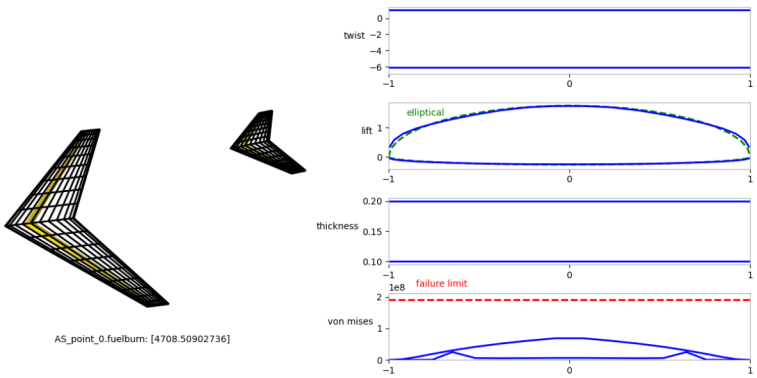
TÉCNICO LISBOA




# Aircraft Optimal Design

Aerospace Engineering

**Luís Martins Pacheco**

**Number 96425**

**João Pedro Gaspar**

**Number 96930**

Report #3

04/06/2023

# Contents

# 1 Introduction

The primary objective of this study is to employ the software tools OpenMDAO and OpenAeroStruct for optimization purposes. Initially, OpenMDAO was utilized to minimize the Rosenbrock Function, a well-known test function commonly employed in optimization tasks. Subsequently, an examination of the model´s $N^2$ diagram was conducted. The minimization process was explored both in an unconstrained manner and with the inclusion of a constraint, employing both finite difference and analytic methods for the determination of the gradients.

Furthermore, an extensive investigation was undertaken to analyze the structural analysis model utilized in OpenAeroStruct. To conclude the study, an aerodynamic optimization of an isolated aircraft wing was performed. The design variables considered for this optimization were a combination of the angle of attack, twist, and chord. An additional constraint pertaining to aerodynamics was incorporated into OpenAeroStruct. The aforementioned optimizations were then repeated, taking into account the new constraint, and the effects of this constraint on the optimal solution were thoroughly analyzed.

# 2 Constrained Optimization using OpenMDAO

Considering the Rosenbrock function defined as:

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \tag{1}$$

The partial derivatives of the Rosenbrock function, with respect to $x_1$ and $x_2$ were analytically determined to be:

$$\frac{\partial f}{\partial x_1} = 400x_1^3 - 400x_2x_1 + 2x_1 - 2 \tag{2}$$

$$\frac{\partial f}{\partial x_2} = 200(x_2 - x_1^2) \tag{3}$$

A model for the Rosenbrock function was defined in OpenMDAO. OpenMDAO models can exhibit complex hierarchies consisting of interconnected groups and components, featuring numerous connections between them. Visualizing the structure of these models is often advantageous in gaining a comprehensive understanding of their organization. To address this need, OpenMDAO provides a model visualization tool known as the $N^2$ diagram. The $N^2$ diagram represents a modified version of a design-structure matrix, where the model hierarchy is presented on the left-hand side. Along the diagonal of the diagram, each input and output of the components is depicted. The off-diagonal blocks indicate the data connections between components. Feed-forward connections are illustrated in the upper triangle, while feed-back connections, which introduce cyclic dependencies, will be shown in the lower triangle. Additionally, by hovering over any block on the diagonal, the associated incoming and outgoing connections can be visually identified through the use of highlighted arrows. For the simple case of the Rosenbrock

function the $N^2$ diagram of Figure 1 was obtained.

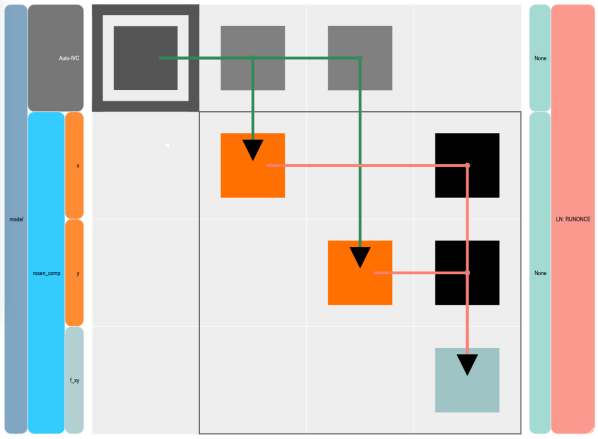


Figure 1: $N^2$ diagram for the Rosenbrock function

The Rosenbrock function $N^2$ diagram is relatively simple. To calculate the value of the Rosenbock function, the values of *x* and *y* are required. The optimizer provides the component Rosen with this inputs. There are no feed-back connections, nor coupling between *x* and *y*. The inputs will be used in $f_{xy}$, representing Equation 1, which will output a value of the function to be optimized.

The unconstrained minimization problem of the Rosenbrock function was solved numerically with OpenMDAO. For the initial guess, point $x = (-1.2, 1.0)$ was used and the partial derivatives were estimated using the forward finite-difference method that exists within OpenMDAO [1].

The *ScipyOptimizeDriver* for optimization was used. The optimizer used was the Sequential Least SQuares Programming (SLSQP), which is the default. All other options for the *ScipyOptimizeDriver* were also used with their default values. Some of the most important are *maxiter*, the maximum number of iterations before termination, which is set to 200 and *tol*, which is the tolerance for termination, which was set $10^{-6}$ [2].

The optimization was completed successfully, with *Exit mode 0*. The optimized solution was $f(x) = 6.50393674 \times 10^{-8}$ and was determined to be at point $x = (0.99976139, 0.99951384)$. To complete this optimization 34 iterations and gradient evaluations were required along with 47 function evaluations. The history of the uncontrained optimization is depicted in Figure 2.

The constrain $g(x) = x1+x2 \leq 1$ was added to the minimization problem of the Rosenbrock function. For the initial guesses of $x = (1.2, 1.0)$ and $x_{opt} = (0.99976139, 0.99951384)$ the optimization results of Table 1 were obtained.

Table 1: Optimization results and computational cost of the minimization of the Rosenbrock function subject to the constrain $g(x) = x1 + x2 \leq 1$. Finite Differences were used in gradient calculations.

| Initial Guess | Func_Val | X_val | Y_val | Iter | Func_Eval | Grad_Eval | Exit Mode |
|---|---|---|---|---|---|---|---|
| ( -1.2, 1.0) | 0.14560738 | 0.61882245 | 0.38117755 | 22 | 29 | 22 | 0 |
| X_opt | 0.14560748 | 0.61882595 | 0.38117405 | 12 | 20 | 12 | 0 |

As can be seen from Table 1, for the constrained optimization problem when the value of the optimum solution obtained from the unconstrained minimization problem is used for the initial guess, the optimal

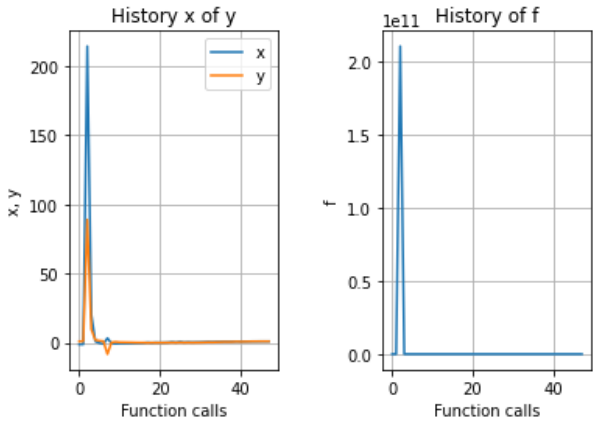



Figure 2: History of the function values and inputs x and y for the unconstrained minimization

Table 2: Optimization results and computational cost of the minimization of the Rosenbrock function subject to the constrain $g(x) = x1 + x2 \leq 1$. Analytical derivatives were used in gradient calculations.

| Initial Guess | Func_Val | X_val | Y_val | Iter | Func_Eval | Grad_Eval | Exit Mode |
|---|---|---|---|---|---|---|---|
| (-1.2, 1.0) | 0.14560715 | 0.61881163 | 0.38118837 | 20 | 28 | 20 | 0 |
| X_opt | 0.14560702 | 0.61879605 | 0.38120395 | 13 | 21 | 13 | 0 |

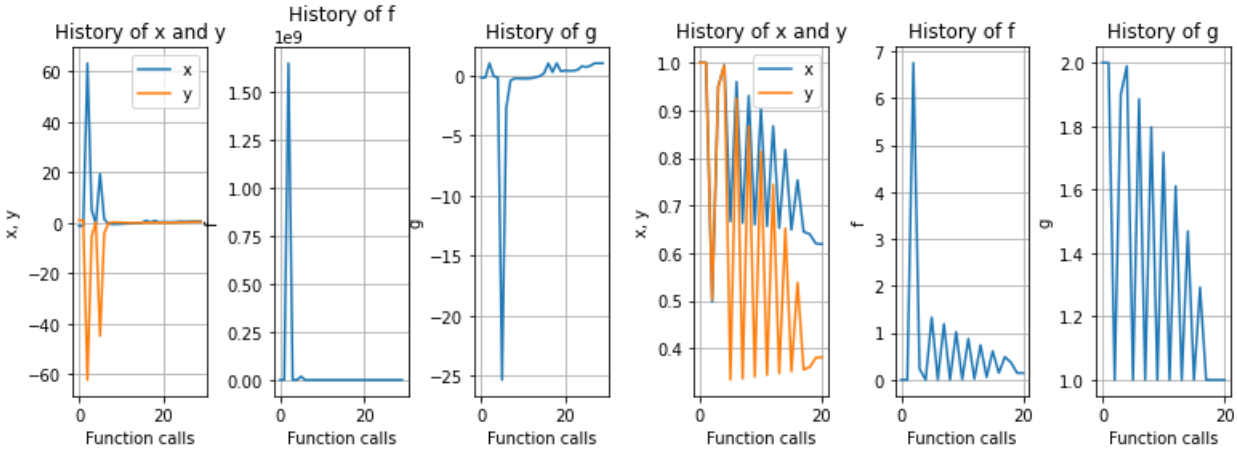
solution is reached with less computational cost. This may be explained by the fact that the optimal solution for the constrained optimization is closer to the optimal solution of the unconstrained problem than to (-1.2, 1). Despite this detail for both initial guesses the optimization was successful, and reached very similar results.

Using profiling capability it was determined that for the initial guess (-1.2, 1), the Rosenbrock function was called was 73 times. Out of those, 44 were for use in the gradient calculation. This is in accordance to the fact that the number of gradient evaluations is 22. As there are two input and one output variables, when determining the partial derivatives using forward finite differences, a calculation of the value of the Rosenbrock function for a certain perturbation set size h, for each input is calculated, meaning two calculations of the Rosenbrock function per gradient evaluation. The number of time the constrained was verified was 29, the same as the number of time the the Rosenbrock function was evaluated without gradient purposes. The CPU time required was 0.421875 seconds approximately.

Taking $x_{opt}$ as the initial guess, and performing the same procedure it was determined that 44 function calls were performed, 24 of those function calls being related to gradient evaluations for the reasons explained previously. The number of constrain verifications were 20 times. The CPU time required was 0.390625 seconds approximately.

Once again, it can be inferred that the computational power needed for the minimization process was lower for the initial guess $x_{opt}$. The functions used this particular case were simple, and did not require much computational time, but it can be seen that more than half of the function calls were due to gradient evaluations. It therefore can be expected that an efficient way to calculate the gradients will be crucial to the optimization success.

The constrained optimization was performed again but this time using the analytic partial derivatives from Equations 2 and 3. The results obtained are present in Table 2.




(a) (b)

Figure 3: History of the function values and inputs x and y for the constrained minimization using finite differences. On the left with initial guess (-1.2, 1.0) and on the right $X_{opt}$.

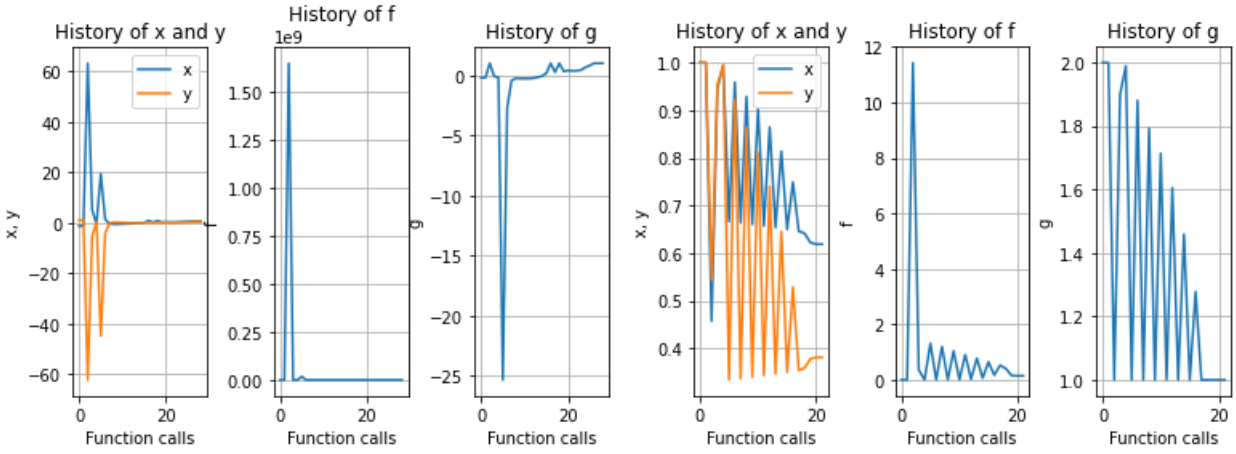
Upon comparing Table 1 and 2, it becomes evident that the computational cost associated with employing Finite Differences or analytically defined partial derivatives is quite similar in the specific case under study. This similarity primarily stems from the fact that both the Rosenbrock function and the analytically defined derivatives do not require significant computational resources. It is important to acknowledge that opting for Finite Differences instead of analytically defined partial derivatives introduces a slight error into the optimization process.

However, if the function under evaluation were considerably more complex in its analysis, it is expected that utilizing analytically defined partial derivatives would prove to be more efficient, as there would be no need to compute the function itself in order to determine the derivatives, as it is done in finite differences. Additionally, the selection of an appropriate step size is a crucial consideration when employing Finite Differences, as an ill-suited step size may yield sub-optimal gradient results.

Furthermore, when derivatives are obtained via the chain rule the choice of differentiation mode, whether forward or backward, can significantly impact the efficiency of gradient calculation methods. For instance, when utilizing the backward mode with a scenario involving a limited number of output variables but multiple input variables, the utilization of analytically defined derivatives would be notably more efficient than using Finite Differences.

# 3 Analysis Models in OpenAeroStruct

The numerical solution of the structural model in OpenAeroStruct initiates with the establishment of a spatial beam element. This element represents a straight bar with a cross section of arbitrary shape, capable of supporting axial and transverse forces, as well as moments. Moreover, the spatial beam element can deform not only along its axial direction but also in directions perpendicular to its axis. The spatial beam element possesses a total of six degrees of freedom (DOFs) at each of its nodes, comprising three translational displacements in the x, y, and z directions, along with three rotational




(a) (b)

Figure 4: History of the function values and inputs x and y for the constrained minimization with analytical partial derivatives. On the left with initial guess (-1.2, 1.0) and on the right $X_{opt}$.

DOFs around the x, y, and z axes, resulting in a cumulative sum of 12 DOFs.

The spatial beam element is obtained through the superimposition of a plane beam element under bending condition, a torsional bar, and a truss element. Subsequently in OpenAeroStruct, the structural model undergoes discretization, and a grid is generated. The structure is divided into multiple elements in the spanwise direction, and the discretization of the spar follows the same spanwise mesh discretization as in aerodynamics. The maximum radius of the spar is determined by the thickness-to-chord ratio. The initial mesh determines the locations of the finite element method (FEM) nodes. All of these computations are performed in Class ComputeNodes.

The geometric properties of a spatial beam element, that will be required for the forthcoming calculations such as the moments of inertia of the y and z axis, Iy and Iz respectively, the polar moment of inertia J and the cross-sectional area are calculated in OpenAeroStruct´s class SectionPropertiesTube for each FEM element.

If the equilibrium of the spatial beam element is expressed by Equation 4 and the displacement vector for the spatial beam element is represented as $u_e = [u_1 v_1 w_1 \alpha_{x1} \alpha_{y1} \alpha_{z1} u_2 v_2 w_2 \alpha_{x_1} \alpha_{x_2} \alpha_{x_1}]^T$

$$[k_e]\{u_e\} = \{f_e\} \tag{4}$$

then the local spatial beam element stiffness matrix is denoted as $K_e$. This matrix is a combination of stiffness matrices from the truss, plane beam, and torsional bar elements. The local stiffness matrix is computed in $local\_stiff.py$ and all the matrices for each node are stored in an array of size (ny-1,12,12). The formulation of the matrix $[K_e]$ is carried out in the local reference frame, where the x-axis aligns with the longitudinal axis of the beam element.

To convert the local degrees of freedom into global degrees of freedom within a global reference frame, a series of three rotations are performed. This is done in Class LocalStiffTransformed using a transformation matrix obtained in class Transform. Subsequently, the local node entries are placed into the corresponding global node entries in the global matrix K.

To solve the structural problem the exterior forces applied to the spatial beam need to be calculated. The loads on the structure due to the thrust of engines are computed in class ComputeThrustLoads. The loads on the structure due to point masses, due to the weight of the wing structure and due to the distributed fuel within the wing are calculated in classes ComputePointMassLoads, StructureWeightLoads, and FuelLoads, respectivelly. Class TotalLoads adds all the loads from aerodynamics, structural weight and fuel weight. Class CreateRHS computes the vector on the right-hand-side of Equation 5 for each node. The RHS is based on the loads, and for the aerostructural case, these are recomputed at each design point based on the aerodynamic loads.

With the global stiffness matrix [K] and load vector F in hand, the linear equation

$$[k]\{u\} = \{f\} \tag{5}$$

is solved to obtain the displacement vector *u* which encompasses the nodal translations and rotations. This is done in class FEM. Incorporating these nodal displacements into the original grid nodes allows the determination of the deformed shape of the structure. This new shape is used by the aerodynamic solver to compute the new aerodynamic loads.

From the translation and rotation displacements, the normal and shear stresses can be computed. By considering the normal and shear stresses, the Von Mises stresses at each node are calculated in Class VonMisesTube). Applying the Von Misses criteria it can be determined if the material will suffer plastic deformation or even fracture. If the Von Misses stresses are superior to the yield stresses the material will yield and this condition will be given by Class FailureExact. Additionally, the total strain energy of the spar can be determined based on the integration of the load vectors with the displacement, which is done in Class Energy.

In Figure 5 a detailed flowchart of the numerical implementation of the structural analysis in OpenAeroStruct is presented with the most important classes and routines and their inputs and outputs.

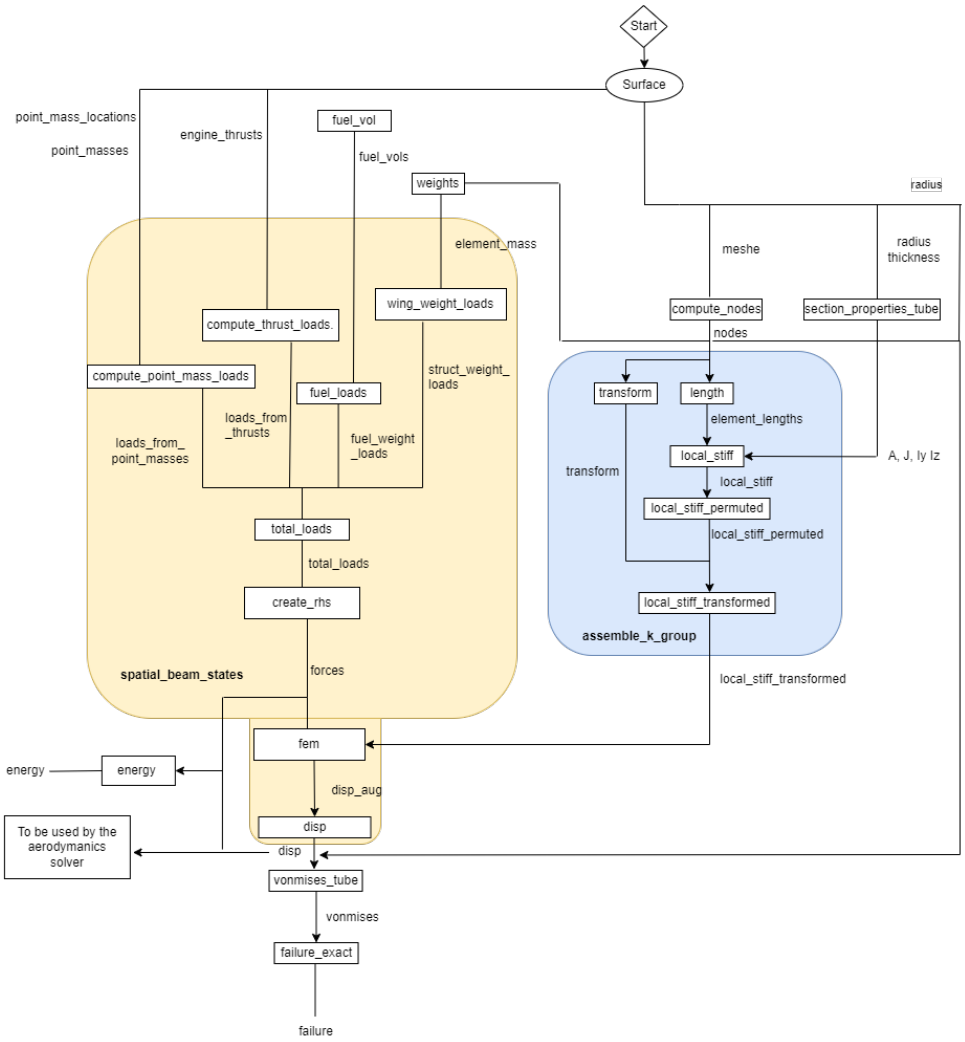



Figure 5: Flowchart of the Structures model in OpenAeroStruct

# 4 Aerodynamic Optimization using OpenAeroStruct

In the optimization problem of the aerodynamic design for minimizing drag on an isolated aircraft wing assuming inviscid drag only, an optimum elliptical lift distribution across the span of the wing is sought. This distribution was found to theoretically minimize induced drag (drag due to lift) for a given amount of lift. The idea behind the elliptical lift distribution is to minimize the creation of vortices at the wingtips, which are responsible for induced drag. These vortices create a downwash and by distributing the lift in an elliptical manner, the downwash is more evenly distributed along the span, reducing the strength and extent of the wingtip vortices. In an elliptical lift distribution the Oswald's efficiency factor is 1.

Fully elliptical wings are uncommon in practical applications, nevertheless, for a variety of reasons. The intricate manufacturing process and related expenses needed in obtaining and maintaining the perfect elliptical shape represent a significant challenge. With an elliptical wing shape it is very technically challenging to construct the leading and trailing edges as well as the spar. Also introducing Flaps, Slats, Aileron and other control surfaces to the wing is much more manufacturing challenging than in a

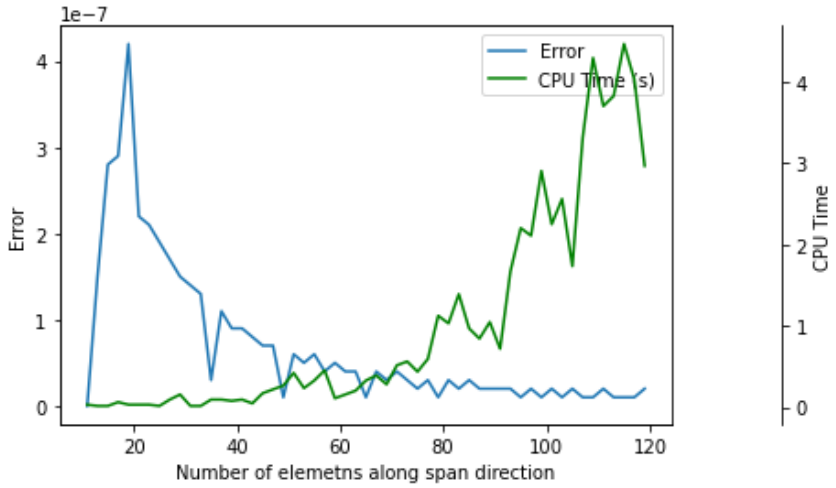



Figure 6: Number of mesh elements vs. Error & CPU time

conventional wing.

Another issue is how the elliptical wing behaves at the wingtips. The local Reynolds number reduces as the chord gets closer to the wingtips, making the wing more prone to stalling. The aircraft's controllability and maneuverability may suffer as a result, especially under difficult flight circumstances.

Consequently, a mix of twist and chord distribution is the preferred method in wing design. Without relying entirely on an elliptical wing shape, it is possible to achieve the appropriate lift distribution by carefully regulating the variations in chord and angle of attack through twist.

Although the elliptical wing is the ideal design in theory, it is rarely used in practice due to manufacturing difficulties and expense, as well as the risk of stalling at the wingtips. To achieve the necessary lift distribution and maximize aerodynamic performance, current wing design methodologies concentrate on combining twist and chord distribution.

**Mesh size study**

Before performing the optimization of the wing shape, involving changes to the angle of attack ($\alpha$), chord distribution ($c(y)$), and twist distribution ($\gamma(y)$), it is crucial to establish an appropriate mesh size to ensures not only efficient computational performance but also that the optimization results are independent of the mesh used. For this purpose, a relationship between the $x$ and $y$ directions has been defined as follows:

$$N_x = \frac{N_y}{AR} \tag{6}$$

Here, $AR$ represents the aspect ratio, while $N_x$ and $N_y$ denote the number of elements in each direction. This equation ensures a mesh with elements of relatively equal length and width. Subsequently, the drag coefficient of the wing was computed without undergoing the optimization process for meshes of multiple sizes, and the associated error and CPU time were evaluated to determine the impact of the mesh size. The results are presented in Figure 6.

As shown in Figure 6, it is evident that the error decreases as the number of mesh elements increases. However, it should be noted that the CPU time also increases as the mesh gets more refined.

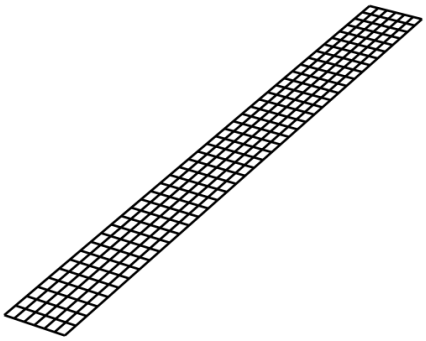


Figure 7: Mesh

After approximately 40 elements in the span direction, the error reduction becomes minimal, less that $10^{-7}$, while the CPU time increases significantly. Therefore, it can be inferred that a mesh with around 40 elements in the span direction is sufficient for accurate computation of the drag coefficient ($C_D$) while still being computationally efficient. The problems were set with a mesh size of 41 by 7 elements. The resulting mesh is presented in Figure 7. The influence of the number of control points in the chord and twist was also studied. It was determined that when varying this number the results were not significantly affected, with errors smaller than the $10^{-7}$, and so 3 control points were used for both twist and chord.

**Tabular results**

The analysis of different scenarios shows consistent constraints (S, CL and F values) and increasing complexity with more design variables included. Comparing the scenarios in terms of increasing number of design variables, it can be concluded that the computational effort increases as the number of design variables increases. In terms of CD minimization, its possible to say that all scenarios achieve very similar value.

| | Constrains | | Optimization Process | | Optimal Solution | | |
|---|---|---|---|---|---|---|---|
| | S | CL | Iter. | F. Eval. | $\alpha$ | CD | CL |
| (i) - $\alpha$ | - | 0.2253 | 3 | 3 | 0.47 | 0.00704915 | 0.2253 |
| (ii) - $\alpha$ and $\gamma(y)$ | - | 0.2253 | 8 | 8 | 0.53 | 0.00704692 | 0.2253 |
| (iii) - $\alpha$ and $c(c)$ | 16.2 | 0.2253 | 18 | 21 | 0.49 | 0.00706894 | 0.2253 |
| (iv) - $\alpha$ and $\gamma(y)$ and $c(y)$ | 16.2 | 0.2253 | 24 | 28 | 0.55 | 0.00706888 | 0.2253 |

Table 3: Optimal solution for the aerodynamic optimization using varied design variables.

**Optimization of $\alpha$**

The only design variable in the beginning of the optimization process is the angle of attack. In light of the fact that the lift coefficient depends on the angle of attack, it is important to note that by limiting the lift coefficient, we may choose from a smaller number of possibilities in terms of angle of attack. Instead of changing the geometry of the wing, this optimization changes the angle the wing makes with the airflow. The achieved lift distribution and the desired elliptical distribution are depicted in the accompanying image, Figure 8. It becomes clear that it was impossible to achieve the elliptical lift distribution by only optimizing the angle of attack.

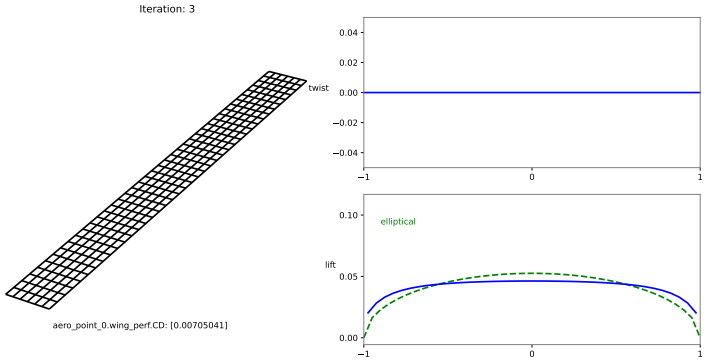



Figure 8: $\alpha$ optimal result

**Optimization of $\alpha$ and $\gamma(y)$**

The incidence angle of the wing can change along its span by adding torsion as a design variable using a three-point control array. At the root of the wing, torsion is deliberately set to zero. In Figure 9 the optimal torsion across the span is depicted. This value is negative and decreases as we move towards the wing tips. The value of the optimal $\alpha$ is 0.53 degrees . Contrary to the preceding situation, it's interesting to note that the lift distribution curve closely resembles the elliptical distribution curve. Considering that the lift distribution almost completely encloses the elliptical distribution curve, this alignment indicates that the wing's drag coefficient (CD) is minimized.

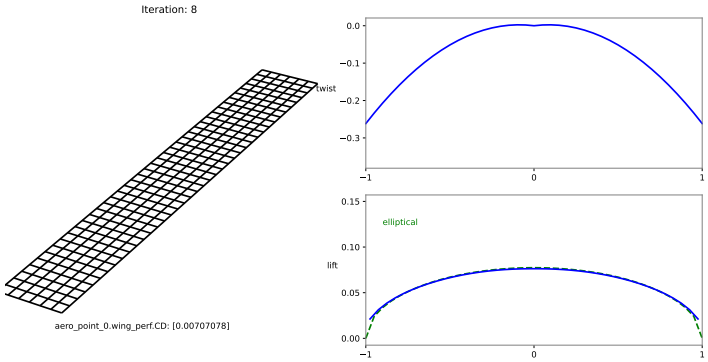



Figure 9: $\alpha$ and $\gamma(y)$ optimal result

**Optimization of $\alpha$ and $c(y)$**

In the optimization using the design variables $\alpha$ and $c(y)$, and keeping in mind what was predicted theoretically, the wing's shape altered from a rectangle to an ellipse. The twist value of the wing stayed constant at 0, and $\alpha$ equal to 0.49 degrees throughout the wing. The chord varies as depicted in Figure 10 with the span in order to obtain an elliptical distribution.

**Optimization of $\alpha$, $\gamma(y)$ and $c(y)$**

In this scenario, all the design variables were used. The wing shape gradually transformed into an elliptical form. The twist of the wing predominantly showed a positive value, increasing from the midpoint towards the tip. However, in the half portion of the span closer to the fuselage, the twist had a negative value, and it was zero at the root, as specified. The value of was $\alpha$ 0.55 degrees and the chord varied in span, in order to obtain an almost elliptical distribution of lift.

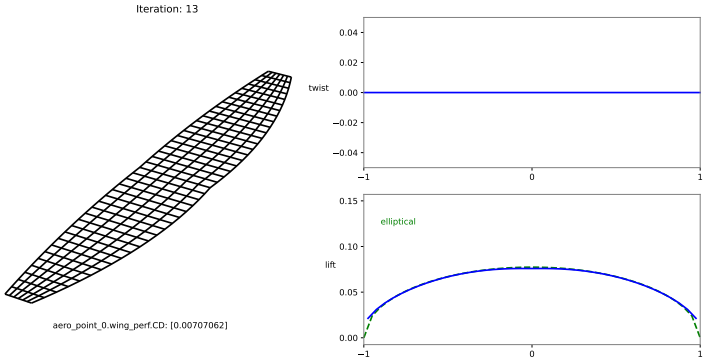



Figure 10: $\alpha$ and $c(y)$ optimal result

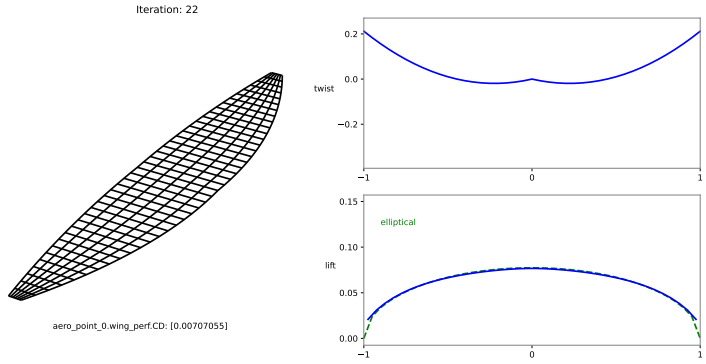



Figure 11: $\alpha$, $\gamma(y)$ and $c(y)$ optimal result

**New constrain to the optimization problem**

To incorporate a new constraint into the optimization problem, one variation of the Berguet Equation, which relates aircraft performance and aerodynamics was taken into consideration:

$$\text{Range} = \frac{CL}{CD}\frac{V}{SFC}\ln\left(\frac{Wf}{Wi}\right) \tag{7}$$

To introduce a constraint factor $\ln\left(\frac{Wf}{Wi}\right)$ was denote as $F$. By limiting $F$ to a lower bound, a new constraint in the optimization was added, and the range can be maximized by focusing on optimizing the wing shape to minimize the drag coefficient.

To properly imlement this, a new breguet_range.py script to implement the function was created and a new subsytem was added. Finally, all connections for variables were made.

**Tabular results of optimization with the new constrain**

Table 4 presents the results of the otimizations problems with the constrain in the previous section. This table has similar values to the previous one, with the addition of an extra column representing factor F. Factor F acts as a constraint, set to a lower bound of 0.8, for all the combinations of designing variables.

| | Constrains | | | Optimization Process | | Optimal Solution | | |
|---|---|---|---|---|---|---|---|---|
| | S | CL | F | Iter. | F. Eval. | $\alpha$ | CD | CL |
| (i) - $\alpha$ | - | 0.2253 | 0.8 | 3 | 3 | 0.47 | 0.00704915 | 0.2253 |
| (ii) - $\alpha$ and $\gamma(y)$ | - | 0.2253 | 0.8 | 8 | 8 | 0.53 | 0.00704692 | 0.2253 |
| (iii) - $\alpha$ and $c(c)$ | 16.2 | 0.2253 | 0.8 | 18 | 21 | 0.49 | 0.00706894 | 0.2253 |
| (iv) - $\alpha$ and $\gamma(y)$ and $c(y)$ | 16.2 | 0.2253 | 0.8 | 24 | 28 | 0.55 | 0.00706888 | 0.2253 |

Table 4: Optimal solution for the aerodynamic optimization using varied design variables and an extra constrain

From the results, we can say that the aerodynamic optimal solution is not greatly affected by the constrain but the shape of the wing, meaning the design variables, can change as it will be seen later.

**Optimization of $\alpha$ with the new constrain**

With the new constrain, in the design aspect, the wing shape remains constant along the span when considering only the angle of attack as a design variable, as depicted in Figure 12. This ensures that the aerodynamic performance, as mentioned earlier, remains consistent throughout.

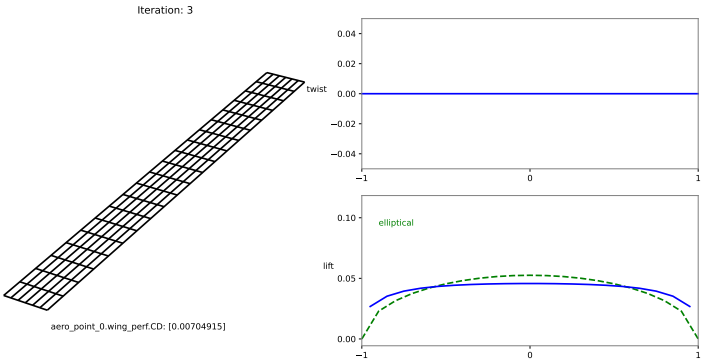



Figure 12: $\alpha$ optimal result with the new constrain

Similar conclusions to the ones in cases without the new constrain to can be drawn to the cases with the new constrain when the angle of attack and twist or the angle of attack and chord are the design variables. Including the constraint will not affect the aerodynamic coefficients nor the wing shape.

**Optimization of $\alpha$ and $\gamma(y)$ with the new constrain**

The wing shape resultant from the optimization considering the angle of the attack and the twist can be seen in Figure 13.

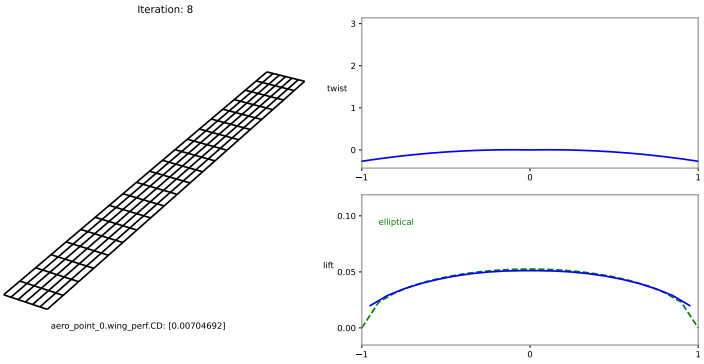



Figure 13: $\alpha$ and $\gamma(y)$ optimal result with the new constrain

**Optimization of $\alpha$ and $c(y)$ with the new constrain**

The wing shape resultant from the optimization considering the angle of the attack and the chord can be seen in Figure 14.

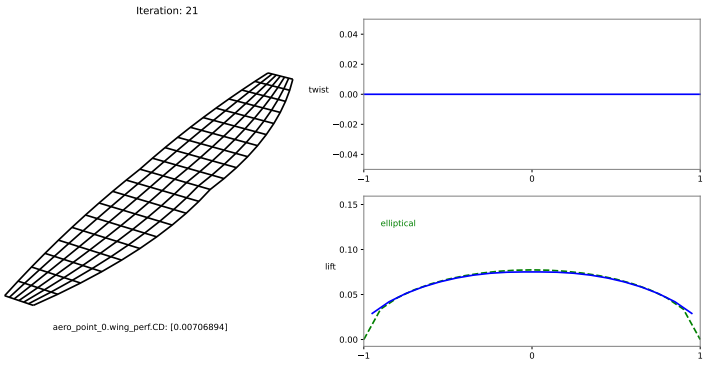



Figure 14: $\alpha$ and $c(y)$ optimal result with the new constrain

**Optimization of $\alpha$, $\gamma(y)$ and $c(y)$ with the new constrain**

To conclude the analysis considering all possible design variables, a significantly different wing shape was achieved, as shown in Figure 15, when incorporating the new constraint. It is important to emphasize the same aerodynamic performance as in the previous analysis was achived.

It is possible to observe a change in the twist distribution. The twist is zero near the fuselage and decreases as we move closer to the middle of the wing and then proceeds to increase towards the wingtips. Closer to the center of the fuselage a greater change in the value of twist is observed. This change is accompanied by a decrease in the chord. This is done to achieve a near perfect elliptic lift distribution on the wing. Near the fuselage, the chord is much smaller compared to the previous case without the constraint.

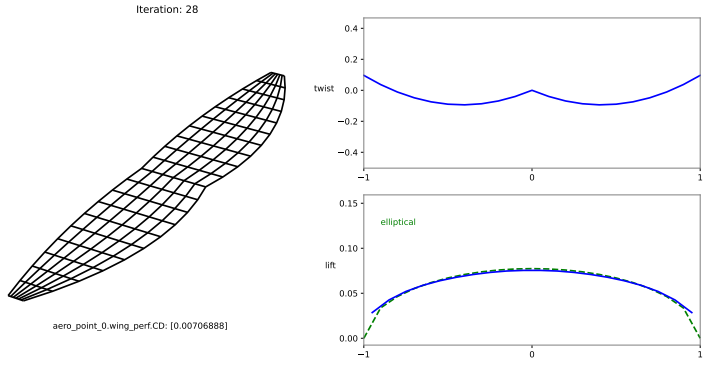



Figure 15: $\alpha$, $\gamma(y)$ and $c(y)$ optimal result with the new constrain

# 5 Conclusions

From analysing the optimization of the Rosenbrock function in OpenMDAO it was concluded that gradient computations, when using finite differences, can account for a great deal of function calls. If the function to be called is a very expensive function to compute the cost of calculating said gradients can

be prohibitive. So whenever advantageous analytical partial derivatives should be used to minimize the optimization computational power required.

In the Aerodynamic Optimization using OpenAeroStruct, primarily, our aim was to have a elliptical circulation distribution, as that is the theoretical optimal solution. In order to ascertain the most optimal aerodynamical approach, we undertook a study on the impact of mesh size, comparing the magnitude of errors incurred and the corresponding computational processing time as we change the mesh elements number. In terms of design variables, our intention was to discern the far-reaching consequences brought about by their manipulation. Moreover, we introduced a novel constraint, probing its influence upon the optimal solution. Furthermore the influence of this newfound constraint, was analyzing in the aerodynamic optimization with all the proposed design variables. The results of including the new constrain were the expected ones. Only in the case of the Optimization with design variables $\alpha$, $\gamma(y)$ and $c(y)$ with the new constrain a new wing shape was obtain, and in all design variables combination cases the the same aerodynamic performance was achieved when compared to the case without the new constrain.

The partial derivatives of the new constrain were analytically calculated and it was determined that using the Finite Differences method existing within OpenMDAO provided very small errors when compared to using the analytical derivatives.

Something interesting to conclude the work is to shown the wing in figure 16. Although it was possible to reach an optimal solution for a value of F greater than 0.8, the wing does not have a structure that can be built due to structural mechanical constraints.

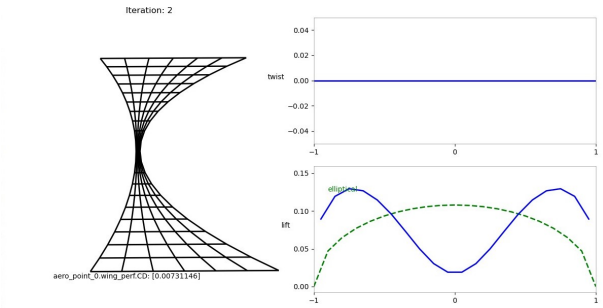



Figure 16: Impact of setting $F = 0.95$

# References


[1] T. O. D. Team. Approximating partial derivatives- openmdao. https://openmdao.org/newdocs/versions/latest/features/core_features/working_with_derivatives/approximating_partial_derivatives.html?highlight=approximations, 2022.

[2] T. O. D. Team. Scipyoptimizedriver- openmdao. https://openmdao.org/newdocs/versions/latest/features/building_blocks/drivers/scipy_optimize_driver.html, 2022.