Aerodynamic Shape Optimization of A Payload Constrained Fuselage Using MACH-Aero and IPOPT
AEROSP 588 - Multidisciplinary Design Optimization
Shashwat Patnaik (0289 1591)

Abstract:

The problem is formulated as a multi-objective optimization task, aiming to minimize the drag coefficient while subject to various constraints, including lift, surface area, angle of attack, thickness, volume, and leading/trailing-edge positions. The classification of the problem reveals its non-linear, constrained, and non-convex nature. The proposed optimization algorithm, IPOPT, is chosen for its efficacy in large-scale nonlinear optimization. The coupled model involves the integration of an optimizer with a Computational Fluid Dynamics (CFD) solver. To ensure uniqueness in the solution, multiple starting points are explored, initializing the geometry with different shapes. The solver of choice, ADflow, is employed for its robustness and efficiency in computing derivatives through a discrete adjoint method. Solver performance metrics indicate the reliability and computational efficiency of ADflow, while the derivative computation using adjoint methods mitigates numerical noise. The constraints, including thickness constraints for payload and surface constraints for weight, are implemented using DVGEO. A simplified problem with two design variables is presented to showcase the optimization framework. The results demonstrate the optimizer's performance, and an investigation of the optimum highlights the impact of design variables on the fuselage shape. The overall conclusions emphasize the significance of the proposed framework in achieving aerodynamic shape optimization for UAV fuselages, offering valuable insights for further research and development in the field.

Background and Motivation:

Unmanned aerial vehicles (UAV) are on the rise and companies face harsh restrictions such as UAS group [1] limitations which impose strict weight limits. UAV companies must design fuselage shapes that are able to enclose odd shapes, be lightweight and remain within regulations all while minimizing drag. Various constraints will be cleverly employed in order to meet the requirements and map physical quantities into the optimization framework. Most military drones are constructed in such a way that the fuselage is integrated into the wings, however non-military companies providing solutions for consumers might find a detachable fuselage desirable. They are detachable in order to allow for interchangeable payloads and fuel tank sizes depending on the mission requirements. Therefore the framework provided acts as a computationally affordable solution for determining the optimal shape given a set of geometric constraints.

The idea of being able to enclose any geometric shape extends beyond payloads and fuel tanks into instrumentation typically mounted in the nose of the aircraft. The end goal is to reduce the drag of the fuselage as to extend the range. An example of a commercially available drone is the G1 from Vayu Aerospace [2] which promises a range of 1200 miles with a standard payload, the range is dependent on the weight of the payload however if the shape of the transported object is

irregular, a standard shaped fuselage would need to grow tremendously, increasing it's surface area that is in direct correlation with the weight. See Figure 1 for a typical drone with uniform thickness, this type of aircraft is great for consumers with a specific payload size in mind. In order to tailor their product toward a wider range of buyers, having interchangeable and variable fuselage shapes would be of great benefit.

Figure 1: Example of a drone construction with a fuselage that has uniform thickness. [2]

Problem Formulation:

The following exploration is based on the Mach-Aero framework from MDOLab. A 2-dimensional geometry of an aircraft fuselage is supplied along with a set of constraints which could include: lift, surface area, angle of attack, thickness, volume, flatness and leading-edge or trailing-edge position. The objective is to minimize the drag of the fuselage while varying the shape of the geometry and each time the geometry is modified both a Computation Fluid Dynamics (CFD) solver is converged and then the optimizer is run. Numerous variations of these constraints will be applied, see the formulation below with all constraints being active; see Table 1 for nomenclature:

$$\begin{aligned} &\text{minimize} && C_D = f(x, c_l, a, AoA, M) \\ &\text{with respect to} && -0.1 \leq x \leq 0.2 \\ &\text{subject to} && 0.14 \leq C_L \leq 0.2 \\ & && SA \geq SA_0 \\ & && 0.1t_{i,0} \leq t_i \leq 3t_{i,0} \\ & && pt_{j,0} \leq t_j \leq 3t_{j,0} \end{aligned}$$

Table 1: Nomenclature of Mathematical Symbols and Their Name

Symbol	Name
x	Shape variables
C_D	Coefficient of drag

C_L	Coefficient of lift
L	Lift
D	Drag
a	Altitude
SA	Surface area
V	Volume
AoA	Angle of attack
M	Mach number
t_i	Thickness of fuselage without payload
t_j	Thickness of fuselage with payload
p	Height of the payload (scaled)

Problem Classification, and Description of Models and Coupling:

In order to classify the problem it is important to note how the aerodynamic shape optimization framework functions and only then is it possible to classify the problem. Looking at Figure 2 it is possible to see the six major modules used in MACH-Aero: Pre-processing, geometry parameterization, volume mesh deformation, flow simulation, adjoint computation and optimization.

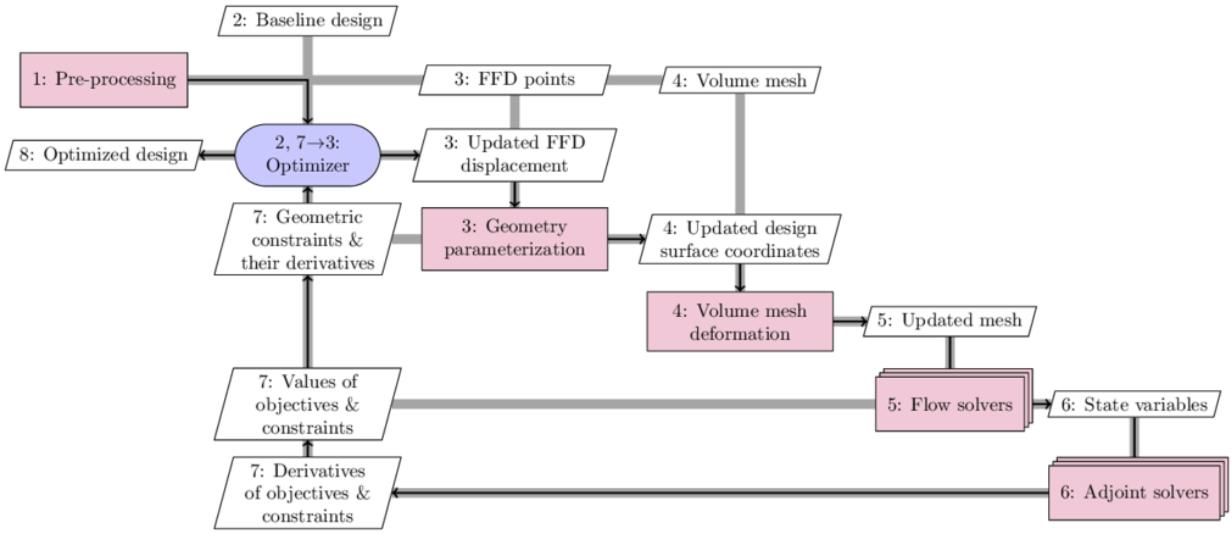

Figure 2: MACH-Aero framework and the interactions between the different components.

As opposed to the optimization problems with prescribed analytical functions, the MACH-Aero framework relies on computing certain quantities based on a baseline design. The baseline design has a mesh which is then deformed and a change in the output is observed. By virtue of the CFD solution only being truly converged at that iteration, any deformation will increase the residuals and thus the solution space only acts as an approximation. The optimization problem classification is as follows:

1. *Problem formulation:*
   Objective: There is a single objective function which is the drag coefficient
   Design Variables: The design variables are the node-positions of the Free Form Deformation (FFD) volume which by themselves are continuous in the sense that they can move smoothly. However the effect of a changing design variable must result in the re-evaluation of the CFD domain which cannot be classified as continuous; it is inherently discrete. Along with the constraints the design variables are said to be mixed.
   Constraints: Inequality constraints are present which means that the problem is constrained. In MACH-Aero using IPOPT equality constraints may be represented as inequality constraints with a tolerance of zero and typically has no effect on potentially encountering infeasible solutions.

2. *Objective and constraint function characteristics:*
   Smoothness: The iterative nature of the framework suggests that the problem is discontinuous, even though smooth continuous variables do exist which might smoothly change the drag coefficient if the solver was replaced with an approximation. This is not the case.
   Linearity: The problem is highly non-linear.

   Modality: Aerodynamic shape optimization problems with clearly defined objectives and constraints, will have a single unique optimal solution and hence it is unimodal.
   Convexity: The problem is not convex.
   Stochasticity: Although seemingly chaotic, the nature of this problem is deterministic.

Proposed Optimization Algorithm:

The optimization algorithm used is called IPOPT which works great for large-scale nonlinear optimization and relies on minimizing an objective function with constraints on the form:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g^L \leq g(x) \leq g^U \\ & x^L \leq x \leq x^U, \end{aligned}$$

The optimizer makes use of an interior point line search filter and adaptive barrier update strategies as explored by Jorge Nocedal [4]. He proposed a new adaptive strategy which outperformed monotone strategies. This will be the chosen foundation of the MACH-Aero framework. This algorithm has a vast array of settings that can be modified, the most important ones are highlighted in the listening below:

Listing 1: IPOPT Arguments

```
optOptions = {
    "max_iter": 10000,
    "tol": 1.0e-6,
    "hessian_approximation": "limited-memory",
    "limited_memory_max_history": 100,
    "output_file": os.path.join(args.output, "IPOPT_print.out"),
}
```

The limited-memory setting of the Hessian approximation is used to reset this sparse matrix at times which greatly speeds up the computation time. The tolerance is set to $1.0e-6$ which is set to terminate when the scaled Non-Linear Program (NLP) error becomes smaller than that value. The objective function of this problem is scaled to a factor of $1.0e4$ which is again done to make it more "well-behaved.

Investigation of Model:

A coupled model typically involves coupling two disciplines such as aerodynamics and structures; this is done in OpenAeroStruct for instance. This project makes use of an optimizer coupled to a CFD solver

Modality:

Typically aerodynamic shape optimization problems with clearly defined objectives and well-constructed objectives will have a unique solution thus it is a unimodal. However, due to the non-linearity of the model, the problem is not convex.

As it is a nonlinear problem, it is not possible to mathematically prove it has a unique solution. Thus, to ensure that it has a unique solution multiple starting points are optimized to ensure that they all converge to the same solution. To verify our problem, the geometry is initialized to an ellipse and 'pill' shaped fuselage, as shown in the figure below.

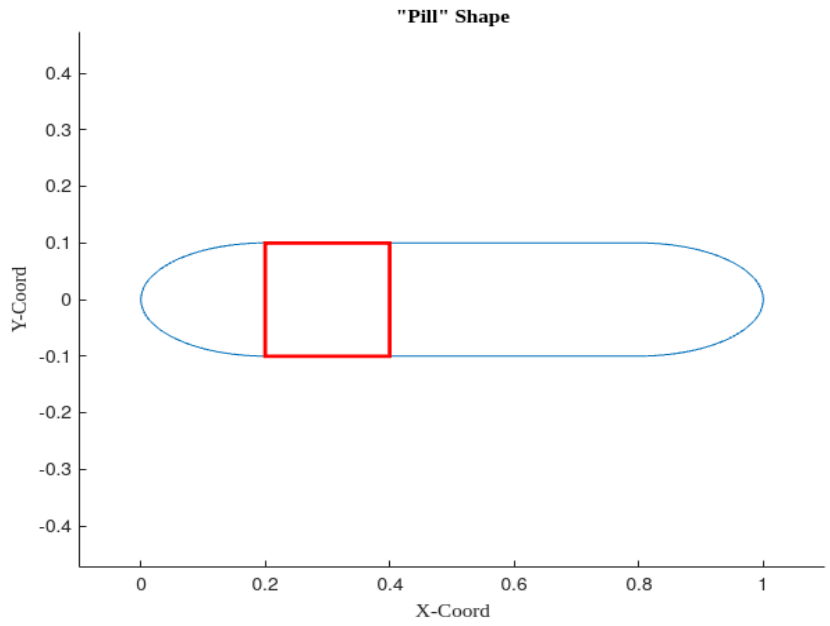

Figure 3: Initial geometry of the fuselage called “pill” shown in blue and the payload as a box with side length 0.2.

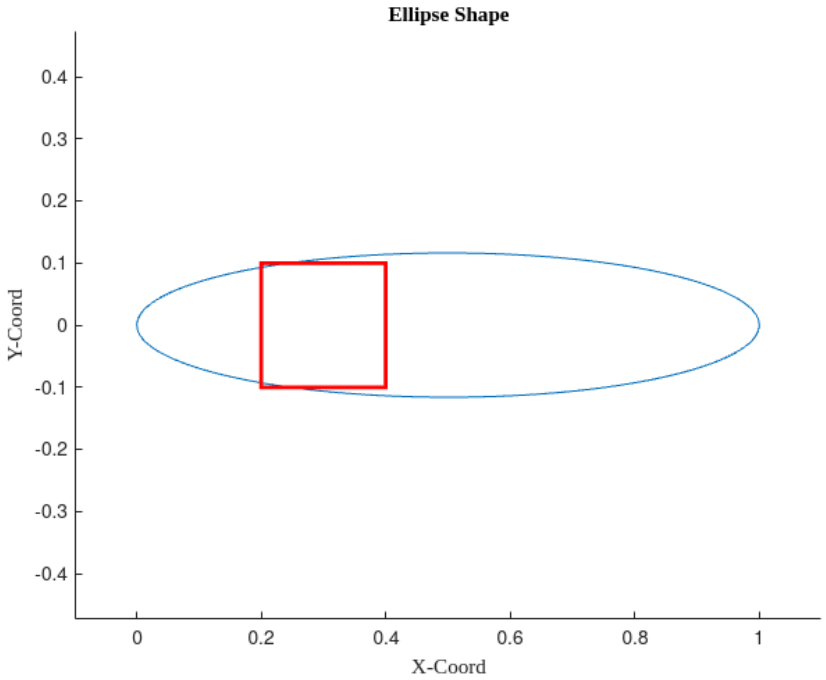

Figure 4: Initial geometry of the ellipse shaped fuselage shown in blue and the payload as a box with side length 0.2.

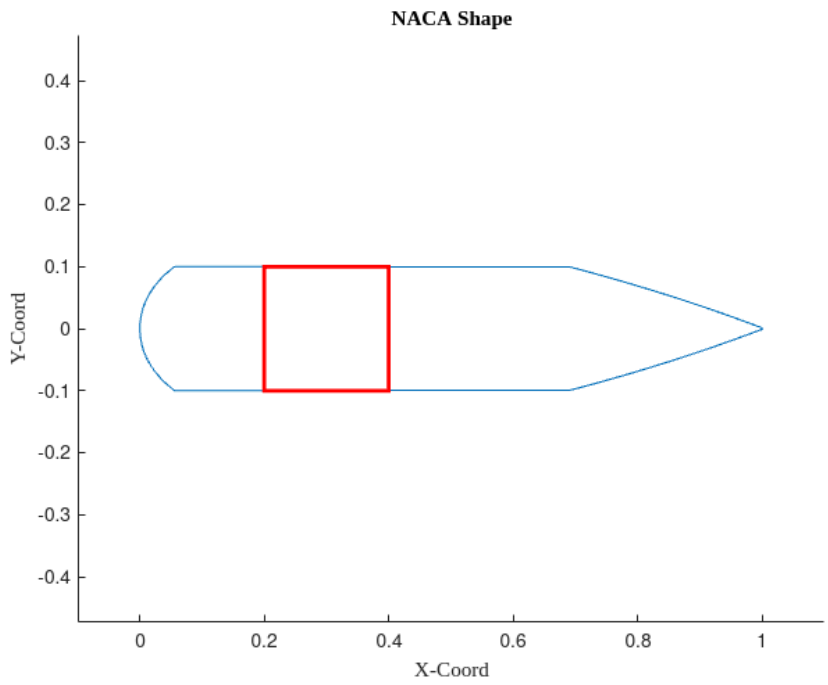

Figure 5: Initial geometry of the ellipse shaped fuselage shown in blue and the payload as a box with side length 0.2.

Choice of Solver:

The Mach-Aero framework provides support for ADflow and DAfoam. Other solver implementation is not possible due to certain quantities not being available for surface constraint like triangulated surface and ability to compute adjoints, which is required to compute total derivative. However, DAfoam is almost five times slower computationally to compute the preconditioner [5].

ADflow was chosen as it is a robust solver and has a discrete adjoint method that efficiently computes the derivatives of objective and constraint functions with respect to large numbers of design variables. With the added advantage, the implementation has already been established in ADflow. It has been coupled with an approximate Newton–Krylov (AKN) and Newton–Krylov (NK) solver for efficient and accurate convergence. There are two main steps when using this method. In the first step, we use the ANK solver to significantly decrease the total residual norm by five orders of magnitude compared to the initial iteration, which starts with free-stream conditions. Once this initial reduction is achieved, we switch to the NK solver for the final step of convergence.

Solver Performance:

To model Navier-Stokes, we use a RANS model. RANS simulations have demonstrated reliable predictions of airfoil performance under cruise conditions with minimal or no separation [6]. If there is a large separation in the flow field then RNAS performs poorly. However, as we are minimizing drag, the optimizer converges towards a smooth shape with minimal or no separation, thus the RANS method is valid for our case.

Additionally, ADflow has a good computational time while minimizing the utilization of memory. For a case with 450,000 cells, running it till L2convergence reaches 1e-8 with NK solver took 1200 sec. For a larger case with 5.2 million cells with parallelization it took 1200 sec. The computational time is comparable or lower than other solvers. On our machine with parallelization, for each initial geometry it took about 28 seconds on average, for it to converge to a residual of 1e-12.

Derivative Computation:

As the aerodynamic shape optimization has a considerably large number of design variables, the choice of derivative computation should not scale with respect to the number of design variables. Thus, the adjoint method is used to compute the total derivatives as it is independent of the number of variables. Adjoints are computed through ADflow using a discrete adjoint method, then the preconditioned GMRES solver from PETSc is used for solving the adjoint equations.

Numerical Noise:

One of the sources of error is the computation of total derivatives as the exact derivatives are not available. It could be reduced by setting a low tolerance for convergence really low. We have set the convergence for the CFD solver to L2 norm of 1e-12 and GMRES solver to 1e-13. It will not completely eliminate the noise but not completely eliminate it. In parametric optimization, round-off errors become critical as small perturbation can have a large impact on the flow; it could have occurred while reading FFD or mesh file, which can drastically affect performance and adds noise. Another is truncation error when computing derivatives, fortunately when comparing the total derivatives computed through GMRES adjoint method and complex-step method it was identical to machine precision [5]. Moreover, to reduce numerical noise from discretization error from mesh, a grid refinement study was performed to reduce any computation noise from the CFD solver further.

<u>Constraint Implementation Using DVGeo:</u>

Thickness Constraint:

Thickness constraints create a matrix of points along chord and span-wise direction, where the thickness is computed, the user can set a lower and upper bound on individual points. By default, thickness constraints act as scaled constraints where the initial value will be represented as 1. However, if scaled=False, then the user can provide physical length as an upper and lower bound.

For a 2D fuselage, the payload location is fixed in the aircraft, thus through thickness constraint We can represent the geometry of the payload in the fuselage by increasing the lower limit of thickness to the size of the fuselage, so it encapsulates the payload. The figure below represents the bounds the fuselage can move:

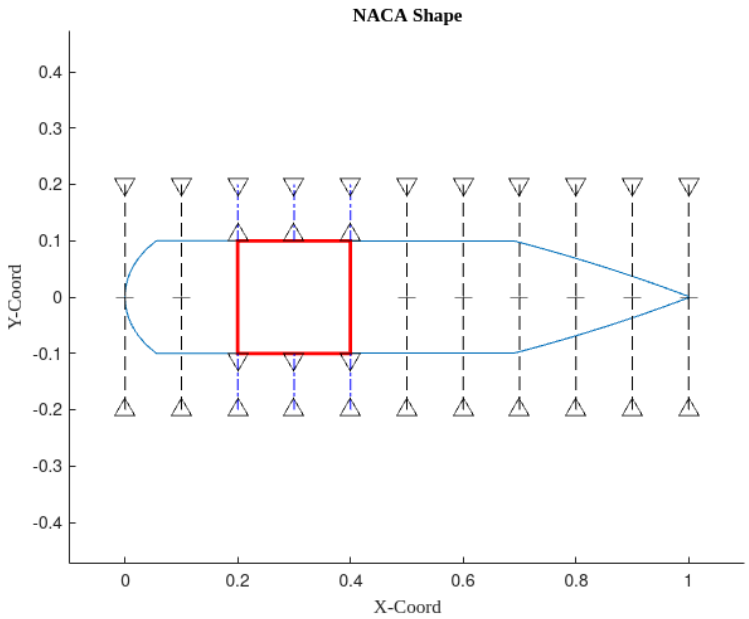

Figure 6: Visualization of the toothpick constraints and how they conform around the payload.

For flat N0012 airfoil, as initial geometry as the thickness where the payload is located is constant along the chordwise direction, the scaled method is implemented. However, for ellipse and 'pill' shapes the initial geometry bounds are given as physical length. It should not make any difference how the bounds are set up as both represent the same physical constraints. See Listing 2.

Listing 2: Thickness constraint defined as a scaling of the initial geometry file.

```
plthick = np.ones((2,100))*0.1
plthick[:,20:40] = 2
DVCon.addThicknessConstraints2D(leList, teList, 2, npl, lower=plthick, upper=2)
```

Surface Area (Mass) Constraint:

The fuselage is represented as a 2-Dimensional shape and hence a method of quantifying the mass of the optimized shape is necessary. One way of doing this is to use the surface area as a constraint and with regards to the geometry it is the circumference of the shape as the depth is unity in z-direction. Using data from MatWEB, a material science property data sheet, carbon fiber infused with epoxy resin has a density of about 1500 kg/m³ and assuming a typical layer thickness of 1mm means that a square meter of carbon fiber laminate equates to about 1.5kg. Hence one is able to initialize the circumference of the starting shape to a known value, for example 2 meters and by virtue of the 2D simulation would also equate to a surface area of 2 meters squared; also known as 3kg. Therefore in the setup of the optimization problem, a upper and lower bound can be specified; see the Listing 3 below:

Listing 3: Syntax for adding a surface area constraint.

```
addSurfaceAreaConstraint(lower=0.7, upper=0.9, scaled=True, scale=1.0, name=None, addToPyOpt=True,
    surfaceName='default', DVGeoName='default', compNames=None)
```

In this example, the lower bound is specified to be 0.7 and the upper bound is 0.9. This type of constraint will force the final surface area to be less than that of the initial shape. In this case one will expect the converged solution to have a surface area of maximum 1.8 meters squared. Again using the mapping of area to mass in the paragraph above, this will equate to a mass of 2.7 kg. This can be precomputed ahead of time or a user-specified function is written in the script to take care of this conversion automatically. This means that if one wishes to construct a symmetrical fuselage then an axis of revolution method could be used to map it into the true surface area; see the Equation below:

$$S = \int_a^b 2\pi y \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx$$

Flatness Constraint:

With respect to the motivation of this project, having an interchangeable fuselage is desirable and in order to achieve this, the finished shape needs to be realistic in terms of fabrication and mounting. One approach is to include a segment of the fuselage that is constrained to remain flat for proper mounting of the wings or landing gear. This is achieved by using the pointSelect function that selects a subset of control points where their displacement can be enforced locally. An example of this is manipulating the coordinate file of the initial shape and prescribing it to be a flat surface, then using bounding boxes to enclose certain control points of the Free Form Deformation (FFD) box and setting their upper and lower bounds to zero while specifying the other control points to move freely. See Figure 7 for how an FFD box may look.

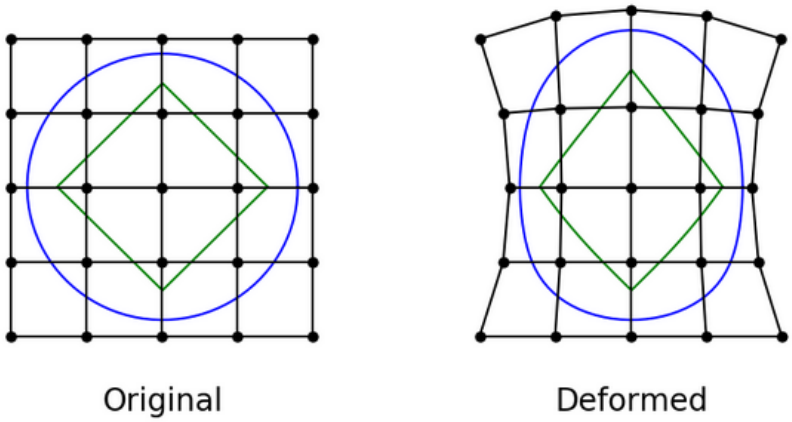

Figure 7: Free Form Deformation (FFD) box marked in black and the shape of the surface inside this box marked in blue and green.

Now using a NACA0012 airfoil for demonstration purposes, a flat region is imposed on the upper surface and its deformation is set to zero; the upper and lower bounds are equal. Then we may view the deformations to ensure that the problem is set up correctly; see Figure 8 below.

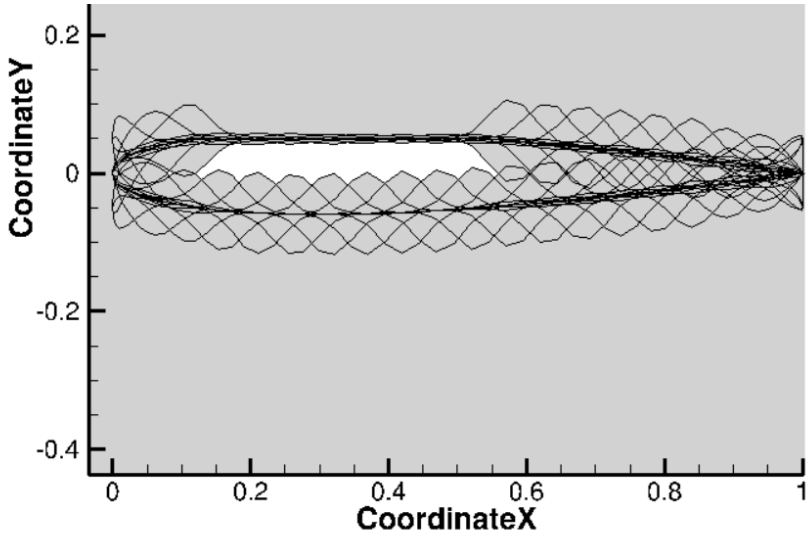

Figure 8: Superposition of FFD node deformations and the effect on the aerodynamic shape with the flat region at the top constrained not to move.

The issue however, is that although the nodes above the surface at the top are set to zero displacement, there is still movement. This can be explained by Figure 7 again, where moving only one control point has an effect on the entire surface shape of the part inside. Looking at this phenomenon in detail is done in Figure 9 where it is clear that the displacement of the bottom node has an effect on the top surface which aims to be completely flat.

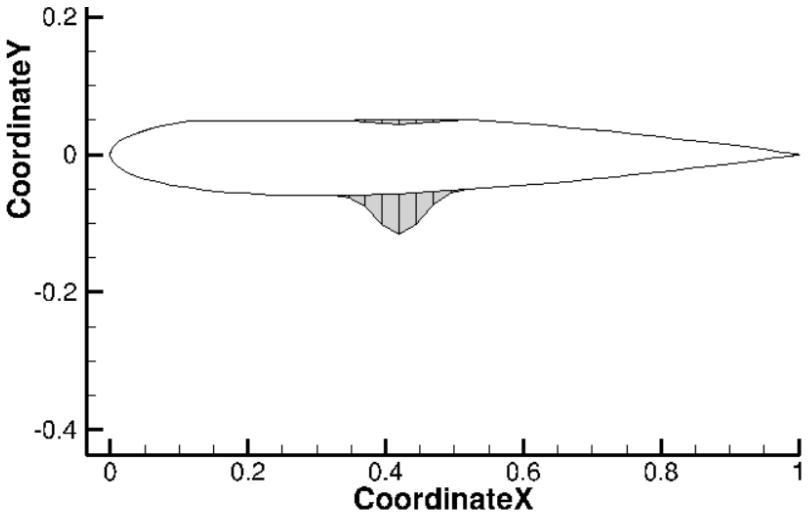

Figure 9: Node displacement of a control point on the bottom of the FFD having an effect on the displacement of the surface not only on the bottom but at the top as well.

This means that although the constraint is enforced, the ideal result will not be achieved and it might hinder the optimizer from converging quickly.

Linear Constraint:

The FFD box representing the control points has to be 3-D shape, however our fuselage is represented in 2-D. Thus, to overcome it we mirror the shape on the other side with a chord length of 1. Note, as this becomes a 3-D problem the control point will double, but there are no control points in between the two edges as shown in Figure 10.

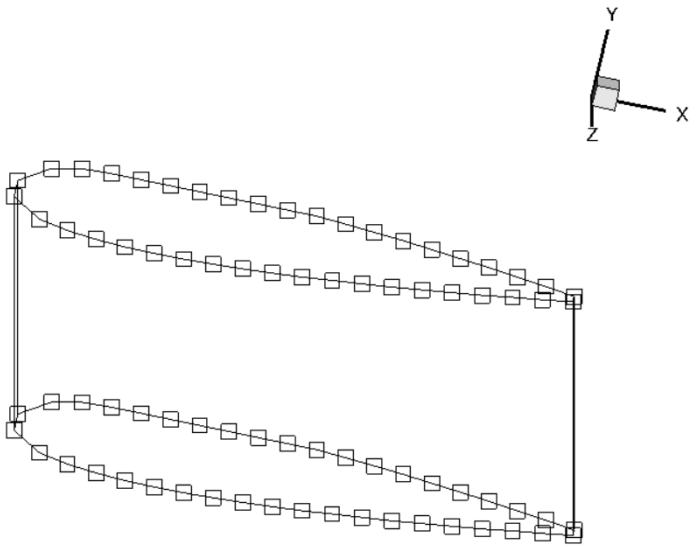

Figure 10: Parametrization of FFD volume of a NACA0012 airfoil

To make sure the shape of the fuselage remains the same, linear constraints are introduced to ensure that the shape deformations on one side of the airfoil mirror that of the other. The global index of the control point on one side is the input parameter as a vector and the other side as the other input. Then factorA is set to 1 and factorB is set to -1, so it mirrors the exact deviation of the control point on both sides. See Listing 5.

$$factorA.dvA - factorB.dvB = 0$$

Listing 4: Implementation of linear constraint which ensures both sides of the FFD volume move equal and opposite directions.

```
for i in range(lIndex.shape[0]):
    indSetA.append(lIndex[i, 0, 0])
    indSetB.append(lIndex[i, 0, 1])
for i in range(lIndex.shape[0]):
    indSetA.append(lIndex[i, 1, 0])
    indSetB.append(lIndex[i, 1, 1])
DVCon.addLinearConstraintsShape(indSetA, indSetB, factorA=1.0, factorB=-1.0, lower=0, upper=0)
```

Optimization of a Simplified Problem:

The goal is to construct a simplified problem that includes two design variables, to verify the MACH-Framework set-up. It is not possible to have only two variables if parameterization of the shape of the fuselage is used as a design variable. Thus, in this case, an aerodynamic optimization problem will be solved using a NACA0012 airfoil and varying the angle of attack and altitude.

$$\begin{aligned} &\text{minimize} && C_D = f(C_L, a, AoA, M) \\ &\text{with respect to} && a,\ AoA \\ &\text{subject to} && C_L = 0.5 \end{aligned}$$

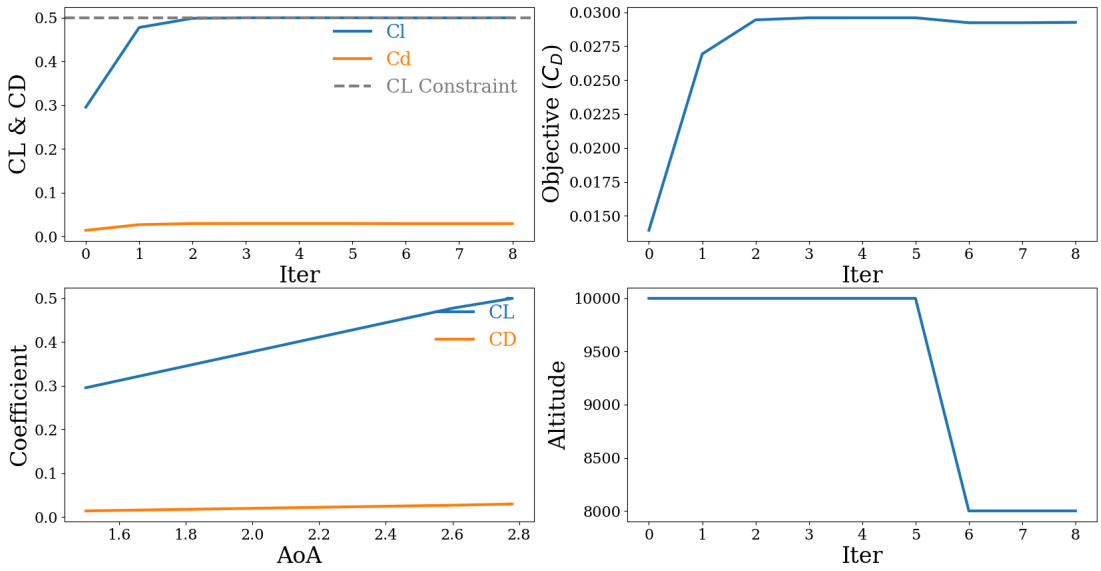

Figure 11: Convergence of MACH-Aero framework problem showing an optimal altitude of 8000 meters and the drag coefficient is 0.0292527

With respect to Figure 11, the results of the optimization problem become clear. Since both angle of attack and altitude are the design variables, one would expect for the altitude to reach the lowest value where density is the highest. An improvement would be to also include the Mach number as a design variable to see the comparison between flight velocity and altitude, however at the given Mach number it appears as if the lowest altitude is optimal. In conclusion, the optimizer has found the angle of attack where the drag is minimized at a lift coefficient of **0.5** and it appears at an angle of **2.724774** degrees.

Table 2: Optimizer Performance of Simplified Problem

Obj. Func. Calls	**9**	*Eq. Con. Jac. Evals.*	**9**
Obj. Grad. Evals.	**9**	*CPU time IPOPT*	**23.742 s**
Eq. Constraint Evals.	**9**	*CPU time NLP Evals*	**199.779 s**

Table 3: Adjoint Solver Performance of Simplified Problem

	Adjoint Solve Time	*Total Sensitivity Time*
C_D	**6.030 s**	**0.081 s**
C_L	**12.224 s**	**0.080 s**

The aerodynamic shape optimization framework was verified by using the tutorial present in the MACH-Aero Framework. The optimizer was repeated for different initial values of altitude and angle-of-attack and it converged toward the same optimum; hence it is an unimodal problem and the robustness of the framework is verified.

Results of the Main Problem Optimization:

Running the optimization framework on the 'pill' shape first, with a square payload of size 0.2 located at x-axis [0.2, 0.4] bound and was symmetric to the x-axis. The FFD was made up of 10 points for each edge thus in total 40 control points. Thus the bounds were set to:

- Deviation of control point - [-0.1, 0.2]
- Coefficient of Lift - [0.1, 0.2]
- Surface Area - [0.3, 1] (all the shape was initialized to have the maximum surface area or weight)
- Thickness Constraint - t(19, 40) = [0.2, 0.4] (there were 100 points in the chord direction)

The following the convergence history of optimization (note: gray lines are Cl bounds):

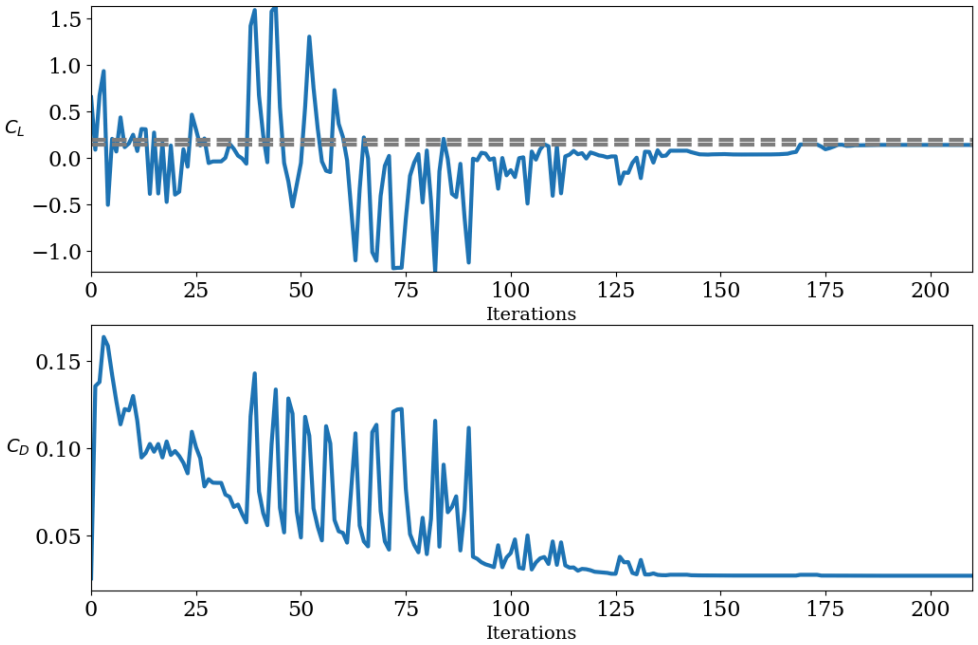

Figure 12: Convergence history with 'pill' shape as initial geometry

The following are the converged shape of the fuselage after 123 iterations:

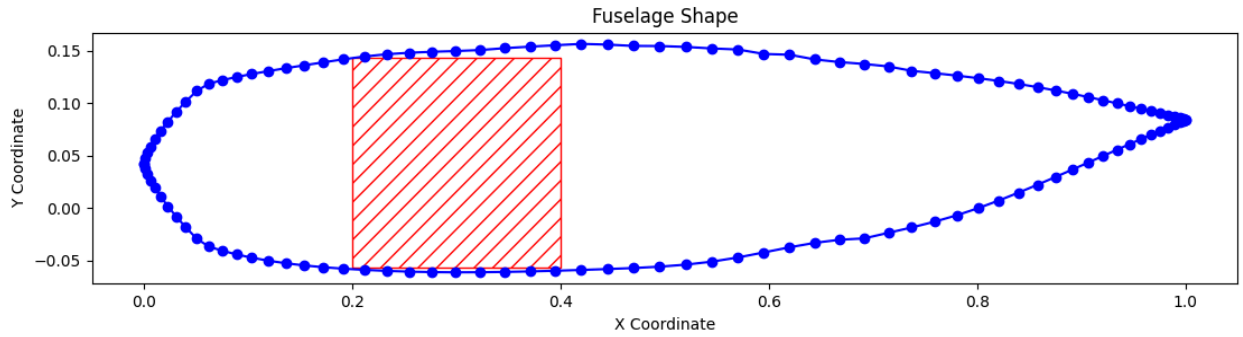

Figure 13: The shape of the fuselage after MACH-Aero optimization framework with the implementation of all constraints

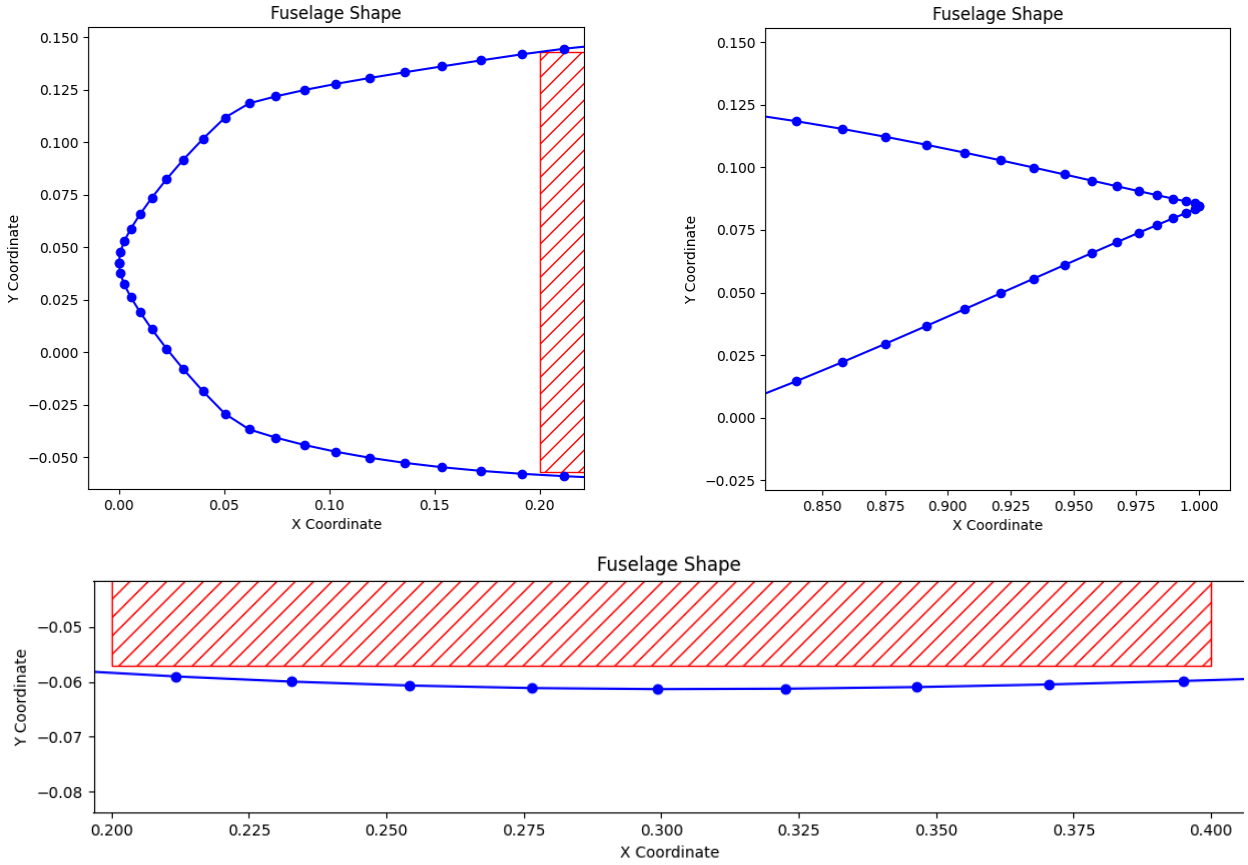

Figure 14: Zoomed in view of final shape after the aerodynamic shape optimization revealing that the payload is sufficiently embedded into the fuselage.

Interpretation of results:

The shape is in line with what was expected, the nose is round which becomes clear looking at the zoomed-in view in Figure 13. The payload is sufficiently embedded inside the shape and the trailing edge contracts to a point that suggests that the drag has been minimized. There is also a prescribed lift coefficient which is always desirable for a fuselage and in this instance, it has

achieved a good balance between the two considering the thickness of the shape. We expected the tail end to be much higher and the nose section to get wider later on. However, this might have happened due to surface constraint as it is exactly 1 at the end of convergence which was the limit.

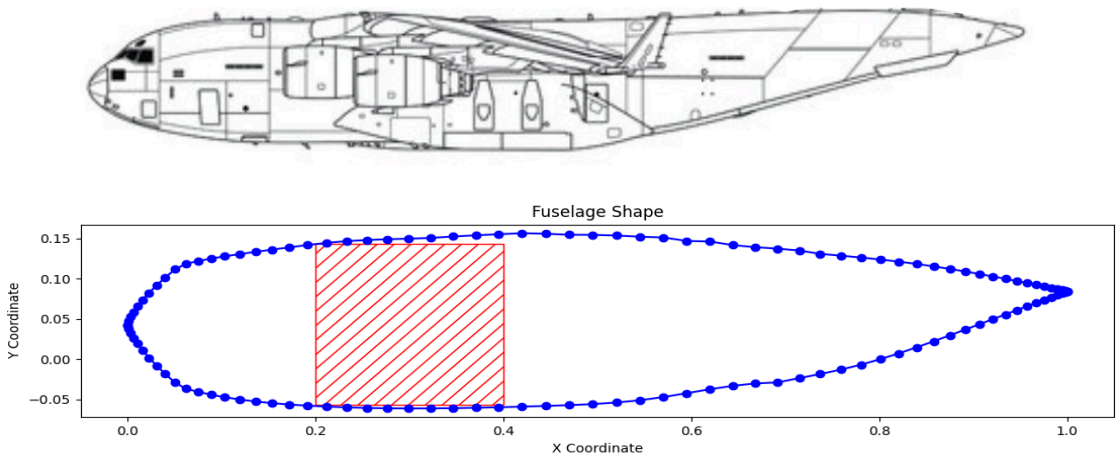

Figure 15: Fuselage of C-17 side-view vs Optimized Fuselage

Ideally it is better to have a slender fuselage with a large length, however, due to payload constraint and length constraints, the optimized aircraft is stout. In the real world, C-17 has similar requirements of a shorter length but is still able to carry a large payload. As shown in Figure 15, our fuselage is quite similar to C-17's fuselage. As mentioned earlier the tail section being lower could be because of surface constraint. Hence, it can be assumed that the optimizer worked as intended.

Investigation of Optimum:

The optimizer converged with values of C_L and C_D visible in Table 4.

Table 4: Solution values of lift and drag coefficients given different initial starting shapes

Starting Point:	*Pill*	*Ellipse*	*NACA*
C_D	**0.0399129**	**0.0399131**	**0.0399132**
C_L	**0.140005**	**0.140009**	**0.14006**

These results bolster the notion that there exists only one optimum for the described problem. We may take this one step further and visualize all three converged fuselage shapes in the same plot and ensure that they are in correlation with each other. See Figure 16 for this comparison, what is interesting is that it appears as if the leading edge has shifted slightly between the converged optima, however, so has the trailing edge. This was not expected but it is reassuring to see that the overall shape appears to be identical in conjunction with the values in Table 4.

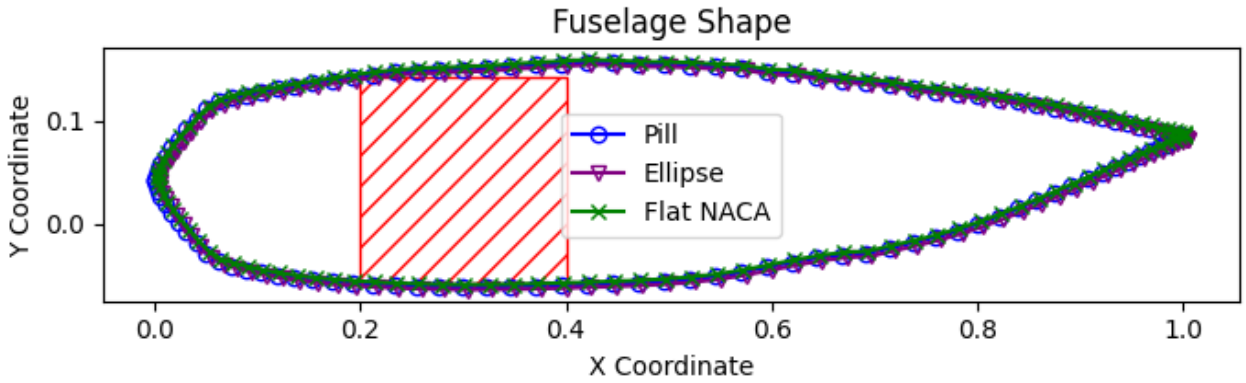

Figure 16: Investigation of Optimum Using Different Initial Guesses

Optimizer Performance:

It is worth noting that there was a difference in the amount of time it took to arrive at the optimal solution given different starting values. This is seen in Table 5 for the case of the "pill" shape.

Table 5: Number of iterations required to arrive at optimum for different starting shapes

Configuration	*Number of Iterations*	*Total CPU Time*
"Pill"	**210**	**4466 s**
Ellipse	**234**	**4710 s**
Flattened NACA	**240**	**5802 s**

Looking at these results, the initial geometry of the pill shape in Figure 3 performed best, both in terms of number of iterations and also total CPU time. The results of the ellipse, as well as the flattened NACA shape, are also reasonable. What is interesting is that the final converged optimal fuselage has a sharp trailing edge and the only initial geometry with this property is the flattened NACA. Although it is true that a sharp trailing edge is desirable, it is also the only initial geometry with sharp corners from the nose as well as where the flattened section blends into the trailing edge. What is not evident upon first inspection is whether the increased computational complexity and time is by virtue of the CFD solver or the optimizer.

Upon further analysis, it becomes apparent that the majority of the time is spent in the CFD solver. This is also strengthened by the fact that the difference between the ellipse and flattened NACA is only **7** in terms of iterations, however in CPU time it has increased significantly.

Also, the number of control points highly affects not only the computational time but also determines if the optimizer will converge. If the number of control points is 20, then the

optimizer will fail to converge, due to the error of bad-quality cells in the CFD solver. It could be happening as adding more control leads the optimizer to try weird shapes as shown in figure 17, which leads to pyHyp failing to mesh correctly. However, having too little control point will not allow the optimizer to generate a smooth shape, even though it may converge. We were on the limit on the minimum number of control points required as it can seen the tail end is not perfectly smooth.

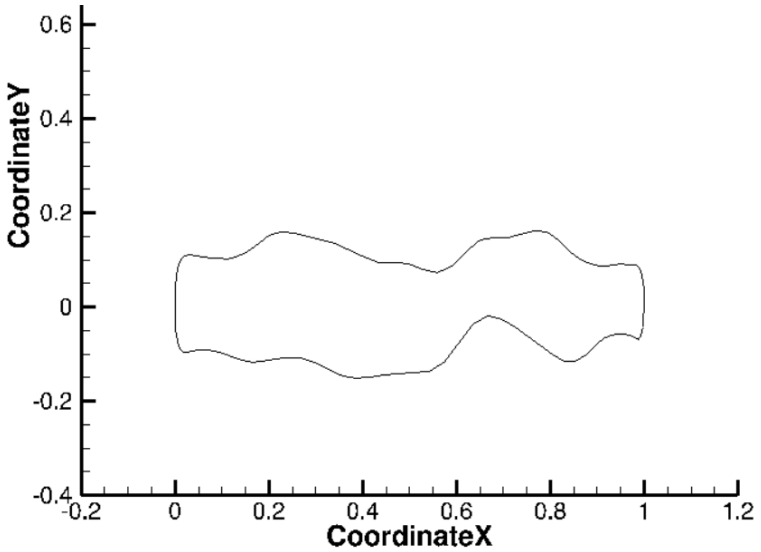

Figure 17: Fuselage after 20 iterations with 20 control points with starting geometry as ellipse

Conclusions, Discussion, and Recommendations:

As one can see, the “pill” shape converged in the least amount of time which is interesting. One may argue that the overall shape is aerodynamic to begin with however it does not have a sharp trailing edge as opposed to the flattened NACA0012 shape. One thing is clear, and that is that the time it takes to converge is highly dependent on the bounds given to the design variables, in this case the upper and lower bounds of the FFD volume control points. For example, the ellipse is thick which means that the points have to move a further distance to find an optima that was hypothesized. In order to keep the investigation fair, these bounds were kept constant throughout the different cases and therefore this forces the nodes to displace more than they have two. If a good initial guess is provided with tighter bounds it is likely that this configuration would converge in the least time.

Incorporating constraints such as surface area along with volume and thickness constraints was proven to be a cumbersome task. The geometry had to be prescribed in such a way that the initial surface area was known and that it would not result in infeasible solutions when the size of the payload was increased. For instance, at some point the surface area constraint was too tight and hence there is no geometric way of creating a shape from the leading to trailing edge with a

length of one enclosing a payload. In order to increase the robustness of the solver a recommendation for future work is to include a function which computes the shortest distance possible in order to enclose the payload and meet the bare minimum leading and trailing edge constraint. Since this report was an exploration, this phenomena was easily worked around after realizing that the optimizer yielded an infeasible solution, however if one wishes to commercialize this approach as described in the background and motivation, it needs to be far more robust and user friendly.

References:

[1] Reist, T. A., & Zingg, D. W. (2017). High-Fidelity Aerodynamic Shape Optimization of a Lifting-Fuselage Concept for Regional Aircraft. Journal of Aircraft, 54(3), 1085–1097. https://doi.org/10.2514/1.c033798.

[2] G-1 MKII Autonomous Aircraft Provides Wide Ranging Functions, Vayu Aerospace Corporation 2023. https://vayuaerospace.com/g-1-mkii.

[3] J. Nocedal, A. Wächter, and R.A. Waltz. Adaptive barrier strategies for nonlinear interior methods. SIAM Journal on Optimization, 19(4):1674–1693, 2008. preprint at http://www.optimization-online.org/DB_HTML/2005/03/1089.html.

[4] Nocedal, J., Wächter, A., & Waltz, R. A. (2009). Adaptive Barrier Update Strategies for Nonlinear Interior Methods. In SIAM Journal on Optimization (Vol. 19, Issue 4, pp. 1674–1693). Society for Industrial & Applied Mathematics (SIAM). https://doi.org/10.1137/060649513.

[5] Kenway, Gaetan K.W., et al. “Effective Adjoint Approaches for Computational Fluid Dynamics.” Progress in Aerospace Sciences, vol. 110, Oct. 2019, p. 100542

[6] He, Xiaolong . “Robust Aerodynamic Shape Optimization—from a Circle to an Airfoil.” Aerospace Science and Technology, vol. 87, 1 Apr. 2019, pp. 48–61