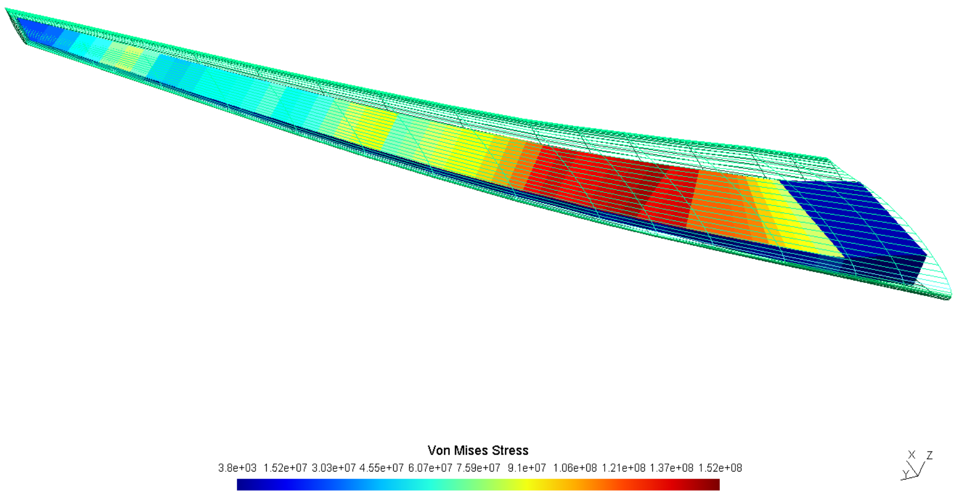
A preliminary review of the MDAO results is presented in subsection 4.1, these results are important because they allow to identify the computational load at both the MDA and the outer-loop parts of the process and propose ways to reduce their cost.

4.1 Single Fidelity

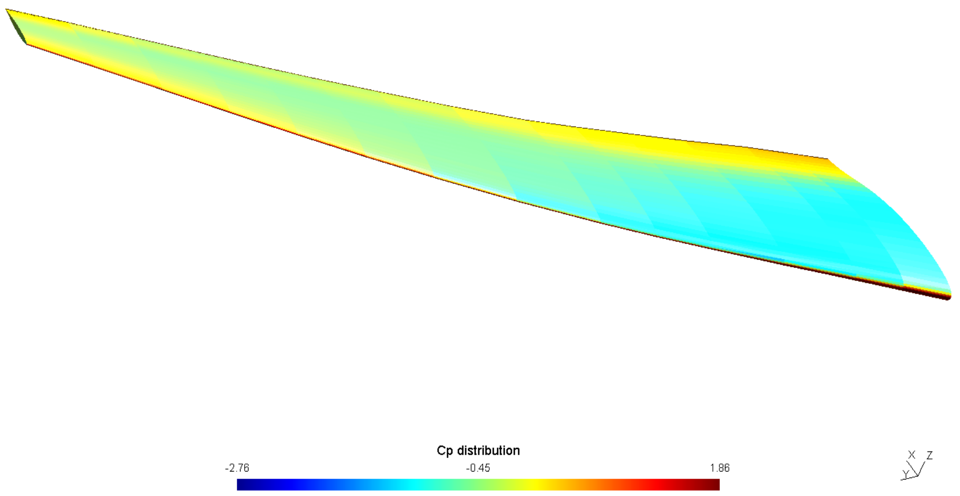
For simplicity reasons, a basic structure was used to test and develop the code before its final implementation on the BWB concept. The wing section of the NASA's Common Research Model [18] was chosen as the starting geometry to be optimized. The objective function of the problem is the induced drag, C_{d_i}, with respect to a series of design variables, and respecting two main constraints: cruise lift and maximum Von Mises stress. These parameters are detailed in appendix A. The sample problem was set up using the low-fidelity solver and NASTRAN. Some slight changes were made on the original scripts to ensure compatibility with newer versions of Python. However, the overall strategy did not change.

The optimized solution showing Von Mises stress is shown in fig. 2a, the C_p distribution was computed as well (see fig. 2b). Both results are presented over the optimized geometry. Aside from the equivalent stress and C_p distribution, it is interesting to look at the objective function convergence and the constraint evaluation plots. These are displayed in fig. 3, please note that they correspond to the outer-loop of the MDAO problem, not the MDA that must be solved iteratively for every step of the optimizer. A single example of the coupled probem solution is shown in fig. 4, it is at this level that an alternation between fidelities is proposed. It takes 286 iterations of the optimizer for the model to converge in low fidelity mode to a C_{d_i} of 0.00718.

Once the multifidelity model is ready, a performance comparison will be carried out between both methods. The main parameters for the comparison will be the total execution time and the distance to the optimum and to the normalized constraints. An equivalent implementation of the problem using a High-Fidelity solver was also reviewed as a way to get familiar with its functioning. The computing time in this case is significantly larger compared to the previous one, some issues with the server where ADFlow is located prevented the implementation of this code.

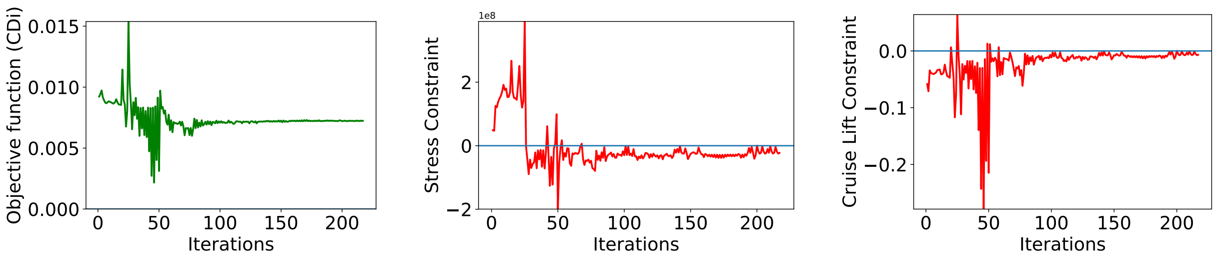

Von Mises Stress
3.8e+03 1.52e+07 3.03e+07 4.55e+07 6.07e+07 7.59e+07 9.1e+07 1.06e+08 1.21e+08 1.37e+08 1.52e+08
X Z
Y

(a) Von Mises stress, aerodynamic mesh shown for reference. (Pa)



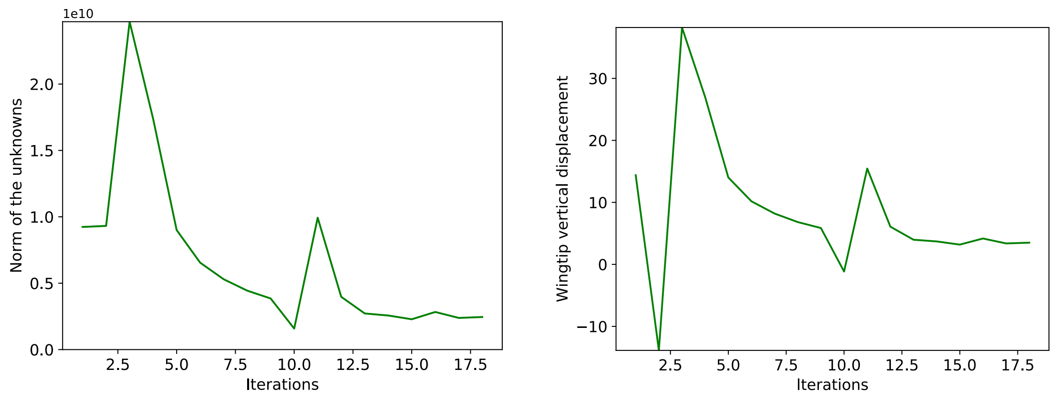
(b) C_p distribution

Figure 2: Optimized wing section for the low-fidelity implementation.



(a) (b) (c)

Figure 3: Objective function, stress constraint and cruise lift constraint plots.



(a) Norm of the unknowns (b) Wingtip vertical displacement

Figure 4: Solution convergence for one sample step of the MDA.

5 Conclusions and perspectives

Up to this point, the main tasks of the present work were the state of the art review and the implementation of sample problems in Single-Fidelity mode in order to get acquainted with the various tools required to create the multifidelity optimization. Results from low fidelity MDAO implementations were presented, with a focus on two levels of convergence: MDA-level and outer-loop optimization level.

Future work for S3 includes writing the python code that connects both fidelity levels within the OpenMDAO environment. This will allow the optimization of the sample case, which in turn can be used as a bench-marking reference to review the performance of the new proposal versus single-fidelity implementations. Depending on the results, minor or major changes will be applied to the multifidelity code in order to improve its efficiency. At the moment, the addition a surrogate model for the outer-loop evaluations of the optimization algorithm is being considered, it would potentially reduce the computational cost with less calls of the MDA. The possible downside of this strategy is that the creation and tuning of the surrogate model may be harder to process than the spared MDA calls. Once the method handles properly the sample case, a BWB configuration will be optimized and the results will be validated using academic sources. Since the complexity of the problem will increase, a second performance review of the code will be carried out to check for any possible improvements.

Acknowledgements

This work is supported by CONACYT (The Mexican National Council for Science and Technology), grant number 2018-000041-01EXTF-00038.

A Sample problem conditions

This is an extract of the code used to run the optimization problem, some parts were left out for clarity. The full code is available on GitHub: `https://github.com/mid2SUPAERO/RP_MAE_GILBERTO_RUIZ_JIMENEZ`.

Flow conditions, reference quantities and material properties:

```
#Problem parameters
#Speed of sound
a = 297.4 #m/s

Sw = 383.689555 #Wing Surface, m^2
V = 252.16168 #Free-stream velocity, m/s
Mach = V/a #Mach number
rho_a = 0.38058496 #Air density kg/m^3
alpha_0 = 1.340 #Initial angle of attack, deg
b_0 = 58.7629 #Wing span, m
b_baseline = 58.7629 #Base line wing span, m
c = 7.00532 #Wing chord, m
E = 6.89e10 #Young's Modulus
nu = 0.31 #Poisson's ratio
rho_s = 2795.67 #Structure density kg/m^3
#Reference aircraft weight (mass units)
W_ref = 226796.185
#Wing weight (full span) of the reference aircraft (mass units)
W_ref_wing = 26400.
#Airframe weight (complete aircraft minus wing structure, mass units)
W_airframe = W_ref - W_ref_wing
#Yield stress (can also be used as ultimate stress if FS = 1.5)
sigma_y = 450.e6
#Factor of safety
FS = 1.
#Cruise load factor
n = 1.
```

Design variable boundaries and constraints.

```
#Design variable boundaries
#Thickness
t_max = 3*t_0
t_min = 0.25*t_0

#Rod section
a_max = 3*a_0
a_min = 0.25*a_0

#Root chord
cr_max = 1.5*cr_0
cr_min = 0.75*cr_0

#Break chord
cb_max = 1.5*cb_0
cb_min = 0.75*cb_0
```

```
#Tip chord
ct_max = 1.5*ct_0
ct_min = 0.75*ct_0

#Sweep angle
sweep_max = 50.
sweep_min = 30.

#Wing span
b_max = 80.
b_min = 40.

#Angle of attack
alpha_max = 5.
alpha_min = -2.

#Add independent variables (parameters)
root.add('wing_area', IndepVarComp('Sw', Sw), promotes=['*'])
root.add('Airspeed', IndepVarComp('V', V), promotes=['*'])
root.add('air_density', IndepVarComp('rho_a', rho_a), promotes=['*'])
root.add('Mach_number', IndepVarComp('Mach', Mach), promotes=['*'])
root.add('baseline_wing_span', IndepVarComp('b_baseline', b_baseline),
          promotes=['*'])
root.add('wing_chord', IndepVarComp('c', c), promotes=['*'])
root.add('Youngs_modulus', IndepVarComp('E', E), promotes=['*'])
root.add('Poissons_ratio', IndepVarComp('nu', nu), promotes=['*'])
root.add('material_density', IndepVarComp('rho_s', rho_s),
          promotes=['*'])
root.add('airframe_mass', IndepVarComp('W_airframe', W_airframe),
          promotes=['*'])
root.add('Tensile_Yield_Strength', IndepVarComp('sigma_y', sigma_y),
          promotes=['*'])
root.add('factor_safety', IndepVarComp('FS', FS), promotes=['*'])
root.add('y_leading_edge_baseline', IndepVarComp('y_le_baseline',
          y_le_baseline), promotes=['*'])
root.add('z_leading_edge', IndepVarComp('z_le', z_le), promotes=['*'])
root.add('airfoil_thickness', IndepVarComp('th', th), promotes=['*'])
root.add('camber_chord_ratio', IndepVarComp('camc', camc),
          promotes=['*'])
root.add('base_aerodynamic_mesh', IndepVarComp('xa_b', xa_b),
          promotes=['*'])
root.add('base_structure_mesh', IndepVarComp('xs_b', xs_b),
          promotes=['*'])
root.add('cruise_load_factor', IndepVarComp('n', n), promotes=['*'])
root.add('root_leading_edge_x', IndepVarComp('xr', xr),
          promotes=['*'])
root.add('wing_twist', IndepVarComp('theta', theta), promotes=['*'])

# Independent variables that are optimization design variables
root.add('thicknesses', IndepVarComp('t', t_0), promotes=['*'])
root.add('rod_sections', IndepVarComp('a', a_0), promotes=['*'])
root.add('root_chord', IndepVarComp('cr', cr_0), promotes=['*'])
root.add('break_chord', IndepVarComp('cb', cb_0), promotes=['*'])
root.add('tip_chord', IndepVarComp('ct', ct_0), promotes=['*'])
root.add('sweep_angle', IndepVarComp('sweep', sweep_0),
          promotes=['*'])
root.add('wing_span', IndepVarComp('b', b_0), promotes=['*'])
root.add('angle_of_attack', IndepVarComp('alpha', alpha_0),
          promotes=['*'])

#Constraint components
#Lift coefficient constraints (two constraints with same value
#to treat equality constraint as two inequality constraints)
root.add('con_lift_cruise_upper', ExecComp(
'con_l_u = CL - n*(W_airframe+2*1.25*mass)*9.81/(0.5*rho_a*V**2*Sw)')
         , promotes=['*'])
root.add('con_lift_cruise_lower', ExecComp(
'con_l_l = CL - n*(W_airframe+2*1.25*mass)*9.81/(0.5*rho_a*V**2*Sw)')
         , promotes=['*'])

#Maximum stress constraint (considering factor of safety)
root.add('con_stress', ExecComp(
         'con_s = FS*2.5*max(VMStress) - sigma_y',
         VMStress=np.zeros(n_stress,dtype=float)), promotes=['*'])

#Stress constraints (considering max load factor and factor of safety)
for i in range(n_stress):
    root.add('con_stress_'+str(i+1), ExecComp('con_s_'+str(i+1)+
    '= FS*2.5*VMStress['+str(i)+'] - sigma_y', VMStress
    =np.zeros(n_stress, dtype=float)), promotes=['*'])
```

References

[1] D. Bryson, M. Rumpfkeil, and R. Durscher. Framework for multifidelity aeroelastic vehicle design optimization. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, number 18 in AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, 2017.

[2] D. Bryson and M. P. Rumpfkeil. Comparison of unifed and sequential-approximate approaches to multifdelity optimization. In *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, number 58 in AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Mar. 2017.

[3] R. Carmichael and L. Erickson. Pan air-a higher order panel method for predicting subsonic or supersonic linear potential flows about arbitrary configurations. In *14th Fluid and Plasma Dynamics Conference*, page 1255, 1981.

[4] C. C. Fischer, R. V. Grandhi, and P. S. Beran. Bayesian-enhanced low-fidelity correction approach to multifidelity aerospace design. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, volume 56, pages 3295–3306. American Institute of Aeronautics and Astronautics, 2017.

[5] A. I. Forrester, A. Sóbester, and A. J. Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2088):3251–3269, dec 2007.

[6] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

[7] J. Gray, K. T. Moore, T. A. Hearn, and B. A. Naylor. Standard platform for benchmarking multidisciplinary design analysis and optimization architectures. *AIAA journal*, 51(10):2380–2394, 2013.

[8] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor. OpenMDAO: An Open-Source Framework for Multidisciplinary Design, Analysis, and Optimization. *Structural and Multidisciplinary Optimization*, 2019. In Press.

[9] K. Jovanov and R. D. Breuker. Accelerated convergence of static aeroelasticity using low-fidelity aerodynamics. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, jan 2015.

[10] M. C. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

[11] Z. Lyu, G. K. Kenway, C. Paige, and J. Martins. Automatic differentiation adjoint of the reynolds-averaged navier-stokes equations with a turbulence model. In *21st AIAA Computational Fluid Dynamics Conference*, page 2581, 2013.

[12] A. N. Marques, R. Lam, A. Chaudhuri, M. M. Opgenoord, and K. E. Willcox. A multifidelity method for locating aeroelastic flutter boundaries. In *AIAA Scitech 2019 Forum*, number 0 in AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2019.

[13] J. Mas-Colomer. *Aeroelastic Similarity of a Flight Demonstrator via Multidisciplinary Optimization*. Theses, Université de Toulouse, Dec. 2018.

[14] B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3):550–591, 2018.

[15] M. J. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*, pages 51–67. Springer, 1994.

[16] J. Quinlan and F. H. Gern. Conceptual design and structural optimization of nasa environmentally responsible aviation (era) hybrid wing body aircraft. In *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, number 0 in AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Mar. 2016.

[17] T. Scholcz, A. van Zuijlen, and H. Bijl. Space-mapping in fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 281:162 – 183, 2014.

[18] J. Vassberg, M. Dehaan, M. Rivers, and R. Wahls. Development of a common research model for applied cfd validation studies. In *26th AIAA Applied Aerodynamics Conference*, page 6919, 2008.

[19] Y. Zhang, N. H. Kim, C. Park, and R. T. Haftka. Multifidelity surrogate based on single linear regression. *AIAA Journal*, 56(12):4944–4952, 2018.