# Efficient Aerodynamic Optimization of Aircraft Wings

Pedro Rodrigues
pedro.miguel.verissimo.rodrigues@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

April 2018

**Abstract**

One of the most important keys to the successful design of complex systems is disciplinary integration. Multidisciplinary Design and Optimization is now a promising methodology for the efficient design of such systems, since it combines multidisciplinary analysis with gradient-based optimization techniques. Therefore, this methodology requires the derivatives evaluation of the functions of interest with respect to the design variables, which is the most demanding computational task in the optimization process. Traditionally, those derivatives are calculated inefficiently and inaccurately using approximate methods. Therefore, the objective of this work is to develop an efficient optimization framework to solve aerodynamic design problems using exact gradient information. Firstly, a survey on sensitivity analysis methods is conducted to identify which tools are available and understand their respective merits. Secondly, an aerodynamic model based on the panel method is reformulated into five smaller modules, in which the respective sensitivity analysis blocks are constructed using exact gradient estimation methods: automatic differentiation, symbolic differentiation and the adjoint method. Both the aerodynamic tool and respective sensitivity analysis are validated using a wing design tool and the finite-differences method, respectively. Finally, aerodynamic optimization problems are solved using the new tool with remarkable success since, when compared to the finite-differences method, the optimization time can be reduced by 90%.

**Keywords:** Gradient-based optimization, Aerodynamic design, Sensitivity analysis, Automatic differentiation, Adjoint method.

## 1. Introduction

Nowadays, aircraft industry is experiencing an economical expansion leading to an increase in the competition for providing new and better aircraft configurations capable of fulfilling new requirements. As those impositions are tighter, the system's complexity increases, requiring efficient techniques to project those systems.

Multidisciplinary Design and Optimization (MDO) is a promising tool to design such systems, combining optimization procedures with coupled multidisciplinary analysis so that the best design is obtained according to some objective function while satisfying the problem's constraints [1]. Since hundred or even thousand design variables are usually required to faithfully parametrize the system, the use of gradient-based optimization techniques are imperative to efficient optimization due to faster convergence rates when comparing with heuristic and gradient free methods. However, the performance of these type of algorithms depends heavily on how efficiently the sensitivities/derivatives of the interest functions with respect to the design variables are calculated. The adjoint method is probably one of the most attractive to efficient sensitivity analysis since the derivatives may be calculated exactly and independently of the number of the system's inputs.

The objective of this work is to develop an efficient aerodynamic optimization tool to be lately incorporated in an aeroelastic framework for wing analysis with static aero-structural capabilities, developed by Almeida [2]. The wing's structure was modeled using a 3D finite-element model, applied to the neutral axis and the aerodynamic loads were calculated using a panel method code developed by Cardeira [3]. Almeida took advantage of the framework's static aero-structural capabilities to minimize the wing mass, subject to lift and stress constraints, and it was concluded that a new improved design was achieved. However, the optimization process was conducted slowly since the sensitivity analysis was performed inefficiently and inaccurately, using finite-differences (FD).

This work finds its motivation in solving the issue of inefficient aerodynamic optimization by providing an efficient sensitivity analysis framework to the panel method code implemented by Cardeira

[3]. The new optimization tool relies on a gradient-based algorithm and it uses the adjoint method to efficient sensitivity analysis.

## 2. Optimization Methods

When categorizing optimization methods, a common division found in literature is into heuristics or deterministic methods. The former is used when the traditional deterministic approach fails. An advantage of using this approach is that very few mathematical properties about the problem must be verified and thus, may be used in a broader set of problems. The latter is usually preferable if a set of assumptions about the problem at hand can be made, e.g. continuous and smooth interest functions where derivatives exist. If those assumptions are possible, deterministic methods provide theoretical guarantee that, at least, a local minimum is found. This type of methods are usually very robust and they are widely used in design problems.

Following a deterministic approach, Belegundu and Chandrupatla [4] define optimization as the process of minimizing a given objective function while satisfying a given set of constraints. A typical engineering optimization problem may be generally expressed as nonlinear programming (NLP) stated as

$$\begin{array}{lll} \text{minimize} & f(\mathbf{x}) & \\ \text{w.r.t} & \mathbf{x} \in \mathbb{R}^n & \\ \text{subject to} & g_i(\mathbf{x}) \leq 0 & \text{for } i = 1, ..., m \\ & h_j(\mathbf{x}) = 0 & \text{for } j = 1, ..., \ell \\ & \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U & \end{array} \tag{1}$$

where $\mathbf{x}$ is the design vector, or the independent variables, $\mathbf{x}^L$ and $\mathbf{x}^U$ are the lower and upper bounds, $f$ is the objective function, $h_j$ and $g_i$ are the equality and inequality constraints, respectively. All of these functions may be nonlinear. Depending whether the gradient of both objective and constraint functions are required or not, this problem may be further categorized in gradient-based or gradient-free methods.

**Gradient-Based Methods** are usually preferable due to faster convergence rates and clear stopping criterion. Furthermore, those are divided into constrained or unconstrained problems whether or not the constraint functions and bounds are present in the optimization problem. Many engineering design problems falls into the first category since project limitations exist and must be accounted. An example is a wing drag minimization problem, subject to lift and stress constraints. Nevertheless, the methodology to solve these problems consists in two common steps:

- Find a descent/feasible direction based on gradient information.
- Minimize the objective function in that direction (line search)

A stopping criterion is also required to decide when the algorithm should stop, which for a constrained NLP corresponds to the Karusch-Kuhn-Tucker (KKT) necessary conditions to optimality

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x_k} = \frac{\partial f}{\partial x_k} + \sum_{i=1}^{m} \mu_i \frac{\partial g_i}{\partial x_k} + \sum_{j=1}^{\ell} \lambda_j \frac{\partial h_j}{\partial x_k} = 0 \\ \mu_i \geq 0, \quad i = 1, ..., m \\ \mu_i g_i = 0, \quad i = 1, ..., m \\ g_i \leq 0, \quad i = 1, ..., m \\ h_j = 0, \quad j = 1, ..., \ell \end{aligned} \tag{2}$$

where, $\mathcal{L} = f + \sum_{i=1}^{m} \mu_i g_i + \sum_{j=1}^{\ell} \lambda_j h_j$ is the Lagrangian function, $\mu_i$ and $\lambda_j$ are Lagrange multipliers. Although the KKT conditions must be necessarily verified, the solution point is a minimum only if the Hessian matrix is positive definite at that point.

Several algorithms are supported by this formulation, including the Reduced Gradient Method, the Method of Feasible Directions and the Sequential Quadratic Programming [5] method. The latter will be used since it presents some advantages including: the initial point may be unfeasible, only gradients of active constraints are needed, higher rate of convergence when comparing with similar methods and it is already implemented in MATLAB®.

## 3. Sensitivity Analysis

A common feature among gradient-based optimization methods is the requirement of providing the gradient of both objective and constraint functions to the optimization algorithm. Since the performance of such methods depends on how those gradients are calculated, it is necessary to survey the methods available to perform accurate and efficient sensitivity analysis.

### 3.1. Symbolic Differentiation

Symbolic differentiation (SD) means to apply the well known rules of differentiation, such as the ones applied to the sum, difference, product or quotient of functions using computational software. The methodology is restricted to explicit functions and therefore, may become impracticable to be implemented in very large problems. Some tools are available to the effect, including the `diff()` function from the *Symbolic Math Toolbox*, written to MATLAB®.

3.2. Finite-Differences Method

The finite-differences method (FD) is one of the oldest and simpler methods to estimate the sensitivities. Consider a vector valued function $\mathbf{F} = [F_1, ..., F_m]^T$ dependent on the vector of independent variables $\mathbf{x} = [x_1, ..., x_n]^T$. Finite-difference formulas approximate the derivative of a function using a quotient of a difference and they may be obtained through Taylor-series expansions. An example is the forward finite-difference (FFD) formula

$$\frac{\partial F_j(\mathbf{x}_0)}{\partial x_i} = \frac{F_j(\mathbf{x}_0 + \mathbf{e}_i h) - F_j(\mathbf{x}_0)}{h} + \mathcal{O}(h) \quad (3)$$

where the truncation error is proportional to the step size $h$. If higher precision is desired, Taylor-series expansions may be combined to obtained the central finite-difference formula

$$\frac{\partial F_j(\mathbf{x}_0)}{\partial \mathbf{x}_i} = \frac{F_j(\mathbf{x}_0 + \mathbf{e}_i h) - F_j(\mathbf{x}_0 - \mathbf{e}_i h)}{2h} + \mathcal{O}(h^2) \quad (4)$$

where the truncation error is now about $\mathcal{O}(h^2)$. These formulas present the advantage of easy implementation and they can be used without detailed knowledge of the system. However, they are approximate formulas which incur in numerical errors of truncation and subtractive cancellation. Moreover, computing the gradient of $F_j$ using FD may be impractical since the cost of the calculation depends linearly on the size of $\mathbf{x}$.

3.3. Complex-Step Derivative

The complex-step derivative (CSD) is a formula to estimate the first derivative of a function using notions of complex-variable calculus. Considering Taylor-series expansion of $\mathbf{F}$ in the imaginary axis direction and taking the imaginary part, it is obtained

$$\frac{\partial F_j(\mathbf{x}_0)}{\partial x_k} = \frac{\mathrm{Im}[F_j(\mathbf{x}_0 + ih\mathbf{e}_k)]}{h} + \mathcal{O}(h^2) \quad (5)$$

where $ih$ is a pure imaginary step. The truncation error is about $\mathcal{O}(h^2)$, but for a sufficiently small step, the derivative may be calculated with as much precision as the machine allows, affected only by round-off errors. This formula presents the disadvantage of operating with complex algebra, which is not supported by all programming languages. Furthermore, estimating the gradient using the complex-step derivative may be too expensive since the calculation depends on the number of independent variables. Nevertheless, this formula allows to faithfully verify derivatives calculated with other methods.

3.4. Semi-Analytical Methods

According to Peter and Dwight [6], semi-analytical methods such as the direct and the adjoint methods are the most efficient to sensitivity analysis. Consider again the vector valued function $\mathbf{F}$, which depends explicitly on $\mathbf{x}$ and implicitly on the state vector $\mathbf{y} = [y_1, ..., y_k]^T$, whose relation between the independent variables and the state vector is given by a system of residual equations

$$\mathbf{R}(\mathbf{x}, \mathbf{y}(\mathbf{x})) = 0 \quad (6)$$

According to the chain-rule, the total derivative of the function with respect to the independent variables is

$$\frac{\mathrm{d}\mathbf{F}}{\mathrm{d}\mathbf{x}} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}} + \frac{\partial \mathbf{F}}{\partial \mathbf{y}} \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}} \quad (7)$$

Since Equation (6) must always be verified, it is true that

$$\frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \mathbf{y}} \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}} = 0 \quad (8)$$

#### 3.4.1 Direct Method

A possible approach to calculate $\frac{\mathrm{d}\mathbf{F}}{\mathrm{d}\mathbf{x}}$ corresponds to solve directly the linear system which arise from Equation (8)

$$\frac{\partial \mathbf{R}}{\partial \mathbf{y}} \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}} = -\frac{\partial \mathbf{R}}{\partial \mathbf{x}} \quad (9)$$

and replace $\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}}$ in Equation (7). This approach is called the direct method and it is best suited if the number of inputs is smaller than the number of outputs ($n < m$), since solving Equation (9) is equivalent to solve $n$ systems of equations.

#### 3.4.2 Adjoint Method

Another approach corresponds to solve algebraically Equation (9) to $\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}}$ and replace the resulting expression in Equation (7). The result is

$$\frac{\mathrm{d}\mathbf{F}}{\mathrm{d}\mathbf{x}} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \underbrace{-\frac{\partial \mathbf{F}}{\partial \mathbf{y}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{y}}\right]^{-1}}_{[\psi]^T} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\right] \quad (10)$$

Since the matrix product is associative, it is possible to assign the transpose of an adjoint matrix to the first two matrices of the second term, in the right hand side, as suggested by the under bracket. Consequently, $\frac{\mathrm{d}\mathbf{F}}{\mathrm{d}\mathbf{x}}$ may be calculated if the adjoint matrix is known, which may be determined by solving the system

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{y}}\right]^T [\psi] = \left[-\frac{\partial \mathbf{F}}{\partial \mathbf{y}}\right]^T \quad (11)$$

This approach is the adjoint method and it is best suited if the number of outputs is smaller than the number of inputs ($m < n$), since solving Equation (11) is equivalent to solve $m$ systems of equations.

### 3.5. Automatic Differentiation

Automatic Differentiation (AD) is an analytical method to calculate derivatives in computer programs based on the idea of an elementary decomposable program. Automatic differentiation consists in differentiate every line of code at the elementary level and accumulate those derivatives according to the chain-rule of differential calculus. According to Martins [7], a computer program with $n$ inputs, $l$ intermediate variables and $m$ outputs can be decomposed into elementary functions such that each computer variable $t_i$ depends only on the previous assigned variables: $t_i = T_i(t_1, ..., t_{i-1})$, where $T_i$ is an elementary function. In automatic differentiation, there are two ways of propagating the derivatives:

$$\text{FM}: \quad \frac{dt_i}{dt_j} = \delta_{ij} + \sum_{k=j}^{i-1} \frac{\partial T_i}{\partial t_k} \frac{dt_k}{dt_j} \tag{12}$$

$$\text{RM}: \quad \frac{dt_i}{dt_j} = \delta_{ij} + \sum_{k=j+1}^{i} \frac{dt_i}{dt_k} \frac{\partial T_k}{\partial t_j} \tag{13}$$

The first is called the forward mode (FM). A sweep in FM correspond to fix $j$ and increment $i$ from $i = 1$ until $i = n + l + m$. The second is called the reverse mode (RM), and a sweep corresponds to choose $i$ while $j$ changes from $j = n + l + m$ until $j = 1$. The first situation corresponds to obtain a column from the matrix whose elements are $\frac{\mathrm{d}t_i}{\mathrm{d}t_j}$, while the second corresponds to obtain a row. Therefore, calculating the program's jacobian using the RM is more efficient than using the FM if $m < n$, and the opposite is true if $m > n$. Although the cost of both modes depends on the number of sweeps, the memory cost is higher in the RM since the code must be executed once forward to store the variables $t_i$ and once backwards to perform the differentiation.

## 4. Aerodynamic Model and Framework

### 4.1. Panel Method

The aerodynamic tool developed by Cardeira [3] is an implementation of the panel method, which is a numerical technique to solve inviscid potential flows around bodies of arbitrary shape by solving the Laplace equation,

$$\nabla^2 \phi = 0 \tag{14}$$

subject to the boundary conditions,

$$\nabla \phi . \mathbf{n} = 0 \tag{15a}$$

$$\lim_{\mathbf{r} \to \infty} (\nabla \phi - \mathbf{V}_\infty) = 0 \tag{15b}$$

where $\phi$ is the velocity potential, $\mathbf{n}$ is the exterior unit normal and $\mathbf{V}_\infty$ is the free-stream velocity vector. The technique consists in distributing and finding the intensities of singularities along the body's surface to consequently determine the velocity field. Based on the third Green's identity, it is possible to prove that the velocity potential in an arbitrary point $\mathbf{P}$ in the body's surface is a function of the singularities intensities and respective distance to point $\mathbf{P}$. In addition, it may also be proved that satisfying Equation (15a) may lead this potential to be constant inside the body [8], which leads to

$$\frac{1}{4\pi} \int_{Body+Wake} \mu \mathbf{n} . \nabla \left(\frac{1}{r}\right) dS - \frac{1}{4\pi} \int_{Body} \sigma \left(\frac{1}{r}\right) dS = 0 \tag{16}$$

where $\sigma$ and $\mu$ are the source and doublet intensities, respectively, and $r$ is a distance from an arbitrary point in the body's surface. This approach is called the *Dirichlet Problem* and it was implemented by Cardeira [3]. Since Equation (16) holds for every point in the surface, the body's geometry must be discretized into smaller regions called *panels* and Equation (16) must be specified in a finite number of control points called *collocation points* to obtain a solution. Attending also the fact $\sigma$ is known since $\sigma = \mathbf{n} . \nabla \phi_\infty$, it is possible to obtain a linear system of equations in which the unknowns are the doublet intensities,

$$A_\mu \boldsymbol{\mu} = \mathbf{b} \tag{17}$$

### 4.2. Aerodynamic Framework

After surveying the methods available for sensitivity analysis, it became clear that the aerodynamic tool needed to be reformulated. Moreover, it was found that it could not handle changes in the airfoil shape along the wing span and some errors were found in the calculation of the aerodynamic coefficients. The process started by dividing the framework in five modules, each of them with specific tasks. A schematic overview of the reformulated tool is depicted in Figure 1.

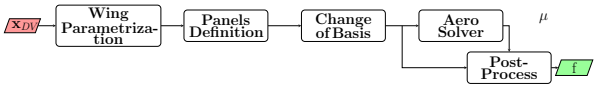


Figure 1: Flowchart illustrating the aerodynamic framework

#### Wing Parametrization

The first module is called *Wing Parametrization* and it translates the design variables $\mathbf{x_{DV}}$, into a discrete set of points representing the wing's geometry. According to Figure 2, the exterior wing shape is defined by the leading edge (LE), which is constructed with the aid of the semi span length $b_2 = \frac{b}{2}$, the sweep and dihedral angles, $\Lambda$ and $\Gamma$, respectively. Both the root and tip of the wing are defined by their respective lengths, $c_r$ and $c_t$, although the taper ratio is used instead since $\lambda = \frac{c_t}{c_r}$. They are

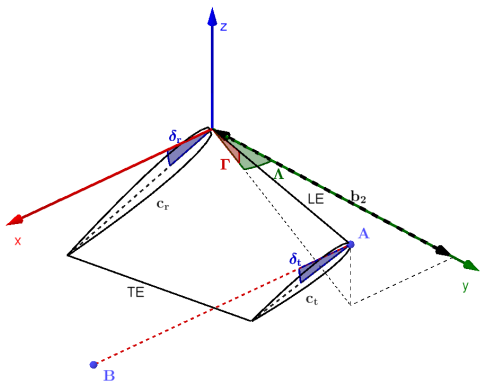


Figure 2: Geometrical description of the aircraft half wing

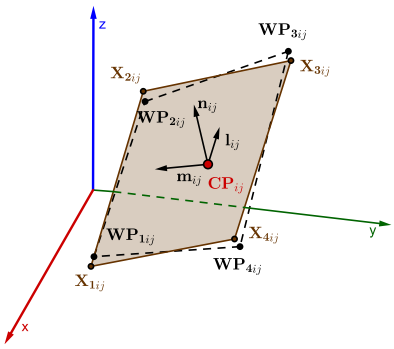


Figure 3: Panel construction through a set of four non-coplanar adjacent points.

also defined by the respective twist angles, $\delta_r$ and $\delta_t$, respectively. The wing is then discretized in the spanwise direction according to user specifications and an airfoil shape is assigned. Moreover, the local twist and chord values are fully defined since these are assumed to vary linearly along the semi wing span between the respective root and tip values. The airfoil shapes are simulated using points from four bezier curves whose control points are used to parametrize their shape. The followed methodology is described in the work of Venkataraman [9]. The resulting set of points are organized as

$$\mathbf{WP} = \begin{bmatrix} \mathbf{WP_1}^T & \mathbf{WP_2}^T & \mathbf{WP_3}^T & \mathbf{WP_4}^T \end{bmatrix}^T \tag{18}$$

which will be clear why later. In addition, the module also outputs the wing area $S$ and the mean aerodynamic chord MAC,

$$S = \left(\frac{1+\lambda}{2}\right) b\, c_r \tag{19a}$$

$$\text{MAC} = \frac{2}{3}\left(\frac{\lambda^2+\lambda+1}{\lambda+1}\right) c_r \tag{19b}$$

### Panels Definition

The next module is called *Panels Definition* and it is responsible for creating the panels and calculate associated quantities. Since the points generated in *Wing Parametrization* cannot be used to form the panels directly, it is necessary to process these points. Consequently, this module outputs the panel's corner points **PP**, the collocation points **CP**, the panel's areas $DS$ and the panel's basis vectors **LV**. Consider the panel $(i, j)$ constructed based on the input points $\mathbf{WP_1}_{ij}$ to $\mathbf{WP_4}_{ij}$, as defined in Figure 3. Note that indexes were used in vectors meaning that they are referred to the location $(i, j)$ in the mesh, opposed e.g. to the meaning of those variables in Equation (18) where the computational indexes are unrolled. It is possible to construct four auxiliary vectors as a function of the input points as

$$\mathbf{P_f}_{ij} = \mathbf{WP_2}_{ij} - \mathbf{WP_1}_{ij} \tag{20a}$$

$$\mathbf{P_s}_{ij} = \mathbf{WP_3}_{ij} - \mathbf{WP_4}_{ij} \tag{20b}$$

$$\mathbf{X_f}_{ij} = \frac{1}{2}\left(\mathbf{WP_1}_{ij} + \mathbf{WP_2}_{ij}\right) \tag{21a}$$

$$\mathbf{X_s}_{ij} = \frac{1}{2}\left(\mathbf{WP_3}_{ij} + \mathbf{WP_4}_{ij}\right) \tag{21b}$$

Using Equations (20a) and (20b), one may define the first basis vector that lies on the panel,

$$\mathbf{l}_{ij} = \frac{\left(\mathbf{P_f}_{ij} + \mathbf{P_s}_{ij}\right)}{\left\|\mathbf{P_f}_{ij} + \mathbf{P_s}_{ij}\right\|} \tag{22}$$

Using the previous definitions, the panel's corner points are automatically defined as

$$\mathbf{X_{1,2}}_{ij} = \mathbf{X_f}_{ij} \mp \frac{1}{2}\left(\|\mathbf{P_f}_{ij}\|.\mathbf{l}_{ij}\right) \tag{23a}$$

$$\mathbf{X_{3,4}}_{ij} = \mathbf{X_s}_{ij} \pm \frac{1}{2}\left(\|\mathbf{P_s}_{ij}\|.\mathbf{l}_{ij}\right) \tag{23b}$$

where the minus assignment corresponds to $\mathbf{X_1}_{ij}$ and $\mathbf{X_4}_{ij}$, and the plus corresponds to $\mathbf{X_2}_{ij}$ and $\mathbf{X_3}_{ij}$. All corner points may be organized as

$$\mathbf{PP} = \begin{bmatrix} \mathbf{X_1}^T & \mathbf{X_2}^T & \mathbf{X_3}^T & \mathbf{X_4}^T \end{bmatrix}^T \tag{24}$$

Subsequently, the unitary normal may is defined as

$$\mathbf{n}_{ij} = \frac{\mathbf{N}_{ij}}{\|\mathbf{N}_{ij}\|} \tag{25}$$

where $\mathbf{N}_{ij}$ is a normal vector which is a function of the corner points

$$\mathbf{N}_{ij} = \left(\mathbf{X_3}_{ij} - \mathbf{X_1}_{ij}\right) \times \left(\mathbf{X_4}_{ij} - \mathbf{X_2}_{ij}\right) \tag{26}$$

Both the collocation point $\mathbf{CP}_{ij}$ and panel's area $DS_{ij}$ may also be defined as a function of the corner points as

$$\mathbf{CP}_{ij} = \frac{1}{4}\left(\mathbf{X_1}_{ij} + \mathbf{X_2}_{ij} + \mathbf{X_3}_{ij} + \mathbf{X_4}_{ij}\right) \tag{27}$$

and

$$DS_{ij} = \frac{1}{2}\Big[\|\mathbf{X_B}_{ij} \times \mathbf{X_A}_{ij}\| + \|\mathbf{X_C}_{ij} \times \mathbf{X_B}_{ij}\|\Big] \quad (28)$$

where $\mathbf{X_A}_{ij} = \mathbf{X_2}_{ij} - \mathbf{X_1}_{ij}$, $\mathbf{X_B}_{ij} = \mathbf{X_3}_{ij} - \mathbf{X_1}_{ij}$, $\mathbf{X_C}_{ij} = \mathbf{X_4}_{ij} - \mathbf{X_1}_{ij}$. Finally, the last basis vector is defined as a function of $\mathbf{l}_{ij}$ and $\mathbf{n}_{ij}$ as

$$\mathbf{m}_{ij} = \mathbf{n}_{ij} \times \mathbf{l}_{ij} \quad (29)$$

Consequently, the panel's basis vectors are organized as

$$\mathbf{LV} = \begin{bmatrix}\mathbf{l}^T & \mathbf{m}^T & \mathbf{n}^T\end{bmatrix}^T \quad (30)$$

### Change of Basis

As it will be clear next, it is helpful to write the corner points **PP** in its own panel's frame of reference since the influence coefficients may then be easily calculated. Consequently, this module is called *Change of Basis* and it receives as inputs **CP**, **PP** and **LV**. A generic point $\mathbf{P} = [P_1, P_2, P_3]^T$ written in the global frame of reference may be translated to the panel's frame of reference as

$$\begin{bmatrix} P_1^{'} \\ P_2^{'} \\ P_3^{'} \end{bmatrix} = \begin{bmatrix} l_{ij1} & l_{ij2} & l_{ij3} \\ m_{ij1} & m_{ij2} & m_{ij3} \\ n_{ij1} & n_{ij2} & n_{ij3} \end{bmatrix} \begin{bmatrix} P_1 - CP_{ij1} \\ P_2 - CP_{ij2} \\ P_3 - CP_{ij3} \end{bmatrix} \quad (31)$$

where $\mathbf{CP}_{ij}$ is the origin of the panel's frame of reference and $\mathbf{P}^{'} = [P_1^{'}, P_2^{'}, P_3^{'}]^T$ is point **P** written in the panel's frame of reference. The panel's corner points in its own frame of reference are easily obtain replacing **P** appropriately. Consequently, the module's outputs may be organized in a vector as

$$\mathbf{LPP} = \begin{bmatrix}\mathbf{X_1}^{'T} & \mathbf{X_2}^{'T} & \mathbf{X_3}^{'T} & \mathbf{X_4}^{'T}\end{bmatrix}^T \quad (32)$$

### Aero Solver

The next module is called *Aero Solver* and it is responsible for assembling and solving the linear system presented in Equation (17). The module receives as inputs the vectors **LPP**, **CP**, **LV** and additionally, the free-stream airspeed $V_\infty$ and the angle of attack $\alpha$. Consequently, it outputs the aerodynamic solution $\boldsymbol{\mu}$ and the residuals **R**. Since the wing could be assumed symmetric with respect to plane $Oxz$ of Figure 2, the method of images was used to diminish the computational cost of the implementation. The residuals may be written appropriately for the computational mesh as

$$R_{ij} = \sum_{n=1}^{N}\Bigg[\sum_{m=1}^{M-1}\left(\mathcal{C}_{ijmn}\mu_{mn} + \mathcal{B}_{ijmn}\sigma_{mn}\right) + \mathcal{C}_{ijMn}\left(\mu_{(M-1)n} - \mu_{1n}\right)\Bigg] = 0 \quad (33)$$

where $\mathcal{C}_{ijmn}$ and $\mathcal{B}_{ijmn}$ are the doublet and source influences of panel $(m,n)$ on panel $(i,j)$. Since the method of images was used, it can be proven that these quantities depend only on the corner points of panel $(m,n)$, the $(i,j)$ panel's collocation point and respective image, all written in $(m,n)$ panel's frame of reference. The source intensities are easily known since

$$\sigma_{mn} = \mathbf{n}_{mn}.\mathbf{V}_\infty \quad (34)$$

### Post-Processing

The last module is responsible for calculating the aerodynamic coefficients and it is called *Post-Process*. The module accepts the angle of attack $\alpha$, the vectors **PP**, **CP**, **LV**, $DS$, the aerodynamic solution $\boldsymbol{\mu}$, and MAC and $S$ as inputs. The velocity on the panel $(i,j)$ is given by

$$\mathbf{V}_{ij} = (V_{\infty l}, V_{\infty m}, V_{\infty n})_{ij} + (v_l, v_m, v_n)_{ij} \quad (35)$$

where the first term is the free-stream and the second is the perturbation velocity, which is a function of the aerodynamic solution $\mu_{ij}$, both written in $(i,j)$ panel's frame of reference. Next, the pressure coefficient is obtained as

$$C_{p_{ij}} = 1 - \frac{|\mathbf{V}_{ij}|^2}{|\mathbf{V}_\infty|^2} \quad (36)$$

Knowing the pressure coefficients in the panels, the aerodynamic coefficients are calculated by numerical integration as

$$C_L = -\frac{2}{S}\sum_{i}^{M-1}\sum_{j}^{N} C_{p_{ij}} DS_{ij}\left(\mathbf{n}_{ij}.\mathbf{e}_L\right) \quad (37a)$$

$$C_D = -\frac{2}{S}\sum_{i}^{M-1}\sum_{j}^{N} C_{p_{ij}} DS_{ij}\left(\mathbf{n}_{ij}.\mathbf{e}_D\right) \quad (37b)$$

$$\mathbf{C_M} = -\frac{2}{S.\,l_0}\sum_{i}^{M-1}\sum_{j}^{N} C_{p_{ij}} DS_{ij}\left(\mathbf{CP}_{ij} \times \mathbf{n}_{ij}\right) \quad (37c)$$

where $\mathbf{e}_L$ and $\mathbf{e}_D$ are the unit vectors in the directions of the lift and drag, respectively, and they are a function of the angle of attack $\alpha$. The variable $l_0$ is an appropriated reference length, which corresponds to MAC and $b$ for the pitching and rolling moment coefficients, respectively.

## 5. Sensitivity Analysis Framework

The sensitivity analysis framework is depicted in Figure 4. Each module presented before has its own sensitivity analysis, where the jacobians of the outputs with respect to the inputs are calculated. After, the sensitivities of the functions of interest $f$, with respect to the design variables $\mathbf{x_{DV}}$, are calculated propagating the intermediate jacobians according to the chain-rule of differential calculus.

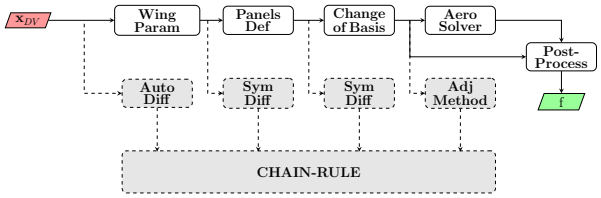


Figure 4: Flowchart illustrating the sensitivity analysis framework

### Mathematical Formulation

First, the vector of design variables must be defined. It is characterized by a segment which contains planform related parameters, a segment containing the control points which parametrize the airfoil shape in each cross section and the angle of attack. Thus, it may be written as

$$\mathbf{x_{DV}} = \begin{bmatrix} \alpha & \mathbf{x_{geo}}^T & \mathbf{x_{airfoil}}^T \end{bmatrix}^T \tag{38}$$

where each of the right hand side vectors are

$$\mathbf{x}_{geo} = \begin{bmatrix} \Lambda & \Gamma & \delta_r & \delta_t & b & c_r & \lambda \end{bmatrix}^T \tag{39a}$$

$$\mathbf{x_{airfoil}}^T = \bigcup_j \begin{bmatrix} A_x & ... & L_x & A_y & ... & L_y \end{bmatrix}_j \quad \forall j \in \{1, ..., N+1\} \tag{39b}$$

### Adjoint Method

Considering both the inputs of the *Aero Solver* and *Post Process* modules, it can be defined an intermediate information as

$$\mathbf{x_2} = \begin{bmatrix} \mathbf{x_1}^T & DS^T & \mathbf{LPP}^T & S & \text{MAC} & \alpha & V_\infty \end{bmatrix}^T \tag{40}$$

where $\mathbf{x_1}$ is defined as

$$\mathbf{x_1} = \begin{bmatrix} \mathbf{PP}^T & \mathbf{CP}^T & \mathbf{LV}^T \end{bmatrix}^T \tag{41}$$

such that $\mathbf{R} = \mathbf{R}(\mathbf{x_2}, \boldsymbol{\mu})$ and $f = f(\mathbf{x_2}, \boldsymbol{\mu})$. Since the size of $\mathbf{x_2}$ is much larger than the size of $f$, the adjoint method is the best suited to calculate the sensitivity of $f$ with respect to $\mathbf{x_2}$ as

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x_2}} = \frac{\partial f}{\partial \mathbf{x_2}} + \begin{bmatrix} \boldsymbol{\psi} \end{bmatrix}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x_2}} \tag{42a}$$

$$\left[\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\right]^T \begin{bmatrix} \boldsymbol{\psi} \end{bmatrix} = -\left[\frac{\partial f}{\partial \boldsymbol{\mu}}\right]^T \tag{42b}$$

### Chain-Rule

The chain-rule is used to ultimately calculate the sensitivities of the interest functions with respect to the design variables as

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x_{DV}}} = \frac{\mathrm{d}f}{\mathrm{d}\mathbf{x_2}} \frac{\mathrm{d}\mathbf{x_2}}{\mathrm{d}\mathbf{x_{DV}}} \tag{43}$$

where $\frac{\mathrm{d}\mathbf{x_2}}{\mathrm{d}\mathbf{x_{DV}}}$ corresponds to

$$\frac{\mathrm{d}\mathbf{x_2}}{\mathrm{d}\mathbf{x_{DV}}} = \begin{bmatrix} \left[\frac{\mathrm{d}\mathbf{x_1}}{\mathrm{d}\mathbf{x_{DV}}}\right]^T & \left[\frac{\mathrm{d}DS}{\mathrm{d}\mathbf{x_{DV}}}\right]^T & \left[\frac{\mathrm{d}\mathbf{LPP}}{\mathrm{d}\mathbf{x_{DV}}}\right]^T & \left[\frac{\partial S}{\partial \mathbf{x_{DV}}}\right]^T & \left[\frac{\partial \mathrm{MAC}}{\partial \mathbf{x_{DV}}}\right]^T & \left[\frac{\partial \alpha}{\partial \mathbf{x_{DV}}}\right]^T & [0] \end{bmatrix}^T \tag{44}$$

Note that some entries were already replaced by the respective partial derivatives where explicit dependence is observed. The remaining derivatives are assembled according to the variable dependencies presented for each module:

$$\frac{\mathrm{d}\mathbf{x_1}}{\mathrm{d}\mathbf{x_{DV}}} = \frac{\partial \mathbf{x_1}}{\partial \mathbf{WP}} \frac{\partial \mathbf{WP}}{\partial \mathbf{x_{DV}}} \tag{45}$$

$$\frac{\mathrm{d}DS}{\mathrm{d}\mathbf{x_{DV}}} = \frac{\partial DS}{\partial \mathbf{WP}} \frac{\partial \mathbf{WP}}{\partial \mathbf{x_{DV}}} \tag{46}$$

$$\frac{\mathrm{d}\mathbf{LPP}}{\mathrm{d}\mathbf{x_{DV}}} = \frac{\partial \mathbf{LPP}}{\partial \mathbf{x_1}} \frac{\partial \mathbf{x_1}}{\partial \mathbf{WP}} \frac{\partial \mathbf{WP}}{\partial \mathbf{x_{DV}}} \tag{47}$$

### Sensitivities of Wing Parametrization

This module calculates three jacobians: $\frac{\partial S}{\partial \mathbf{x_{DV}}}$, $\frac{\partial MAC}{\partial \mathbf{x_{DV}}}$ and $\frac{\partial \mathbf{WP}}{\partial \mathbf{x_{DV}}}$. The non zero derivatives of the first two are calculated by hand since the expressions are simple and correspond to differentiate Equations (19a) and (19b) with respect to their dependencies. The last jacobian is calculated with the aid of automatic differentiation. A choice had to me made which AD mode should be implemented. It was concluded that the forward mode is more efficient since the number of outputs is larger than the number of inputs for the mesh sizes expected to be used in the optimization problems. The implementation was benchmarked with the complex-step derivative to compare both the results and performance. As it may be observed in Table 1, the AD implementation is faster, with savings up to 40.5%.

| **No. panels** | 50 | 200 | 450 | 800 | 1250 | 1800 |
|---|---|---|---|---|---|---|
| **CSD time [s]** | 0.639 | 3.011 | 9.332 | 19.815 | 38.970 | 64.553 |
| **AD time [s]** | 0.380 | 1.151 | 4.191 | 10.400 | 27.219 | 55.999 |
| **Savings [%]** | 40.5 | 61.8 | 55.1 | 47.5 | 30.2 | 13.3 |

Table 1: Computational cost of the *Wing Parametrization* sensitivity analysis module.

### Sensitivities of Panels Definition

The sensitivity analysis of *Panels Definition* corresponds to the calculation of four jacobians: $\frac{\partial \mathbf{PP}}{\partial \mathbf{WP}}$, $\frac{\partial \mathbf{CP}}{\partial \mathbf{WP}}$, $\frac{\partial \mathbf{LV}}{\partial \mathbf{WP}}$ and $\frac{\partial DS}{\partial \mathbf{WP}}$. These matrices were all obtained with the aid of symbolic differentiation. Although the procedure was quite lengthy, this approach allowed to obtain huge computational savings since only different from zero partial derivatives were calculated, algebraic simplifications were made and MATLAB® vectorization techniques were applied. According to Table 2, the implementation

can be about 1000 times faster, comparing with the CSD and AD. Since this approach is very susceptible to errors, the implementation was benchmarked with AD and the complex-step derivative. Figure 5 shows the absolute error for each entry of $\frac{\partial \mathbf{CP}}{\partial \mathbf{WP}}$ when benchmarked with the CSD and AD. As observed, the error is bounded and really small. Similar results were obtained for the remaining jacobians thus validating the module.

| **No. panels** | 200 | 450 | 800 | 1250 | 1800 |
|---|---|---|---|---|---|
| **CSD time** [s] | 20.05006 | 109.4079 | 289.9539 | 761.1905 | 2821.0137 |
| **AD time** [s] | 22.93737 | 148.1591 | 599.1266 | 2296.589 | 8623.2376 |
| **SD time** [s] | 0.117399 | 0.294621 | 0.554629 | 1.005761 | 1.822698 |
| **Savings CSD** [%] | 99.4 | 99.7 | 99.8 | 99.9 | 99.9 |
| **Savings AD** [%] | 99.5 | 99.8 | 99.9 | 100 | 100 |

Table 2: Computational cost of the *Panels Definition* sensitivity analysis module.

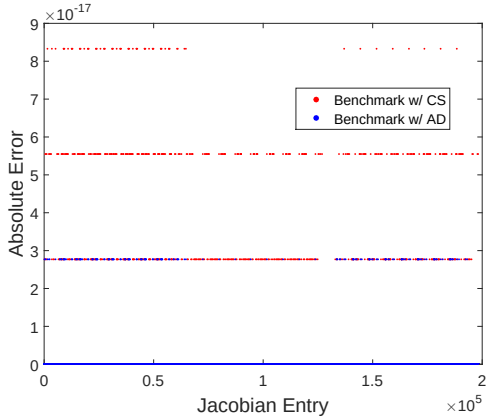


Figure 5: Absolute error for all the entries of $\frac{\partial \mathbf{CP}}{\partial \mathbf{WP}}$

### Sensitivities of Change of Basis

This module is responsible for calculating the jacobians $\frac{\partial \mathbf{LPP}}{\partial \mathbf{PP}}$, $\frac{\partial \mathbf{LPP}}{\partial \mathbf{CP}}$ and $\frac{\partial \mathbf{LPP}}{\partial \mathbf{LV}}$ or simply $\frac{\partial \mathbf{LPP}}{\partial \mathbf{x_1}}$. The building blocks to calculate these matrices are obtained differentiating Equation (31) with respect to $\mathbf{P}$, $\mathbf{CP}_{ij}$, $\mathbf{l}_{ij}$, $\mathbf{m}_{ij}$ and $\mathbf{n}_{ij}$. The differentiation was performed by hand since the expression was easy enough. No benchmark is provided here, although the module's verification was indeed performed.

### Sensitivities of Aero Solver

Paying attention to the inputs of *Aero Solver* module, it may be concluded that the respective sensitivity analysis corresponds to the calculation of six jacobians that will be used to construct $\frac{\partial \mathbf{R}}{\partial \mathbf{x_2}}$ and $\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}$, where these will be used in the adjoint method, in Equations (42a) and (42b). The sensitivity analysis corresponds to differentiate Equation (33) with respect to the inputs of Aero Solver, which was performed by hand with the aid of symbolic differentiation. The reason to follow this methodology was the same as for the sensitivity analysis of *Panels Definition*. The benefits in computational terms is described in Table 3, where it may be observed that savings of about 99.5% were obtained, when comparing the followed approach with the forward mode of AD. Since obtaining the derivatives using this approach is very susceptible to programming errors, a benchmark with AD differentiation was performed. Figure 6 presents the absolute error for all entries of $\frac{\partial \mathbf{R}}{\partial \mathbf{LPP}}$, taking the respective values calculated using AD as reference. As it can be observed, the error is bounded and small. A similar analysis was performed for all the produced jacobians and the results were similar, proving that the implementation was correct.

| **No. panels** | 50 | 200 | 450 |
|---|---|---|---|
| **AD time** [s] | 749.964 | 8653.022 | 40582.198 |
| **SD time** [s] | 3.796 | 40.797 | 222.411 |
| **Savings** [%] | 99.5 | 99.5 | 99.5 |

Table 3: Computational cost of the *Aero Solver* sensitivity analysis module.

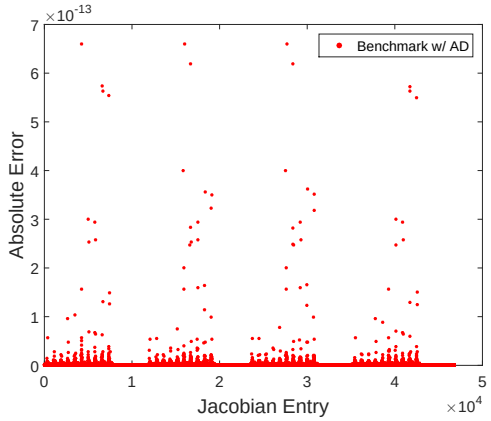


Figure 6: Absolute error for all the entries of $\frac{\partial \mathbf{R}}{\partial \mathbf{LPP}}$

### Sensitivities of Post Process

The sensitivity analysis of *Post Process* corresponds to calculate the jacobians $\frac{\partial f}{\partial \mathbf{x_2}}$ and also $\frac{\partial f}{\partial \boldsymbol{\mu}}$. These matrices are required to the adjoint method in Equations (42a) and (42b). These are calculated using the reverse mode of automatic differentiation since the module's number inputs is much higher than the number of outputs. Furthermore, the number of outputs is, at most, equal to five, corresponding to all the aerodynamic coefficients the program may compute.

### Benchmark with Finite Differences

After the sensitivity analysis framework had been constructed, it was benchmarked with the finite-differences method to guarantee that the program was free of programming errors and also to measure its performance. Figure 7 shows the absolute error of each entry of the aerodynamic coefficients sensitivities with respect to the design variables, when compared to the finite-differences method with a step size of $h = 10^{-7}$. As observed, the error is at most of $\mathcal{O}(10^{-6})$, thus verifying the framework's results. On the other hand, Table 4 shows the time spent by the sensitivity analysis framework and the time spent by the implementation using FD, taking the time spent by the aerodynamic model as reference, for an increasing number of design variables. As it can be observed, using the sensitivity analysis framework translates into increased time savings for increasing number of design variables. This result allows to conclude that the new tool is much more efficient than the implementation of FD for accurate wing description.

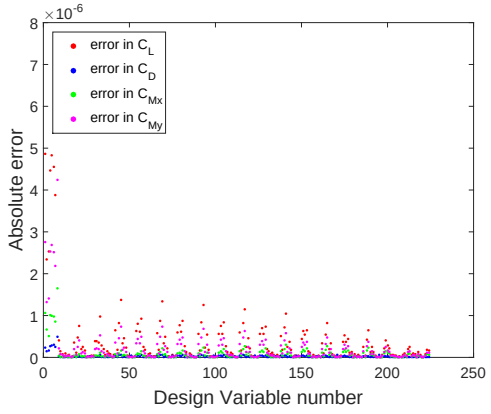


Figure 7: Absolute error of the aerodynamic coefficients sensitivities w.r.t. $\mathbf{x_{DV}}$, benchmarked with FD

| No. panels | 32 | 64 | 128 | 192 | 240 |
|---|---|---|---|---|---|
| No. DV | 80 | 128 | 224 | 320 | 392 |
| $t_{model}$ [s] | 0.083 | 0.239 | 0.876 | 1.946 | 3.037 |
| $t_{sens}/t_{model}$ [-] | 41.65 | 34.56 | 27.77 | 25.07 | 24.01 |
| $t_{FD}/t_{model}$ [-] | 61.47 | 120.07 | 224.78 | 316.24 | 390.19 |
| savings [%] | 32.2 | 71.1 | 87.6 | 92.1 | 93.8 |

Table 4: Runtime comparison for increasing number of design variables

## 6. Aerodynamic Optimization

Two optimization problems were solved to illustrate the benefits of gradient-based optimization using efficient sensitivity analysis. In each example, the optimization problem was solved using both the new sensitivity analysis framework (FW) and the finite-difference (FD) method, to compare the required computational time, the number of iterations and the number of function evaluations.

### Wing Planform Optimization

The first optimization problem is expressed as

$$\begin{aligned} &\text{minimize} && C_D \\ &\text{w.r.t.} && \mathbf{x_{DV}} \\ &\text{subject to} && C_L = 0.3, \quad S = S_0 \\ & && \mathbf{x}^L \leq \mathbf{x_{DV}} \leq \mathbf{x}^U \end{aligned} \tag{48}$$

where $\mathbf{x_{DV}}$ corresponds to planform related variables and the angle of attack. Table 5 shows both the baseline and optimized values of the design variables and output functions. As observed, the wing drag was reduced by 33% mainly due to an increase of the aspect ratio and lift redistribution changing the taper ratio to near 0.4. Moreover, the results obtained using the framework are quite similar to the results obtained using FD. The computational cost, number of function evaluations and iterations were also tracked and they are presented in Table 6. As observed, the computational cost of FD is less than the cost of using the framework. The reason is related with the fact that the cost of evaluating the aerodynamic model to estimate the gradient is less than evaluating the sensitivity analysis framework when very few design variables are being considered.

| Design variables | Baseline | Optimized FW | Optimized FD |
|---|---|---|---|
| $\alpha\,[^\circ]$ | 4 | 3.177316 | 3.177317 |
| $b\,[\mathrm{m}]$ | 6 | 8.000000 | 8.000000 |
| $c_r\,[\mathrm{m}]$ | 1 | 1.079303 | 1.079325 |
| $\lambda$ | 1 | 0.389785 | 0.389757 |
| **Outputs** | **Baseline** | **Optimized FW** | **Optimized FD** |
| $C_D$ | 0.012777 | 0.008595 | 0.008595 |
| $C_L$ | 0.314576 | 0.300000 | 0.300000 |
| $C_{M_x}$ | 0.071026 | 0.064134 | 0.064133 |
| $C_{M_y}$ | 0.221416 | 0.287423 | 0.287421 |
| $S$ | 6.000000 | 6.000000 | 6.000000 |
| $AR$ | 6.000000 | 10.66667 | 10.666667 |

Table 5: Baseline, optimized design vector and output values in the first optimization problem

| Gradient Calculation Method | Time [s] | Iterations | Function Evaluations |
|---|---|---|---|
| Sensitivity Framework | 4681.8 | 11 | 31 |
| Forward Finite Differences | 1314.7 | 11 | 79 |

Table 6: First case optimization performance benchmark between different sensitivity analysis methods

### Full Wing Optimization

The second problem is similar to the first, except that the pitching moment coefficient was constrained to the baseline value and $\mathbf{x_{DV}}$ includes now all design parameters, as defined in Equation (38). Both the baseline and optimized wing configurations are depicted in Figure 8.

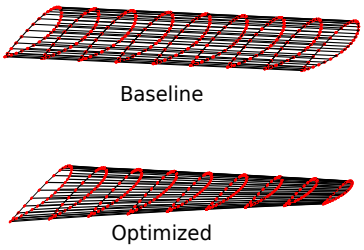


Figure 8: Baseline and optimized wing configurations

The output functions and part of the design vector baseline and optimized values are presented in Table 7, for both approaches of sensitivity analysis. As observed, the drag was reduced in 72% again due to increased aspect ratio, lift reduction and lift redistribution. The performance of both approaches

| Design variables | Baseline | Optimized FW | Optimized FD |
|---|---|---|---|
| $\alpha\,[^\circ]$ | 4 | 1.000000 | 1.000000 |
| $\Lambda\,[^\circ]$ | 0 | 6.011407 | 6.007157 |
| $\Gamma\,[^\circ]$ | 0 | 5.000000 | 5.000000 |
| $\delta_r\,[^\circ]$ | 0 | 2.230745 | 2.227752 |
| $\delta_t\,[^\circ]$ | 0 | -0.124150 | -0.124778 |
| $b\,[\mathrm{m}]$ | 6 | 8.000000 | 8.000000 |
| $c_r\,[\mathrm{m}]$ | 1 | 1.000000 | 1.000000 |
| $\lambda$ | 1 | 0.500000 | 0.500000 |
| Outputs | Baseline | Optimized FW | Optimized FD |
| $C_D$ | 0.013429 | 0.003724 | 0.003725 |
| $C_L$ | 0.313735 | 0.300000 | 0.300000 |
| $C_{M_x}$ | 0.071235 | 0.065081 | 0.065076 |
| $C_{M_y}$ | 0.220788 | 0.220788 | 0.220788 |
| $S$ | 6.000000 | 6.000000 | 6.000000 |
| $AR$ | 6.000000 | 10.666667 | 10.666667 |

Table 7: Baseline, optimized design vector and output values in the second optimization problem

was also measured and presented in Table 8. As it can be observed, the number of function evaluations using the new framework is much less when compared to the use of FD. As a consequence, the optimization time is also much less, about 9 times faster than the implementation using FD, proving its efficiency for accurate wing optimization using many design variables.

| Gradient Calculation Method | Time [s] | Iterations | Function Evaluations |
|---|---|---|---|
| Sensitivity Framework | 7607.1 | 44 | 75 |
| Forward Finite Differences | 68726.9 | 44 | 10155 |

Table 8: Second optimization case performance benchmark between different sensitivity analysis methods

## 7. Conclusions

An efficient aerodynamic optimization tool was developed. To accomplish that, a sensitivity analysis framework was constructed based on exact gradient calculation using analytical methods such as automatic differentiation, symbolic differentiation and the adjoint method. Special concern was employed to obtain high computational efficiency, which was translated in combining good programming practices in MATLAB® with code simplifications, whenever it was possible. Aerodynamic optimization problems were solved to illustrate the performance of the new sensitivity analysis framework and it was concluded that savings up to 90% in the computational cost may be achieved.

## References

[1] Jaroslaw Sobieski and Raphael T. Haftka. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural optimization*, 14(1):1–23, 1997.

[2] João Almeida. Structural dynamics for aeroelastic analysis. Master's thesis, Instituto Superior Tcnico, November 2015.

[3] André Cardeira. Aeroelastic analysis of aircraft wings. Master's thesis, Instituto Superior Tcnico, December 2014.

[4] Ashok D. Belegundu and Tirupathi R. Chandrupatla. *Optimization concepts and applications in engineering*. Cambridge University Press, 2011.

[5] Klaus Schittkowski and Ya-xiang Yuan. Sequential quadratic programming methods. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[6] Jacques Peter and Richard Dwight. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, 39(3):373–391, 2010.

[7] J. R. R. A. Martins and John Hwang. Review and unification of methods for computing derivatives of multidisciplinary computational models. *AIAA Journal*, 51(11):2582–2599, November 2013. DOI: 10.2514/1.J052184.

[8] Joseph Katz and Allen Plotkin. *Low-speed aerodynamics*, volume 13. Cambridge university press, 2001.

[9] P. Venkataraman. A new procedure for airfoil definition. *AIAA Paper*, pages 95–1875, 1995.