MAE 3303 Compressible Flow

Fall 2024

Design Project Report

Design Optimization of a Supersonic Airfoil

Submitted to

Dr. Zhen X. Han, Professor

by

Zayd Salcedo
E-mail: zgs9738@mavs.uta.edu
in Partial Fulfillment of Course Requirements

I pledge, on my honor, to uphold UT Arlington's tradition of academic integrity, a tradition that values hard work and honest effort in the pursuit of academic excellence. This Lab Report contains only work that I have personally created or that I have appropriately referenced as work from other sources. I followed the highest standards of integrity and upheld the spirit of the Honor Code.

Summary

A hypothetical design of a hypersonic biconvex airfoil at Mach 3 was designed using MATLAB functionality. The geometry was created using a continuous cubic piecewise spline, and observations were recorded on how parameters changed the shape of the element. Then, the element was digitized and transformed into a finite series of flat panels and shock-expansion theory was used to calculate the aerodynamic coefficients of the airfoil. Finally, a brute force method was developed to optimize the Lift-Drag Ratio of the element within specific design constraints. An optimal L/D Ratio was achieved to be 3.3109 while minimized computational time.

Table of Contents

Summary	i
Table of Contents	ii
List of Symbols	iii
List of Figures	iv
List of Tables	v
Design Objectives and Constraints	1
Geometrical Setup	1
Airfoil Analysis Process	5
Optimal Design Process	10
Results	15
Appendix	18

List of Symbols

Greek Symbols:

α	Angle of Attack (AOA)
β	Shock Wave Angle
γ	Specific Heat Ratio of Air
θ	Deflection Angle

Roman Symbols:

nsl	Number of Lower Surfaces
nsu	Number of Upper Surfaces
tl	Lower Thickness
tu	Upper Thickness
x	X Position
xtl	X Position of Maximum Lower Thickness
xtu	X Position of Maximum Upper Thickness
yl	Lower Y Position
yu	Upper Y Position
AOA	Angle of Attack
C	Coefficient
L/D	Lift-Drag Ratio
M	Mach Number
OSW	Oblique Shock Wave
P	Pressure
v	Prandtl-Meyer Function

Subscripts:

a	Tangential
i	Iterative Variable
inf	Freestream
l	Lower
n	Normal
L	Lift
D	Drag
M	Pitching Moment about Leading Edge
P	Pressure
U	Upper

List of Figures

Figure 1 Symmetric Airfoil 2
Figure 2 High Upper Thickness Airfoil 3
Figure 3 Lower Upper Thickness Airfoil 3
Figure 4 Finite Wing Element 4
Figure 5 β-θ-M Graph 6
Figure 6 Prandtl-Meyer Function 7
Figure 7 CL, CD, CM, L/D vs tu 9
Figure 8 CL, CD, CM, L/D vs xtu 9
Figure 9 CL, CD, CM, L/D vs xtl 9
Figure 10 CL, CD, CM, L/D vs tl 10
Figure 11 CL, CD, CM, L/D vs Angle of Attack 10
Figure 12 Constants and Variable Ranges 11
Figure 13 Nested Loop Organization 12
Figure 14 Analysis Code 12
Figure 15 Optimal Design Plot 13
Figure 16 Number of Solutions vs NSU 13
Figure 17 Time Elapsed vs Sensitivity 14
Figure 18 Max L/D vs Sensitivity 14
Figure 19 Viable Designs vs Sensitivity 15
Figure 20 Optimal Airfoil Shape 16

List of Tables

Table I. Optimal Design Parameters 16
Table II. Top 5 Optimal Design Parameters 16
Table II. Top 100 Feasible Designs 19

1.1 Design Objectives and Constraints

The objective of this design project is to optimize the Lift to Drag ratio of a supersonic, biconvex 2D airfoil via MATLAB programming. The goal is to have a fully generated model of an airfoil from MATLAB, simulate the effect of shock-expansion theory across multiple elements, similar to how Finite Element Analysis works, and run an optimization study where the selected wing element has the highest L/D over all viable solutions given the following constraints,

The selected wing element is assumed to be an 2D, infinitely long element flying at a Mach number of 3.0, where it has a unit chord of one meter such that the total thickness "t" must be at least ten percent or greater than the chord. The element also must have an angle of attack "α" between zero and ten degrees inclusive. Finally, the magnitude of the pitching moment about the leading edge must be less than or equal to 0.1, while also having a coefficient of lift "C_L" greater than or equal to 0.3.

$$\begin{Bmatrix} t = tu + tl \geq .1 \\ 0^o \leq \alpha \leq 10^2 \\ |C_M| \leq .1 \\ C_L \geq .3 \end{Bmatrix}$$

Variable Design Constraints

While the goal is to minimize drag while maximizing lift separately, a straightforward approach to completing this is via maximizing L/D which in turn takes both values as one consideration whenever designing for L/D. A viable solution for the project is a design that meets all the design constraints, but is not necessarily maximized for L/D. A single optimal solution therefore exists where all design constraints are met AND L/D is maximized for high efficiency,

1.2 Geometrical Setup

The first step of the design process is to create an analytical model of any wing element. Using a piecewise cubic spline, the following equation can be used for graphing the outer profile of an airfoil for both the upper and lower side, where x is the independent variable while *tu, tl, xtu,* and *xtl* – which represent maximum upper thickness, maximum lower thickness, x-position of *tu*, and x-position of *tl* – are all constants for a given airfoil. Note that the following only uses values for the upper wing section:

$$y_u = \frac{t_u}{x_{tu}^2} \cdot \frac{x}{(1-x_{tu})^2} \cdot [(1-2x_{tu}) \cdot x^2 + (3 * x_{tu}^2 - 1)x + x_{tu}(2-3x_{tu})]$$

Since this is a cubic function, the equation can be reduced to the form:

$$y_u = \frac{a_u x}{e_u} \cdot [b_u x^2 + c_u x + d_u]$$

$$y_l = \frac{a_l x}{e_l} \cdot [b_l x^2 + c_l x + d_l]$$

A function "airfoil_plot" is then created in MATLAB where, given values for *tu, tl, xtu,* and *xtl*, the airfoil is then graphed. Using a for loop, a few different designs shown inf Figs. 1-3 were created to show the effect of different design constants, which will soon be iterated upon as variables, on the design of a wing element. Figure 1 depicts a symmetric airfoil, while Figs. 2-3 show how the airfoil changes when thickness increases or decreases drastically.

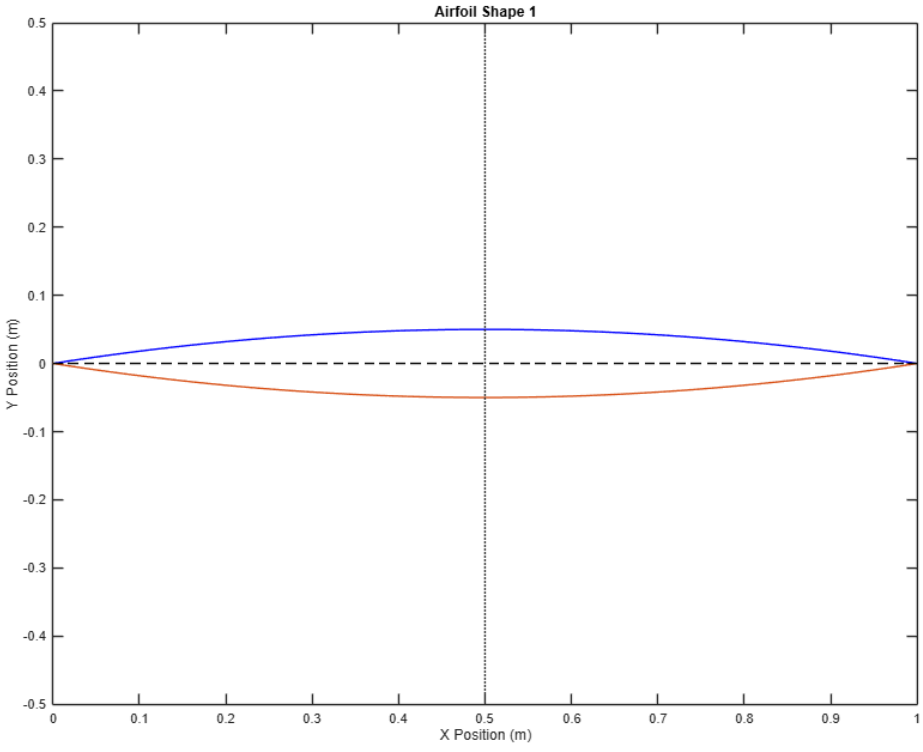


Figure 1 Symmetric Airfoil

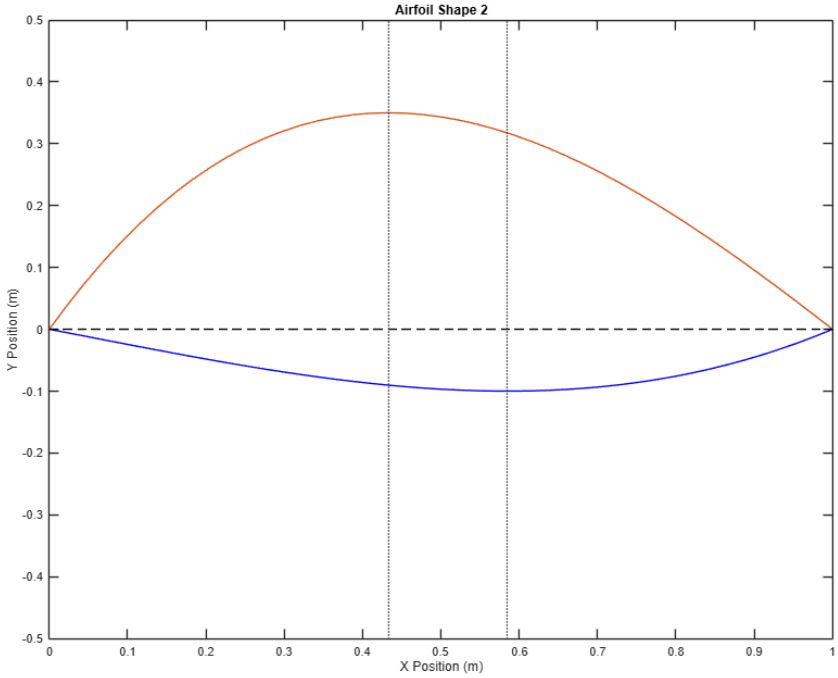


Figure 2 High Upper Thickness Airfoil

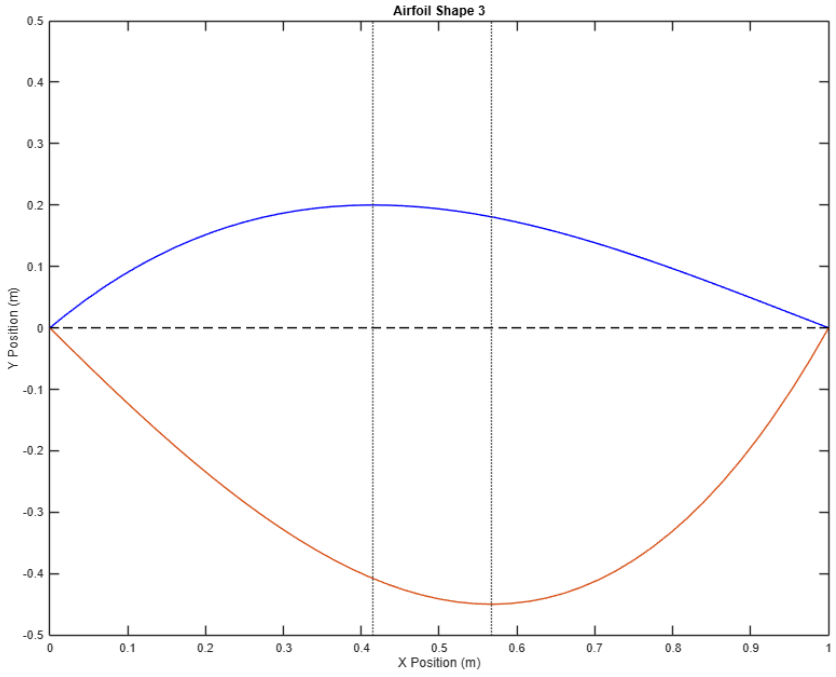


Figure 3 High Lower Thickness Airfoil

While these plots are useful for depicting what the airfoil shape will look like, they cannot be used in a numerical method approach to modelling a wing element. Using the built equation earlier, a set of points can be made for a range of x values, notated as “x_space”, between 0 and 1 with some incremental step value. A line is then created and treated as the wing surface. The step value is a constant called “nsu” which represents the number of surfaces to be created by the for loop and works as a fidelity setting for the given wing element. Throughout the analysis, nsu and nsl, which is the same but for the lower surface, are kept equal to ensure consistency of analysis. Figure 4 depicts a wing element where there are only 4 wing surfaces.

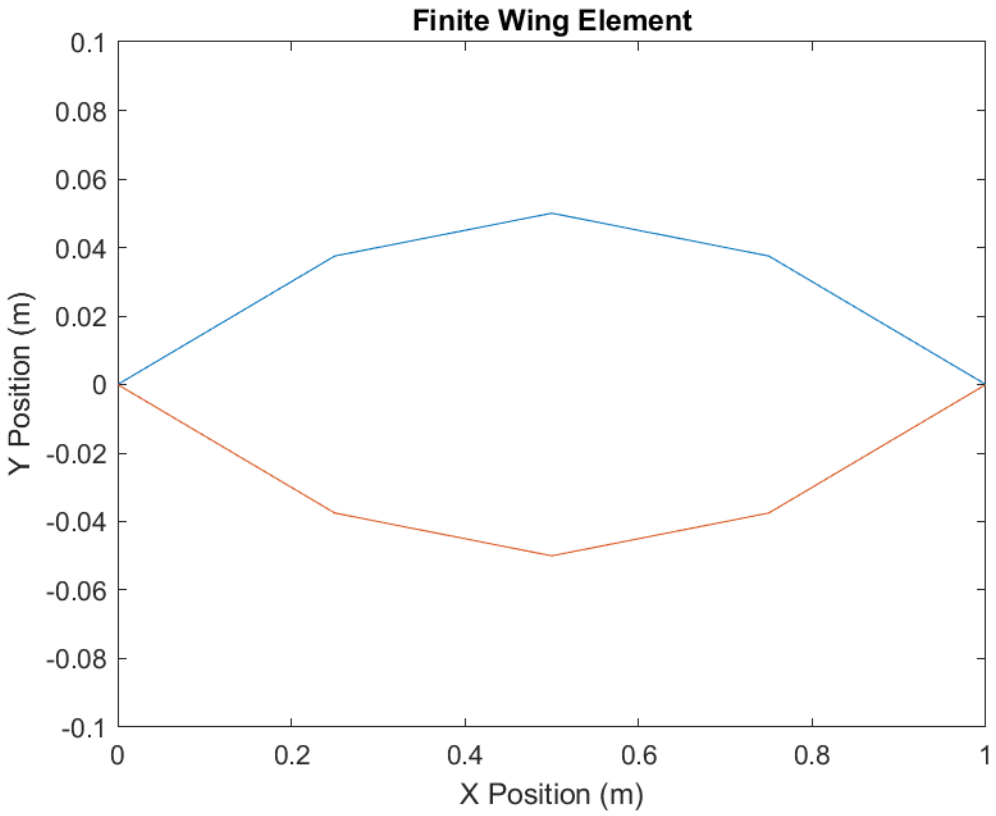


Figure 4 Finite Wing Element

It is also important to think about how different values of nsu will change the number of viable solutions. Theory depicts that as fidelity increases, the number of viable solutions to be selected will converge to a single, or small range, of values that will determine the ideal nsu value to set.

1.3 Airfoil Analysis Process

Now that the geometry has been constructed, aerodynamic analysis must be done for any given element. The goal is to calculate C_L, C_D, and C_M of an airfoil which is done via shock-expansion theory.

An airfoil can be separated into two sections from its leading edge: the upper surface and lower surface. For supersonic compressible flow, a shock wave will appear on the leading edge due to a sudden displacement within the air, causing the freestream air at M_{inf} to slow down and release energy. This displacement angle "θ" is measured as the angle between the direction of the free steam air "α", or angle of attack, and the tangent line of first contact with the surface. For a 2D finite wing element with "nsu" amount of upper and lower surfaces, this becomes the angle between the first finite surface and the AOA.

$$\theta_1 = \theta_o - \alpha$$

For an airfoil with no camber with low AOA's, typically both the upper and lower surface will see a shock wave being induced by the leading edge. Next, check whether or not θ_1 is positive or negative. If the deflection value is positive, then flow is being compressed into itself and creates an oblique shock wave. If it is negative, then the flow creates expansion waves as space is opened up. Finally, if the initial deflection angle equals the AOA of the wing, then there is no shock nor expansion wave forming due to no disturbances being created in the flow.

For the case that the deflection angle is positive, Oblique Shock Wave *"OSW"* Relations must be used. The shock wave creates a shock angle "β" such that flow separates into two components: perpendicular to the shock wave and parallel to the shock wave. This shock wave angle can be solved via the following equation:

$$tan(\theta) = 2\cot(\beta)\left[\frac{M_1 sin^2(\beta) - 1}{M_1(\gamma + cos(2\beta) + 2}\right]$$

This equation is plotted in Figure 5 where two solutions of β exists for a given deflection angle can exist: one where a weak shock wave permeates and one where a strong shock wave exists. For OSW relations, the weak solution is what is picked for analysis, as the strong solution tends to NSW relations. Note how this equation in reality is a 3D function of Mach number and deflection angle.

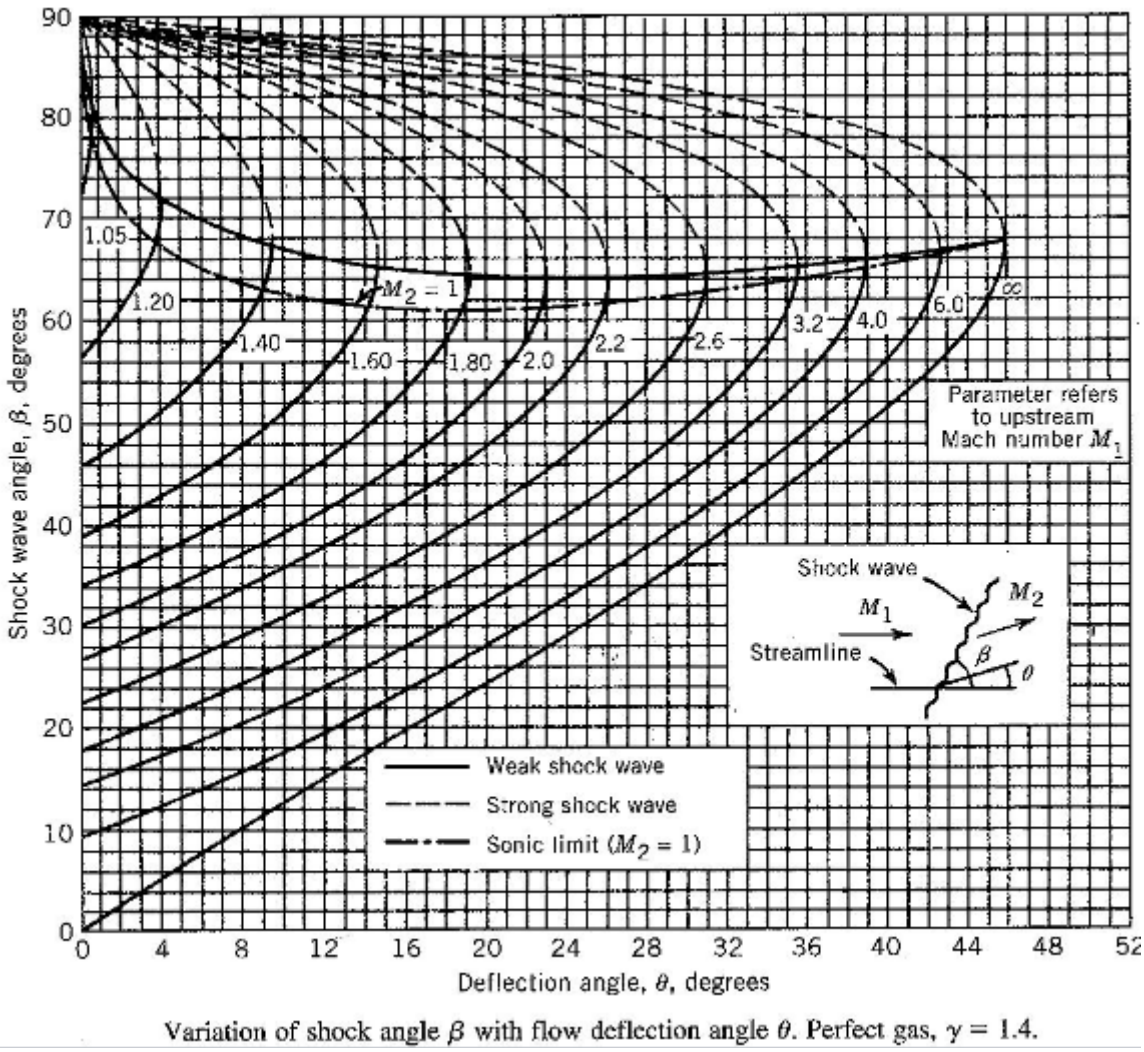


Figure 5 β-θ-M Graph

After finding the shock wave angle, the problem can be simplified to NSW relations via: $M_{n1} = M_1 \sin(\beta)$, which can be used in replace of M_1 when typically using NSW relations. Using this, pressure and flow normal to the shock wave can be found after the shock wave via:

$$\frac{P_2}{P_1} = 1 + \frac{2\gamma}{\gamma - 1}(M_1^2 - 1)$$

$$M_{n2} = \sqrt{\frac{1 + \frac{\gamma - 1}{2} M_{n1}^2}{\gamma M_{n1}^2 - \frac{\gamma - 1}{2}}}$$

After finding M_{n2}, the Mach number after the shock wave can be found: $M_2 = \frac{M_{n2}}{\sin(\beta - \theta_1)}$.

For the case that θ_1 is negative, the Prandtl-Meyer function, shown in Figure 6, is used to find the Mach number and flow properties after a shock wave:

$$v(M) = \sqrt{\frac{\gamma+1}{\gamma-1}}\tan^{-1}\left(\sqrt{\frac{\gamma-1}{\gamma+1}(M^2-1)}\right) - \tan^{-1}\left(\sqrt{M^2-1}\right)$$

Such that:

$$\theta_1 = v(M_2) - v(M_1)$$

Since θ_1 and M_1 is given, the second Mach number can be found via using the inverse of the Prandtl-Meyer function, which can be easily done by using Figure 6, along with the pressure along the surface via using the following isentropic relation.

$$\frac{P_2}{P_1} = \left(\frac{1 + \frac{\gamma-1}{2}M_1^2}{1 + \frac{\gamma-1}{2}M_2^2}\right)^{\frac{\gamma}{\gamma-1}}$$

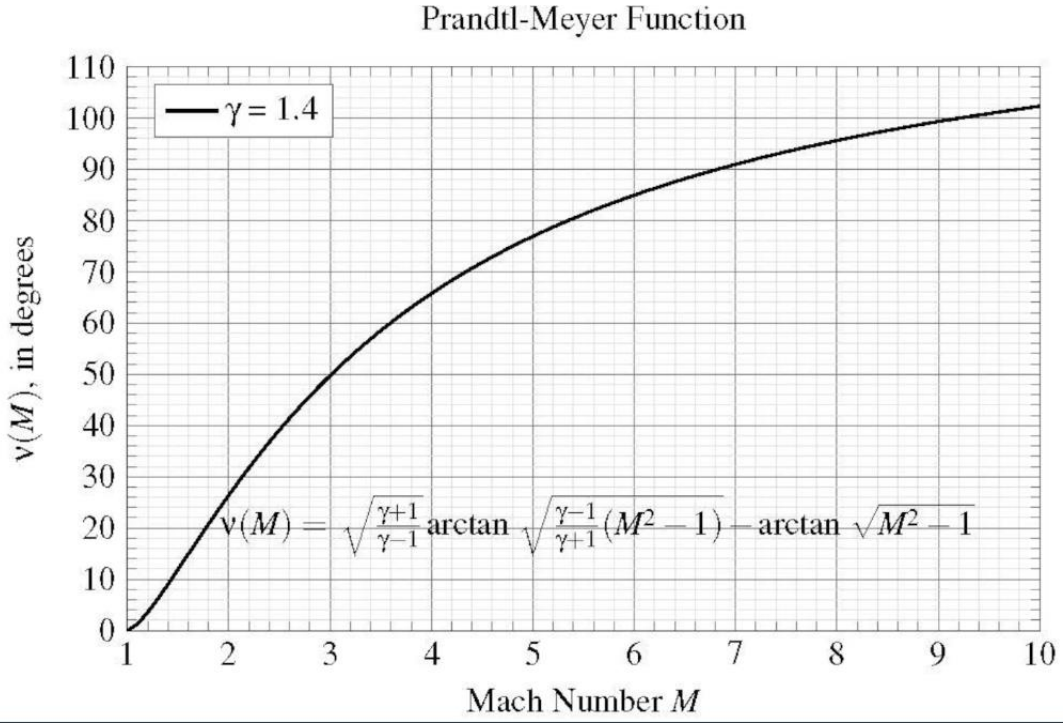


Figure 6 Prandtl-Meyer Function

For the initial case, M_1 is equal to the freestream Mach number M_{inf}. For any iterative cases, θ_1 becomes θ_i such that the leading edge θ_o becomes θ_{i-1}. The difference between each iterative panel is then used as a comparison for when to use OSW or Prandtl-Meyer relations, such that: $\Delta\theta = \theta_i - \theta_{i-1}$.

Coefficient of Pressure C_P for both the upper and lower section can be found via using the pressure at every iterative surface "i" such that:

$$C_{p,i} = \frac{\frac{P_i}{P_{inf}} - 1}{.5\gamma M_{inf}^2}$$

The coefficient of force normal and tangential to the surface can be calculated:

$$C_n = \int_0^1 (C_{P,l,i} - C_{P,U,i})\, d\bar{x}$$

$$C_a = \int_0^1 (C_{P,u,i} \frac{d\bar{y}_u}{d\bar{x}} - C_{P,l,i} \frac{d\bar{y}_l}{d\bar{x}})\, d\bar{x}$$

Where $\bar{x}$ and $\bar{y}$ are the x and y positions normalized to the chord: $\bar{x} = x/chord$ and $d\bar{y}_u/d\bar{x} = \tan(\Delta\theta)$. This equation inherently sums up all of the pressure coefficients along the airfoil and creates the non-dimensional force components on or along the surface. These equations can then rotated by the AOA and are used to find the Lift, Drag, and Moment coefficients of an airfoil:

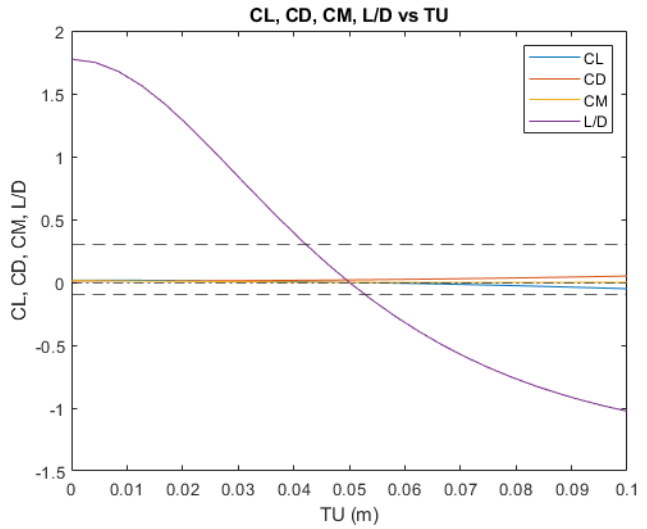
$$C_L = C_n \cos\alpha - C_a \sin\alpha$$

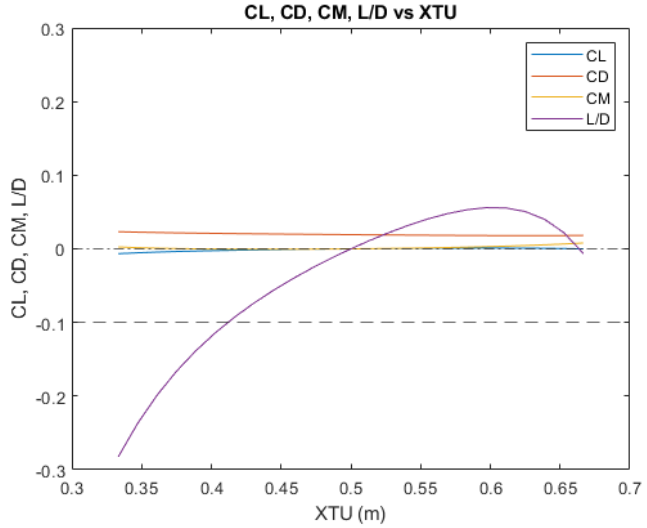
$$C_D = C_n \sin\alpha + C_a \cos\alpha$$

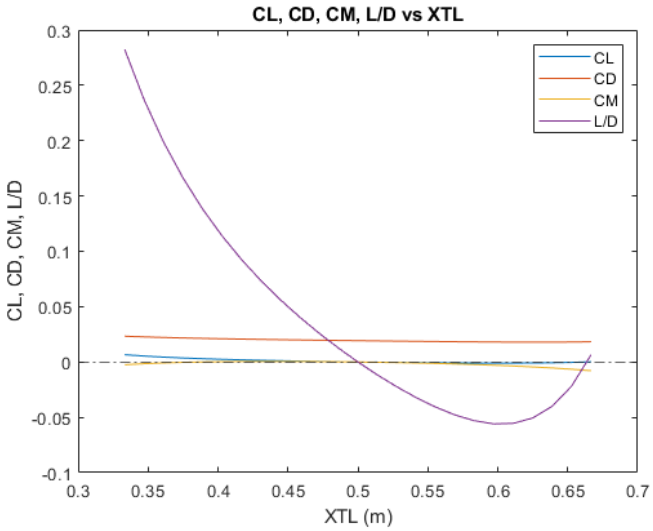
$$C_M = \int_0^1 (C_{P,u,i} - C_{P,L,i})\, \bar{x} d\bar{x} + \int_0^1 (C_{P,u,i} \frac{d\bar{y}_u}{d\bar{x}} - C_{P,l,i} \frac{d\bar{y}_l}{d\bar{x}})\, \bar{y} d\bar{x}$$

$$L/D = \frac{C_L}{C_D}$$

This now gives the total performance of the wing element normalized to its chord length. This process is completed by the p-code named "clcdcms", where it takes in the finite wing model geometry as a series of points, as well as freestream Mach number and the AOA, in order to efficiently solve for all of the components of lift and drag. Note that the p-code can't be opened unless permission is given and is to be used within MATLAB scripts as a function.



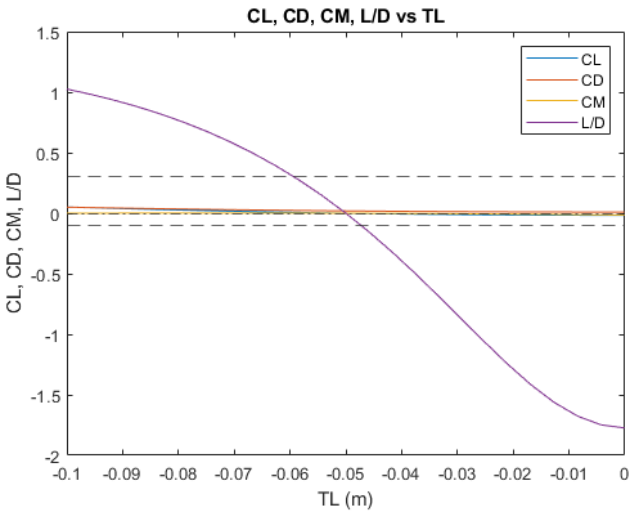




Now that the geometry is created and shock-expansion theory can be used, the different variables can be iterated upon to see how they affect C_L, C_D, C_M, and L/D as shown in Figs. 7-11. These plots were created via iterating on variables for a symmetric airfoil at nsu = 50 and show where the maximum L/D is for every change in variables. A line is drawn for the x-axis, as well as the C_M design cut off at 0.1 and the C_L minimum at 0.3. These graphs tend to have a maximum L/D at the values of:

$$\begin{bmatrix} \alpha \approx 6^\circ \\ t_U \approx 0 \\ t_l \approx -.1 \\ x_{tu} \approx 0.62 \\ x_{tl} \approx 0.32 \end{bmatrix}$$

Although the thickness values are related to one another from the constraint equation, this shows that theoretically, a high lower thickness and curvature creates lots of static pressure on the bottom surface while expansion waves are created on the very flat top surface. Realistically, the optimization should have approached these values but not equal 0 and -.1, since they still need to meet the C_M and C_L requirements. Likewise, the optimal values for xtu and xtl should also approach the



previously stated values. Please note that there will be a range of feasible designs that are close to these options.

Figures 7-10 C_L, C_D, C_M, L/D vs Design Variables

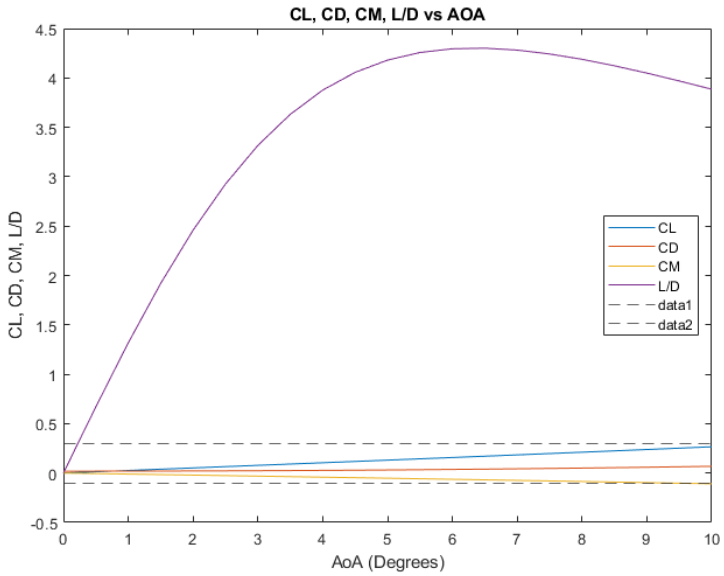


Figure 11 C_L, C_D, C_M, L/D vs Angle of Attack

1.4 Optimal Design Process

Finally, the optimization process can go underway. To construct an optimization program, one can use a brute force method where all the design variables are iterated upon with some finite specified step over one at a time. Before iterating however, the constants need to be set, and the ranges of the code need to be created using the

linspace(lower bound, upper bound, step-over sensitivity) function such that the step over sensitivity is how many times the code iterates between the lower and upper bound. The function then creates an array, or matrix, that contains the entire range of values, which is named as *variable*_space. For example, in Fig. 12 the range of upper thicknesses values is labeled as tu_space. Notice how the x_space range has one more value than the number of surfaces. This is because there must be n+1 number of points to create an n number of lines on an open loop for any shape.

```
%CONSTANTS
results = [];
temp_results = {};
Minf = 3.0;
chord = 1;
viable_design_count = 0;

%SENSITIVITIES
nsu = 50;
nsl = nsu;
aoa_sense = nsu;
xtu_sens = nsu;
xtl_sens = nsu;
tu_sens = nsu;

yu = zeros(1, nsu+1);
yl = zeros(1, nsl+1);

%VARIABLE RANGE
alpha_space = linspace(0,10, aoa_sense+1);
xtu_space = linspace(1/3, 2/3, xtu_sens);
xtl_space = linspace(1/3, 2/3, xtl_sens);
tu_space = linspace(0.0001, .1, tu_sens);
x_space = linspace(0,chord, nsu+1);
```

Figure 12 Constants and Variable Ranges

Next, the optimization “for loop” iteration code must be made. A For loop takes a variable and lets it equal to a value within a series of values, contained in an array. Initially, every single variable was looped through without much consideration as to how long the code will take to parse. A series of seven for nested for loops was first used for a brute-force methodology of calculating the optimal wing design. This gives a time complexity of $O(n^7)$, meaning that for a function parsing through a dataset size of “n” takes proportionally n^7 times the amount of time to finish the code. This yielded unnecessarily longer results and was eventually shortened down to only four nested for loops, which later proved to have a much smaller iteration time as it has a time complexity of $O(n^4)$. The only variables that are actually iterated via a for loop are categorized in Figure 13.

```
for tu = tu_space
    tl = -abs(.1-tu);
    if (tu + abs(tl) >= 0.1)
        for xtl = xtl_space
            for xtu = xtu_space
                for alphad = alpha_space
```

Figure 13 Nested Loop Organization

Notice how "tl" is a function of tu such that it will always meet the minimum design criteria. This was chosen due to small thickness consistently yielding the maximum L/D criterion, as any attempt at iterating tl separately caused a decrease in maximum L/D from 3.31 to 2.46. This L/D is calculated using the previously given p-code that calculates L/D using the above explained finite analysis process, as shown in Fig. 14.

The airfoil constants are calculated in the same manner as the geometrical setup section, and another for loop is used to create the finite wing element, as mentioned previously. Finally, the AOA is iterated upon last since it is an independent variable from xtu, tu, etc., and the aerodynamic properties are calculated from the p-code. These values are then checked from an if statement, checking that "if" C_M and C_L exceed the design requirements, they will be saved in a temporary results matrix, as multiple copies of the same feasible design are sometimes created. Otherwise, the design is ignored.

```
for xtu = xtu_space
    %AIRFOIL EQN. CONSTANTS
    a1 = (tu / (xtu^2));
    a2 = (tl / (xtl^2));
    b1 = 1-2*xtu;
    b2 = 1-2*xtl;
    c1 = 3*xtu^2 - 1;
    c2 = 3*xtl^2 - 1;
    d1 = xtu * (2 - 3*xtu);
    d2 = xtl * (2 - 3*xtl);
    e1 = (1-xtu)^2;
    e2 = (1-xtl)^2;

    for i = 1:length(x_space)
        yu(i) = (x_space(i) * a1/e1) * (b1 * x_space(i)^2 + c1 * x_space(i) + d1);
        yl(i) = (x_space(i) * a2/e2) * (b2 * x_space(i)^2 + c2 * x_space(i) + d2);
    end

    %P CODE
    for alphad = alpha_space
        [cl, cd, cm, xmu, cpu, xml, cpl] = clcdcms(Minf, alphad, nsu, x_space, yu, nsl, x_space, yl);

        %CM & L/D VALUE
        if abs(cm) <= .1 && cl >= .3
            LD = cl/cd;
            specs = [alphad, tu, tl, xtu, xtl, cl, cd, cm, LD];
            temp_results{end + 1} = specs;
        end
    end
```

Figure 14 Analysis Code

After shortening the general form of the code, the results for every iteration was saved and organized in an “n x 9” matrix, such that the first 5 columns is the airfoil’s geometry features and the last 4 columns are the C_L, C_D, C_M, and L/D respectively. The maximum L/D ratio is found in the final column, and the row that has all the design features is appended to an array that graphs the optimal shape via “airfoil_plot” as mentioned earlier and shown in Figure 15.

```
results = unique(cell2mat(temp_results'), 'rows');
viable_design_count = length(results);

%FINDS MAXIMUM L/D
[max_LD, max_index] = max(results(:, 9));
optimal_design = results(max_index, :); %SORTED VIA: ALPHA = 1, TU=2, TL=3, XTU=4, XTL=5, CL=6, CD=7, CM=8, LD=9

%PLOT AIRFOIL SHAPE
tu = optimal_design(2);
tl = optimal_design(3);
xtu = optimal_design(4);
xtl = optimal_design(5);
LD_max = optimal_design(9);

airfoil_plot(tu, tl, xtu, xtl, LD_max, 1)
time = toc; %ENDS TIME MEASUREMENT
fprintf('Number of Viable Designs: %d\n', viable_design_count)
fprintf('Time Elapsed: %.2f seconds\n', time)
fprintf('Max L/D = %.3f\n', LD_max)
```

Figure 15 Optimal Design Plot

The first design consideration that should be narrowed down is the number of panels needed for creating the geometry of the wing. A rough convergence study was done to see at what value of “nsu” will cause the number of solutions to converge. Figure 16 shows that the number of solutions converges at around 60 and thus should be used for the optimization process.

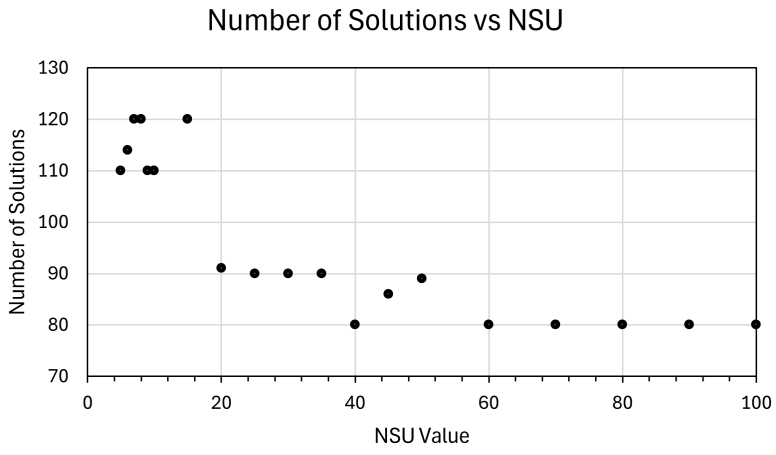


Figure 16 Number of Solutions vs NSU

Practically however, since the time complexity is still $O(n^4)$, a value of around 30-50 is more reasonable. Figures 17-18 depicts this issue when varying the "sensitivities" - the step over between each iteration for a variable inputted into the linspace() MATLAB function – and measuring the time elapsed and the Max L/D. A higher sensitivity means a lower increase in a variable's value between each iteration.

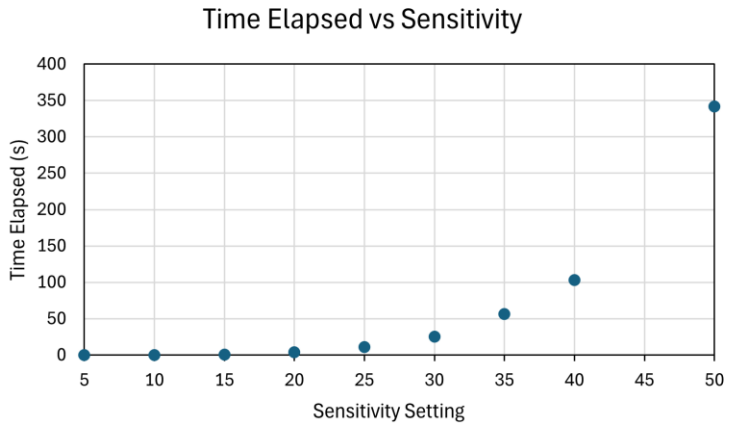


Figure 17 Time Elapsed vs Sensitivity

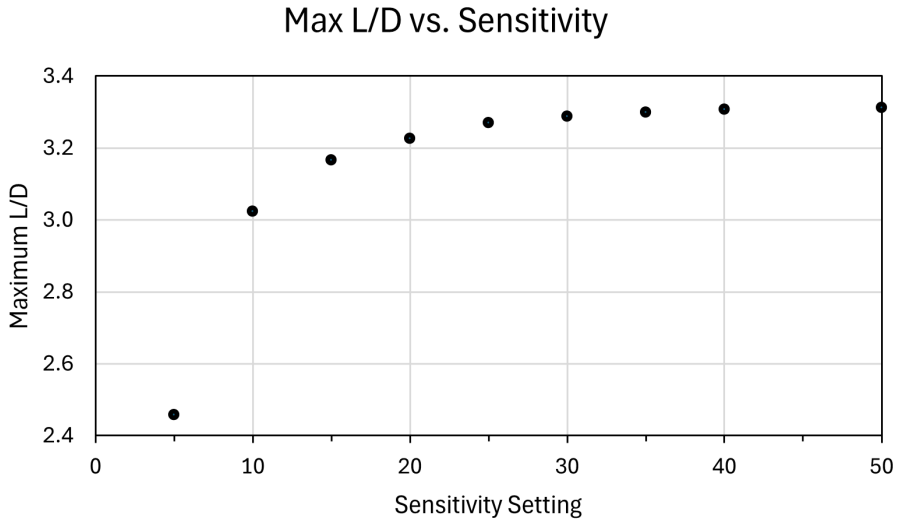


Figure 18 Max L/D vs Sensitivity

This shows that an average sensitivity setting of around 30-40 should be used if multiple optimizations in succession are wanted to be completed. Otherwise, a sensitivity of 50 is best used for a one-time run, as the % increase between 40 and 50

is only 0.15% and thus not worth the increase in calculation time. Likewise, the number of viable solutions increases exponentially as shown in Figure 19 as the sensitivity goes up while keeping NSU constant.

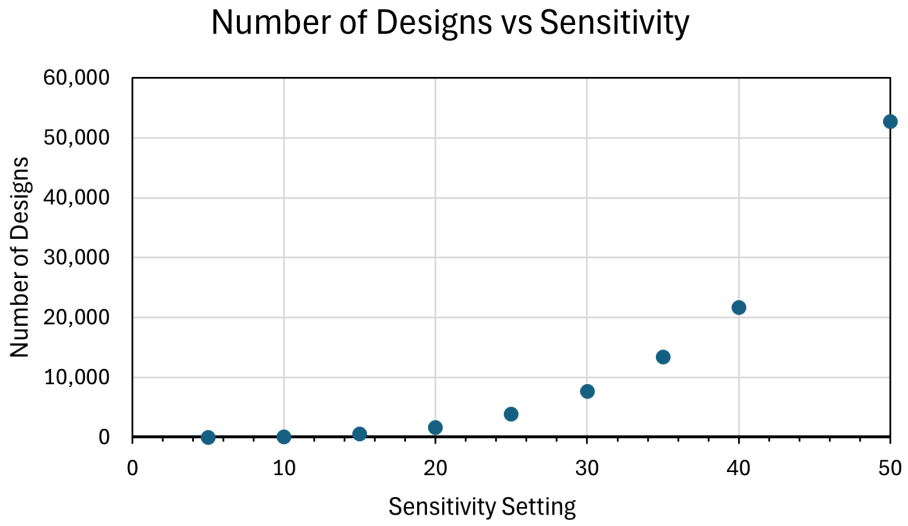


Figure 19 Viable Designs vs Sensitivity

This means that it is not feasible to graph and tabulate every single design, which makes sense if one thinks about it as a mean value theorem problem. If two designs work at a lower value for a variable, say AOA, and another works at a higher AOA: there must exist a feasible design such that AOA is between the lower and higher value. With high fidelity, the number of feasible designs should increase drastically, as was shown in Figure 19.

In the end, it is reasonable to set an NSU value of 50 as well as a sensitivity of 50 as well, as the run time should ideally be under 10 minutes maximum. Since a sensitivity of 50 has a six-minute run time, this will be the iteration step over for every variable. The top 100 feasible designs were saved and organized in Table III in the Appendix.

1.5 Results

After parsing through 52,717 viable designs, the solution that had the optimal L/D is seen in Fig. 20, where its L/D value was 3.311. Notice how the top surface is generally flat meanwhile the bottom surface is larger near the leading edge. This is generated from the stance that thickness is exactly the minimum required by the design constraints, as any larger would potentially decrease the L/D ratio. Please note that the top feasible designs are very similar to the optimal airfoil.

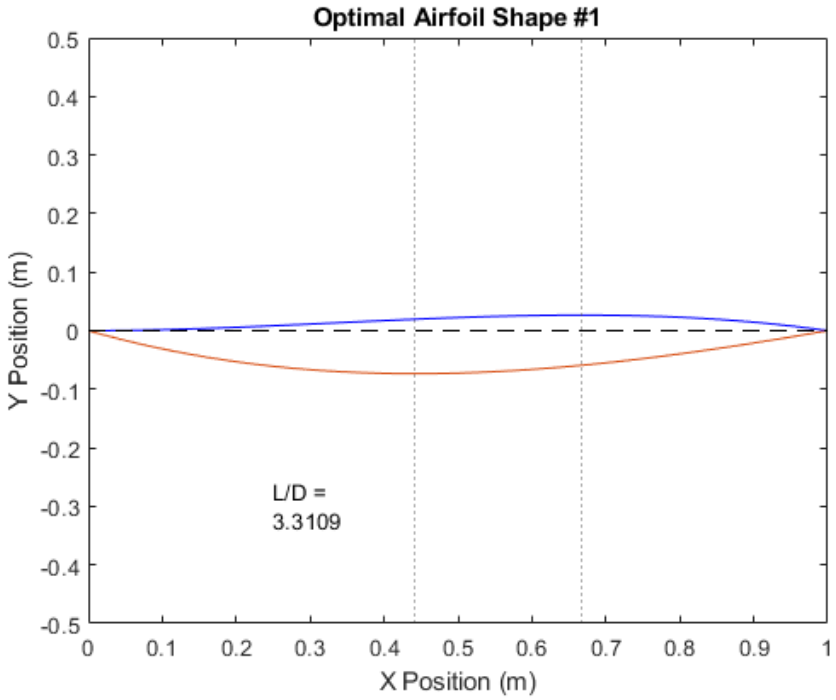


Figure 20 Optimal Airfoil Shape

Table I shows the associated design parameters, where CL is right above 0.3 but CD is much lower. The following 5 optimal designs all have an L/D of 3.295, with very similar airfoil design as Fig. 19, as shown in Table II.

Table I Optimal Design Parameters

AOA (degrees)	TU	TL	XTU	XTL	CL	CD	CM	L/D
9.6	0.0266	-0.0734	0.6667	0.4422	0.3001	0.0906	-0.1000	3.3109

Table II Top 5 Optimal Designs

AOA (degrees)	Tu (m)	Tl (m)	Xtu (m)	Xtl (m)	CL	CD	CM	L/D
9.6	0.0266	-0.0734	0.6667	0.4422	0.3001	0.0906	-0.1000	3.3109
9.6	0.0246	-0.0754	0.6054	0.4626	0.3005	0.0913	-0.0999	3.2935
9.6	0.0246	-0.0754	0.6122	0.4626	0.3005	0.0913	-0.0999	3.2935
9.6	0.0246	-0.0754	0.6190	0.4626	0.3005	0.0913	-0.0998	3.2934
9.6	0.0246	-0.0754	0.6259	0.4626	0.3005	0.0913	-0.0997	3.2932

Remember that from the preliminary analysis for a symmetric airfoil, the expected values for optimal L/D were:

$$\begin{bmatrix} \alpha \approx 6^\circ \\ t_U \approx 0 \\ t_l \approx -.1 \\ x_{tu} \approx 0.62 \\ x_{tl} \approx 0.32 \end{bmatrix}$$

The geometry values match closely to the values seen in the optimal design, excluding the angle of attack. In the optimization, AOA was observed to vary between 7.5 and 10 degrees. Which is quite different than the previously predicted value of alpha as from Fig. 11, alpha decreased downwards after 6 degrees, although the graph did not meet the design requirements. After accounting for all the design constraints, the final L/D value was settled to be 3.311

Whilst an optimal design was found for the dataset produced from the code, there is still room for improvement. The upper and lower thicknesses could be separated and iterated differently such that an optimal solution above the minimum thickness is found, along with C_M and C_L being further from the desired minimum. Similarly, given a more powerful computer, the sensitivities and iteration steps for every variable can be separated and varied separately to approach a more ideal value. Another improvement can be the time complexity of the system, as four nested for-loops creates an exponentially increasing amount of solving time. This can be done by designing a better algorithm for optimization than a brute-force method. However, this can be left as an exercise to the computer scientists.

Appendix A

Optimization Code (Code is also Attached):

```
clc;
clear;
format long

%CONSTANTS
results = [];
temp_results = {};
Minf = 3.0;
chord = 1;
viable_design_count = 0;

%SENSITIVITIES
nsu = 50;
nsl = nsu;
aoa_sense = nsu;
xtu_sens = nsu;
xtl_sens = nsu;
tu_sens = nsu;

yu = zeros(1, nsu+1);
yl = zeros(1, nsl+1);

%VARIABLE RANGE
alpha_space = linspace(0,10, aoa_sense+1);
xtu_space = linspace(1/3, 2/3, xtu_sens);
xtl_space = linspace(1/3, 2/3, xtl_sens);
tu_space = linspace(0.0001, .1, tu_sens);
x_space = linspace(0,chord, nsu+1);

tic %TIME MEASUREMENT

%OPTIMIZATION FOR-LOOP
for tu = tu_space
   tl = -abs(.1-tu);
   if (tu + abs(tl) >= 0.1)
      for xtl = xtl_space
         for xtu = xtu_space
            %AIRFOIL EQN. CONSTANTS
            a1 = (tu / (xtu^2));
            a2 = (tl / (xtl^2));
            b1 = 1-2*xtu;
            b2 = 1-2*xtl;
            c1 = 3*xtu^2 - 1;
            c2 = 3*xtl^2 - 1;
            d1 = xtu * (2 - 3*xtu);
            d2 = xtl * (2 - 3*xtl);
            e1 = (1-xtu)^2;
            e2 = (1-xtl)^2;

            for i = 1:length(x_space)
               yu(i) = (x_space(i) * a1/e1) * (b1 * x_space(i)^2 + c1 * x_space(i) + d1);
               yl(i) = (x_space(i) * a2/e2) * (b2 * x_space(i)^2 + c2 * x_space(i) + d2);
            end

            %P CODE
            for alphad = alpha_space
               [cl, cd, cm, xmu, cpu, xml, cpl] = clcdcms(Minf, alphad, nsu, x_space, yu, nsl, x_space, yl);

               %CM & L/D VALUE
```

```
            if abs(cm) <= .1 && cl >= .3
               LD = cl/cd;
               specs = [alphad, tu, tl, xtu, xtl, cl, cd, cm, LD];
               temp_results{end + 1} = specs;
            end
          end
        end
      end
    end
end

results = unique(cell2mat(temp_results'), 'rows');
viable_design_count = length(results);

%FINDS MAXIMUM L/D
[max_LD, max_index] = max(results(:, 9));
optimal_design = results(max_index, :); %SORTED VIA: ALPHA = 1, TU=2, TL=3, XTU=4, XTL=5, CL=6, CD=7, CM=8, LD=9

%PLOT AIRFOIL SHAPE
tu = optimal_design(2);
tl = optimal_design(3);
xtu = optimal_design(4);
xtl = optimal_design(5);
LD_max = optimal_design(9);

airfoil_plot(tu, tl, xtu, xtl, LD_max, 1)
time = toc; %ENDS TIME MEASUREMENT
fprintf('Number of Viable Designs: %d\n', viable_design_count)
fprintf('Time Elapsed: %.2f seconds\n', time)
fprintf('Max L/D = %.3f\n', LD_max)

fprintf('Max L/D = %.3f\n', LD_max)
```

Table III Top 100 Feasible Designs

Angle of Attack (deg.)	Tu (m)	Tl (m)	Xtu (m)	Xtl (m)	CL	CD	CM	L/D
9.6	0.0266	-0.0734	0.6667	0.4422	0.3001	0.0906	-0.1000	3.3109
9.6	0.0246	-0.0754	0.6054	0.4626	0.3005	0.0913	-0.0999	3.2935
9.6	0.0246	-0.0754	0.6122	0.4626	0.3005	0.0913	-0.0999	3.2935
9.6	0.0246	-0.0754	0.6190	0.4626	0.3005	0.0913	-0.0998	3.2934
9.6	0.0246	-0.0754	0.6259	0.4626	0.3005	0.0913	-0.0997	3.2932
9.6	0.0246	-0.0754	0.6327	0.4626	0.3005	0.0913	-0.0996	3.2930
9.6	0.0246	-0.0754	0.6395	0.4626	0.3005	0.0913	-0.0995	3.2927
9.6	0.0246	-0.0754	0.6463	0.4626	0.3005	0.0913	-0.0994	3.2923
9.6	0.0246	-0.0754	0.6531	0.4626	0.3004	0.0913	-0.0992	3.2917
9.6	0.0246	-0.0754	0.6599	0.4626	0.3004	0.0913	-0.0991	3.2909
9.6	0.0246	-0.0754	0.6667	0.4626	0.3003	0.0913	-0.0990	3.2900
9.4	0.0266	-0.0734	0.6599	0.4150	0.3002	0.0913	-0.0999	3.2890
9.4	0.0266	-0.0734	0.6667	0.4150	0.3001	0.0913	-0.0997	3.2879
9.6	0.0246	-0.0754	0.6327	0.4558	0.3018	0.0920	-0.1000	3.2819
9.6	0.0246	-0.0754	0.6395	0.4558	0.3018	0.0920	-0.0999	3.2816
9.6	0.0246	-0.0754	0.6463	0.4558	0.3017	0.0920	-0.0997	3.2811

9.6	0.0246	-0.0754	0.6531	0.4558	0.3017	0.0920	-0.0996	3.2805
9.6	0.0246	-0.0754	0.6599	0.4558	0.3017	0.0920	-0.0995	3.2798
9.6	0.0246	-0.0754	0.6667	0.4558	0.3016	0.0920	-0.0993	3.2789
9.6	0.0225	-0.0775	0.6054	0.4830	0.3007	0.0919	-0.0991	3.2740
9.6	0.0225	-0.0775	0.5986	0.4830	0.3007	0.0919	-0.0992	3.2739
9.6	0.0225	-0.0775	0.6122	0.4830	0.3007	0.0919	-0.0991	3.2739
9.6	0.0225	-0.0775	0.5918	0.4830	0.3007	0.0919	-0.0993	3.2739
9.6	0.0225	-0.0775	0.6190	0.4830	0.3007	0.0919	-0.0990	3.2739
9.6	0.0225	-0.0775	0.5850	0.4830	0.3007	0.0919	-0.0993	3.2738
9.6	0.0225	-0.0775	0.5782	0.4830	0.3007	0.0919	-0.0994	3.2737
9.6	0.0225	-0.0775	0.6259	0.4830	0.3007	0.0919	-0.0989	3.2737
9.6	0.0225	-0.0775	0.5714	0.4830	0.3007	0.0919	-0.0994	3.2736
9.6	0.0225	-0.0775	0.6327	0.4830	0.3007	0.0919	-0.0988	3.2735
9.6	0.0225	-0.0775	0.5646	0.4830	0.3007	0.0919	-0.0995	3.2734
9.6	0.0225	-0.0775	0.5578	0.4830	0.3007	0.0919	-0.0995	3.2733
9.6	0.0225	-0.0775	0.6395	0.4830	0.3007	0.0919	-0.0987	3.2732
9.6	0.0225	-0.0775	0.5510	0.4830	0.3007	0.0919	-0.0995	3.2731
9.6	0.0225	-0.0775	0.5442	0.4830	0.3007	0.0919	-0.0996	3.2729
9.6	0.0225	-0.0775	0.6463	0.4830	0.3007	0.0919	-0.0986	3.2728
9.6	0.0225	-0.0775	0.5374	0.4830	0.3007	0.0919	-0.0996	3.2727
9.6	0.0225	-0.0775	0.5306	0.4830	0.3007	0.0919	-0.0996	3.2725
9.6	0.0225	-0.0775	0.6531	0.4830	0.3006	0.0919	-0.0985	3.2723
9.6	0.0225	-0.0775	0.5238	0.4830	0.3007	0.0919	-0.0997	3.2723
9.6	0.0225	-0.0775	0.5170	0.4830	0.3007	0.0919	-0.0997	3.2720
9.6	0.0225	-0.0775	0.5102	0.4830	0.3007	0.0919	-0.0997	3.2718
9.6	0.0225	-0.0775	0.6599	0.4830	0.3006	0.0919	-0.0984	3.2717
9.6	0.0225	-0.0775	0.5034	0.4830	0.3007	0.0919	-0.0997	3.2715
9.6	0.0225	-0.0775	0.4966	0.4830	0.3007	0.0919	-0.0998	3.2713
9.6	0.0225	-0.0775	0.4898	0.4830	0.3007	0.0919	-0.0998	3.2710
9.6	0.0225	-0.0775	0.6667	0.4830	0.3005	0.0919	-0.0982	3.2709
9.6	0.0225	-0.0775	0.4830	0.4830	0.3007	0.0919	-0.0998	3.2708
9.4	0.0246	-0.0754	0.6054	0.4354	0.3002	0.0918	-0.0992	3.2707
9.4	0.0246	-0.0754	0.6122	0.4354	0.3002	0.0918	-0.0991	3.2707
9.4	0.0246	-0.0754	0.5986	0.4354	0.3002	0.0918	-0.0992	3.2707
9.4	0.0246	-0.0754	0.6190	0.4354	0.3002	0.0918	-0.0990	3.2706
9.4	0.0246	-0.0754	0.5918	0.4354	0.3002	0.0918	-0.0993	3.2706
9.4	0.0246	-0.0754	0.6259	0.4354	0.3001	0.0918	-0.0989	3.2705
9.4	0.0246	-0.0754	0.5850	0.4354	0.3002	0.0918	-0.0994	3.2705
9.6	0.0225	-0.0775	0.4762	0.4830	0.3007	0.0919	-0.0998	3.2705

9.4	0.0246	-0.0754	0.5782	0.4354	0.3002	0.0918	-0.0994	3.2703
9.4	0.0246	-0.0754	0.6327	0.4354	0.3001	0.0918	-0.0988	3.2703
9.6	0.0225	-0.0775	0.4694	0.4830	0.3007	0.0919	-0.0998	3.2702
9.4	0.0246	-0.0754	0.5714	0.4354	0.3002	0.0918	-0.0995	3.2701
9.4	0.0246	-0.0754	0.5646	0.4354	0.3002	0.0918	-0.0995	3.2699
9.4	0.0246	-0.0754	0.6395	0.4354	0.3001	0.0918	-0.0987	3.2699
9.6	0.0225	-0.0775	0.4626	0.4830	0.3007	0.0919	-0.0998	3.2699
9.4	0.0246	-0.0754	0.5578	0.4354	0.3002	0.0918	-0.0996	3.2697
9.6	0.0225	-0.0775	0.4558	0.4830	0.3006	0.0920	-0.0998	3.2696
9.4	0.0246	-0.0754	0.5510	0.4354	0.3002	0.0918	-0.0996	3.2695
9.4	0.0246	-0.0754	0.6463	0.4354	0.3001	0.0918	-0.0986	3.2695
9.6	0.0225	-0.0775	0.4490	0.4830	0.3006	0.0920	-0.0998	3.2692
9.4	0.0246	-0.0754	0.5442	0.4354	0.3002	0.0918	-0.0997	3.2692
9.4	0.0246	-0.0754	0.5374	0.4354	0.3002	0.0918	-0.0997	3.2690
9.6	0.0225	-0.0775	0.4422	0.4830	0.3006	0.0920	-0.0998	3.2689
9.4	0.0246	-0.0754	0.6531	0.4354	0.3000	0.0918	-0.0985	3.2689
9.6	0.0246	-0.0754	0.6599	0.4490	0.3030	0.0927	-0.0999	3.2687
9.4	0.0246	-0.0754	0.5306	0.4354	0.3002	0.0918	-0.0998	3.2687
9.6	0.0225	-0.0775	0.4354	0.4830	0.3006	0.0920	-0.0998	3.2685
9.4	0.0246	-0.0754	0.5238	0.4354	0.3001	0.0918	-0.0998	3.2684
9.6	0.0225	-0.0775	0.4286	0.4830	0.3006	0.0920	-0.0998	3.2682
9.4	0.0246	-0.0754	0.6599	0.4354	0.3000	0.0918	-0.0983	3.2682
9.4	0.0246	-0.0754	0.5170	0.4354	0.3001	0.0918	-0.0998	3.2681
9.6	0.0246	-0.0754	0.6667	0.4490	0.3030	0.0927	-0.0998	3.2678
9.6	0.0225	-0.0775	0.4218	0.4830	0.3006	0.0920	-0.0998	3.2678
9.4	0.0246	-0.0754	0.5102	0.4354	0.3001	0.0918	-0.0999	3.2678
9.4	0.0246	-0.0754	0.5034	0.4354	0.3001	0.0919	-0.0999	3.2674
9.6	0.0225	-0.0775	0.4150	0.4830	0.3006	0.0920	-0.0998	3.2674
9.4	0.0246	-0.0754	0.4966	0.4354	0.3001	0.0919	-0.0999	3.2671
9.6	0.0225	-0.0775	0.4082	0.4830	0.3006	0.0920	-0.0998	3.2669
9.4	0.0246	-0.0754	0.4898	0.4354	0.3001	0.0919	-0.0999	3.2667
9.6	0.0225	-0.0775	0.4014	0.4830	0.3005	0.0920	-0.0998	3.2664
9.4	0.0246	-0.0754	0.4830	0.4354	0.3001	0.0919	-0.0999	3.2664
9.4	0.0246	-0.0754	0.4762	0.4354	0.3001	0.0919	-0.1000	3.2660
9.6	0.0225	-0.0775	0.3946	0.4830	0.3005	0.0920	-0.0998	3.2659
9.4	0.0246	-0.0754	0.4694	0.4354	0.3001	0.0919	-0.1000	3.2656
9.6	0.0225	-0.0775	0.3878	0.4830	0.3005	0.0920	-0.0997	3.2653
9.4	0.0246	-0.0754	0.4626	0.4354	0.3001	0.0919	-0.1000	3.2653
9.4	0.0246	-0.0754	0.4558	0.4354	0.3001	0.0919	-0.1000	3.2649

9.6	0.0225	-0.0775	0.3810	0.4830	0.3005	0.0920	-0.0997	3.2647
9.6	0.0225	-0.0775	0.3741	0.4830	0.3005	0.0921	-0.0997	3.2641
9.6	0.0225	-0.0775	0.3673	0.4830	0.3004	0.0921	-0.0996	3.2633
9.6	0.0225	-0.0775	0.3605	0.4830	0.3004	0.0921	-0.0996	3.2625
9.6	0.0225	-0.0775	0.6054	0.4762	0.3020	0.0926	-0.0994	3.2625
9.6	0.0225	-0.0775	0.5986	0.4762	0.3020	0.0926	-0.0994	3.2625