Airfoil Boundary Layer Separation Prediction

A project present to
The Faculty of the Department of Aerospace Engineering
San Jose State University

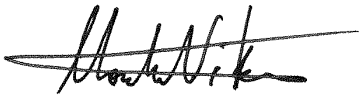
in partial fulfillment of the requirements for the degree
Master of Science in Aerospace Engineering

By

Kartavya Patel

May 2014

approved by



Dr. Nikos Mourtos
Faculty Advisor

ABSTRACT

Airfoil Boundary Layer Separation Prediction by Kartavya Patel

This project features a MatLab complied program that predicts airfoil boundary layer separation. The Airfoil Boundary Layer Separation program uses NACA 4 series, 5 series and custom coordinates to generate the airfoil geometry. It then uses Hess-Smith Panel Method to generate the pressure distribution. It will use the pressure distribution profile to display the boundary layer separation point based on Falkner-Skan Solution, Stratford's Criterion for Laminar Boundary Layer and Stratford's Criterion for Turbulent Boundary Layer. From comparison to Xfoil, it can be determined that for low angle of attacks the laminar flow separation point can be predicted from Stratford's LBL criterion and the turbulent flow separation point can be predicted from Stratford's TBL criterion. For high angle of attacks, the flow separation point can be predicted from Falkner-Skan Solution. The program requires MatLab Compiler Runtime version 8.1 (MCR) which can be downloaded free at http://www.mathworks.com/products/compiler/mcr/.

ACKNOWLEDGEMENTS

I would like to thank Dr. Nikos Mourtos for his support and guidance throughout this project. I would also like to thank Hai Le, Tommy Blackwell and Ian Dupzyk for their contribution in previous project “Determination of Flow Separation Point on NACA Airfoils at Different Angles of Attack by Coupling the Solution of Panel Method with Three Different Separation Criteria”. I would also like to thank the excellent support community at MatLab Central which provides open source MatLab codes and feedback on debugging the program. Finally, I would like to thank my family for all of their support throughout my academic career.

Table of contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation:

The motivation behind this project was to investigate the flow within the boundary layer region. Boundary layer is one of the most interesting subjects in Aerodynamics because it is independent of the global flow. On a very large scale, all human dwellings, including high rise building, is contained within the Earth's Atmospheric Boundary Layer. On small scale, boundary layer is very small region of flow where the velocity varies from zero to free stream. In the field of Aerodynamics, boundary layer is a fascinating subject to study due to its complex nature of attachment, detachment, and transition between laminar and turbulent. Therefore, this project will examine the complex nature of boundary layer separation and with appropriate assumptions, it will develop program that can approximate region of flow separation.

1.2 Project Objective:

The objective of this project is to develop a robust, standalone executable MatLab code that will predict separation point for a given airfoil at an angle of attack. The input parameter of the airfoil will be determined from NACA 4-series, 5-series, and from custom airfoil coordinates. Pressure distribution around the airfoil will then be determined using the Panel Method and the geometry input of the airfoil. Using the pressure distribution, the separation point on an airfoil will be determined using the following separation criteria: Falkner-Skan Solution, Stratford's Criterion for Laminar Boundary Layer and Stratford's Criterion for Turbulent Boundary Layer. The results will be verified for different airfoils to validate the code and determine the accuracy of the criteria.

1.3 Literature Review & Background Information:

When an airfoil is traveling through a flow field, it acquires a boundary layer around the surface where the viscous forces occur. This boundary layer can be laminar or turbulent. As the flow moves further downstream, the pressure gradually increases, reaching a value slightly above the free-stream pressure at the trailing edge. This region where the pressure increases in the flow direction is the region of adverse pressure gradient. Consider the motion of the fluid element in the boundary layer already being hindered by the effect of friction; in addition, it is now being hindered by increase of pressure which will further reduce its velocity. As the fluid element continues to move downstream, it may completely stop and reverse its direction and start moving backward. The figure below illustrates the station as the fluid element goes against adverse pressure gradient. [9]

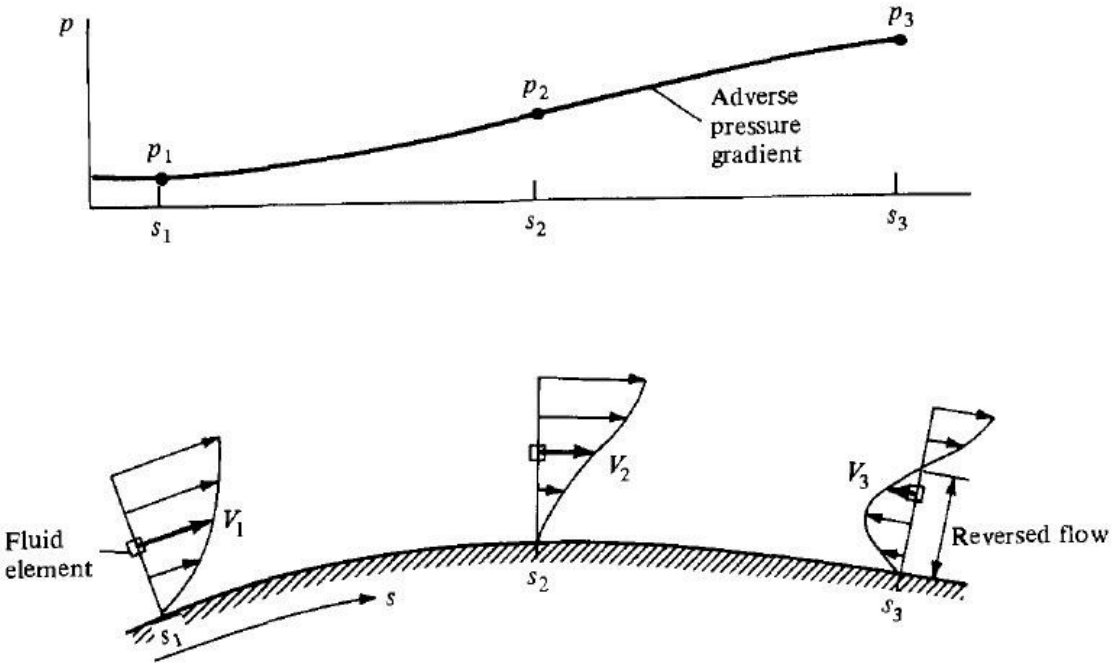


Figure 1: Flow in adverse pressure gradient [1]

The result of this reverse-flow phenomenon is to cause the flow to separate from the surface and create a wake of recirculating flow downstream of the surface as shown below. At the point (b) where the boundary layer is being separated, the shear stress is zero and occurs where dV/dn = 0 at the surface.

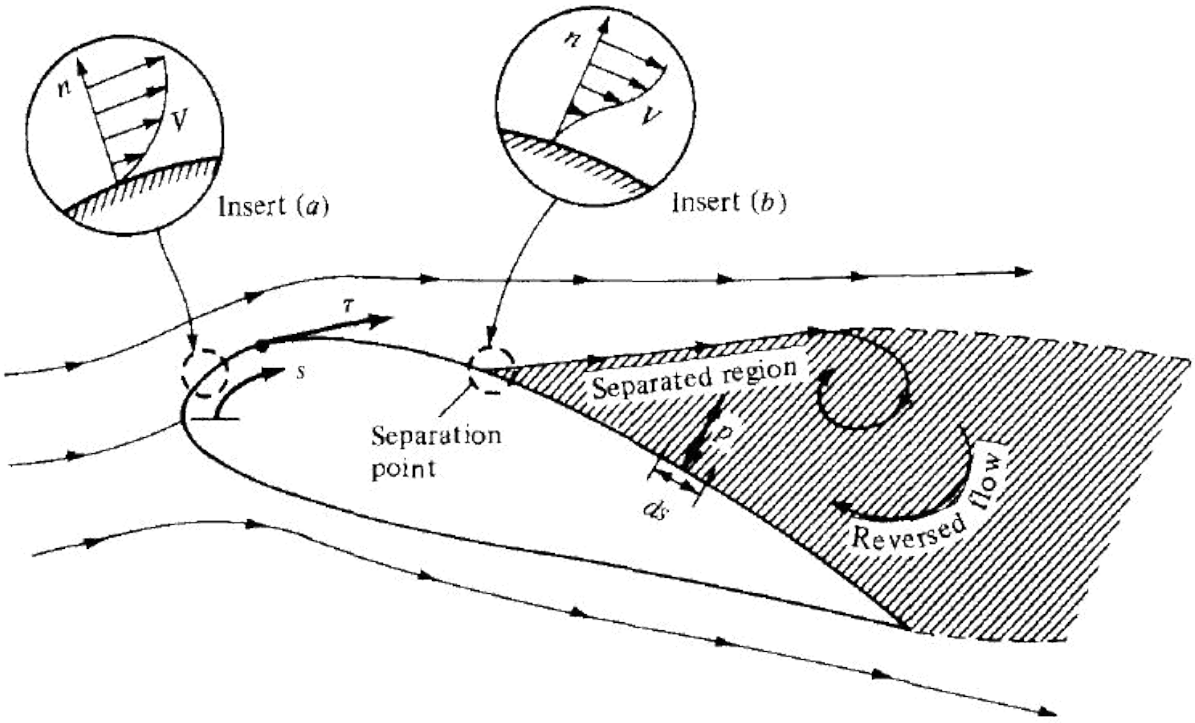


Figure 2: Effect of viscosity on a body [1]

The separation of the boundary layer causes change in the pressure distribution around the airfoil. As shown in the figure below, the separated flow has much greater pressure distribution at the top of the airfoil. This causes the airfoil to have less lift as the flow is separated. Also, in the tail section of the airfoil, the separated flow has much less pressure distribution. The horizontal component of that distribution causes the airfoil to have much greater drag component. Therefore, the separation of the boundary layer causes the airfoil to have a drastic loss in lift which causes stall, and a major increase in drag. [9]

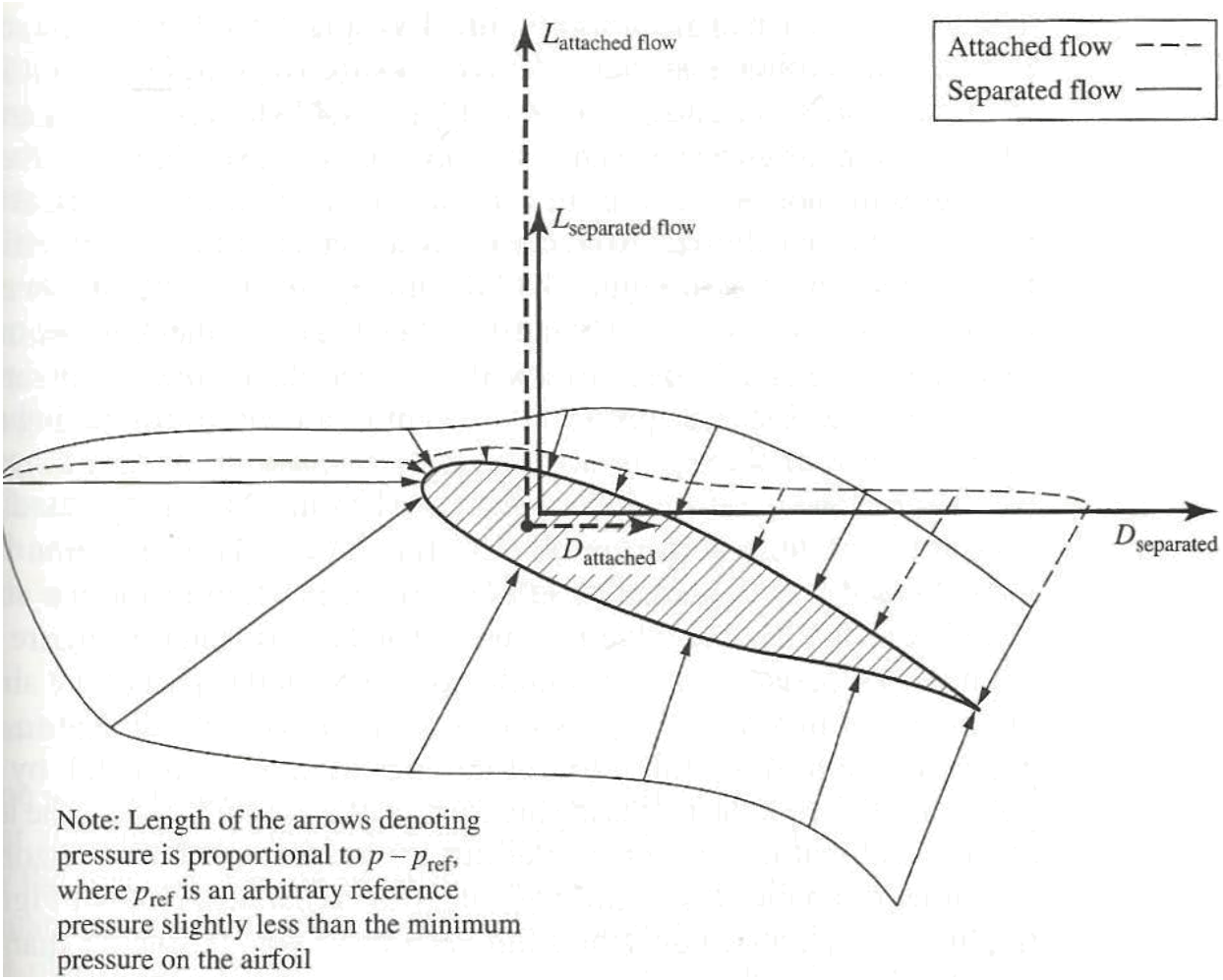


Figure 3: Pressure distribution [1]

Knowing where the flow separation occurs in different airfoils, greatly contributes to the design of the wing. A wing with single airfoil design may experience an immediate stall where wing with multiple airfoil can ensure that the tip stalls first and causes the plane to pitch down which allows the flow to reattach itself and recover from the stall.

The project "Determination of Flow Separation Point on NACA Airfoils at Different Angles of Attack by Coupling the Solution of Panel Method with Three Different Separation Criteria" was done in 2009 by San Jose State University students for Advanced Aerodynamics class. The authors created a MatLab program for NACA 4 and used panel method to determine the pressure distribution and used the Stratford and Falkner-Skan criteria to determine the separation point on the airfoil. They were successfully able to develop the code and implement the separation criteria to predict separation point on NACA 4 airfoils. Their code was benchmarked against their experimental data and showed that the Stratford's turbulent boundary

layer criterion had highest accuracy for predicting separation point on NACA airfoil. Their future work was to refine the program and extend it to predict separation on custom airfoils and make the program independent from the MatLab service. [7]

This project will extend the work done in the past and expand the geometry selection from NACA 4 series to 5; and it will include the option of importing user specified geometry. The project will also validate the program by benchmarking to various other 2D simulation programs such as JavaFoil and Xfoil.

2. Geometry

The first step in developing Airfoil Boundary Layer Separation program is to obtain the geometry of any given airfoil. NACA airfoils were designed in 1930s to 1950s by the National Advisory Committee for Aeronautics. The shape is described in the numerical code of each airfoil. The geometry can be numerically calculated by using various equations as shown in detail in this paper. [6]

2.1 NACA 4 Series:

NACA 4 series are named in the following method: NACA MPXX, where:

- M: is the maximum camber divided by 100.
- P: is the position of the maximum camber divided by 10.
- XX: is the thickness as a percentage.

To generate the NACA 4 series airfoil, we must first determine the equation of the camber line. Camber line is the line that is equidistance from the top and bottom surface of the airfoil. For symmetric airfoils, that line would be the horizontal chord line. For cambered airfoil, that line has an equation for before the max camber point and after the max camber point. In NACA 4 series, the camber equations are given in the figure below.

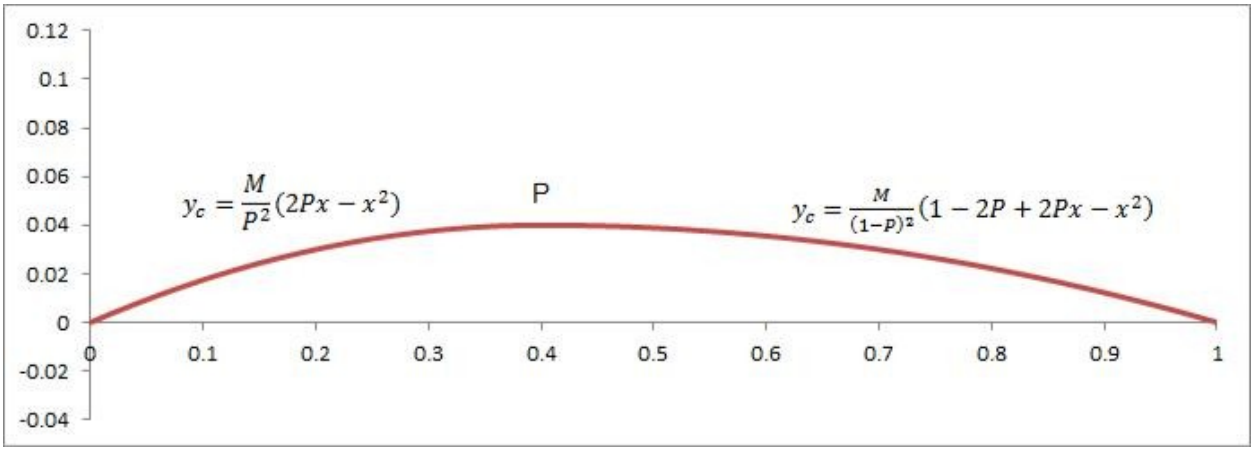


Figure 4: Camber line NACA 4 series

Once we have the front and back camber equations, we must identify camber gradient. This is necessary since the thickness of the airfoil is given as perpendicular distance from the tangent of the camber line. Therefore derivative of the camber equation will provide the gradient of the camber as shown in the figure below.

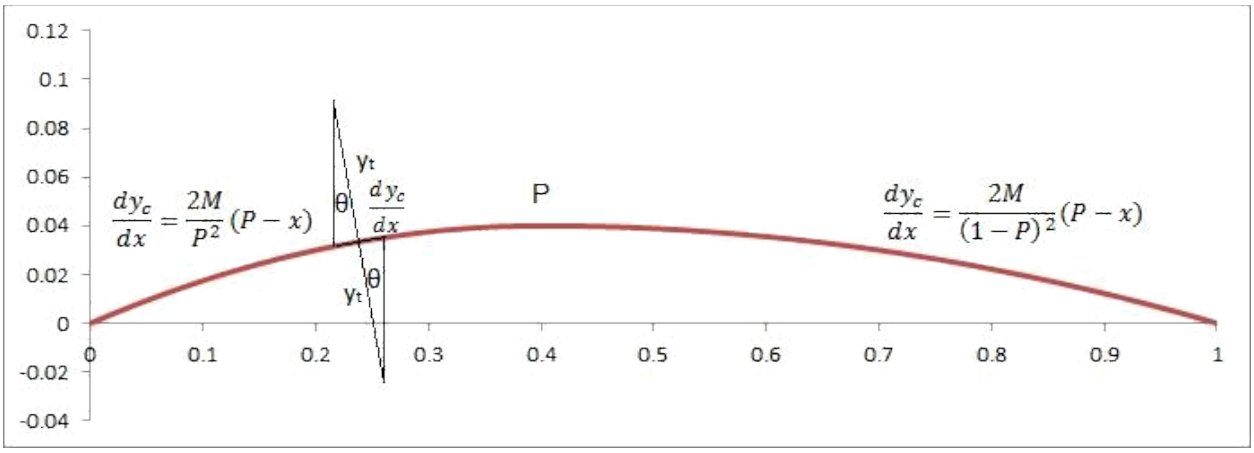


Figure 5: Gradient camber equation.

Thickness distribution equation for all NACA airfoil series is same. The constants of the equations are calculated for 20% airfoil thickness and then readjusted into the equation as a variable. The a_4 constant determines if the airfoil needs to have closed trailing edge or open trailing edge. For numerical purposes, it is necessary to have closed trailing edge to use the panel method for pressure distribution calculations. The thickness distribution equation is given in the figure below.

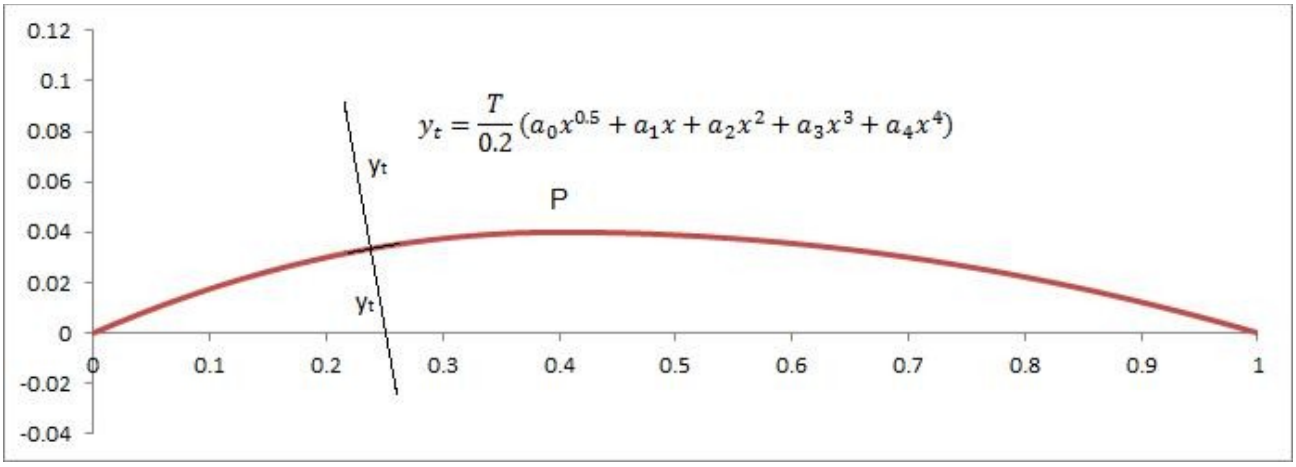


Figure 6: Thickness equation

The program requires x and y coordinate of the top and bottom of the airfoil. Therefore, the equation of the camber gradient is used to obtain the angle theta as shown in the figure below. The equations below identify coordinate change used to obtain the

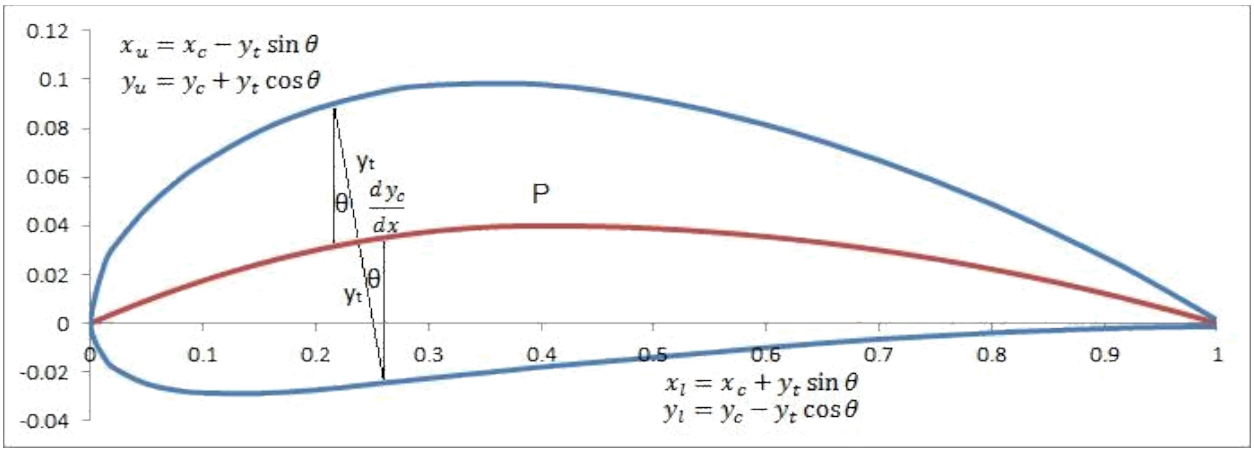


Figure 7: Geometry of NACA 4 series

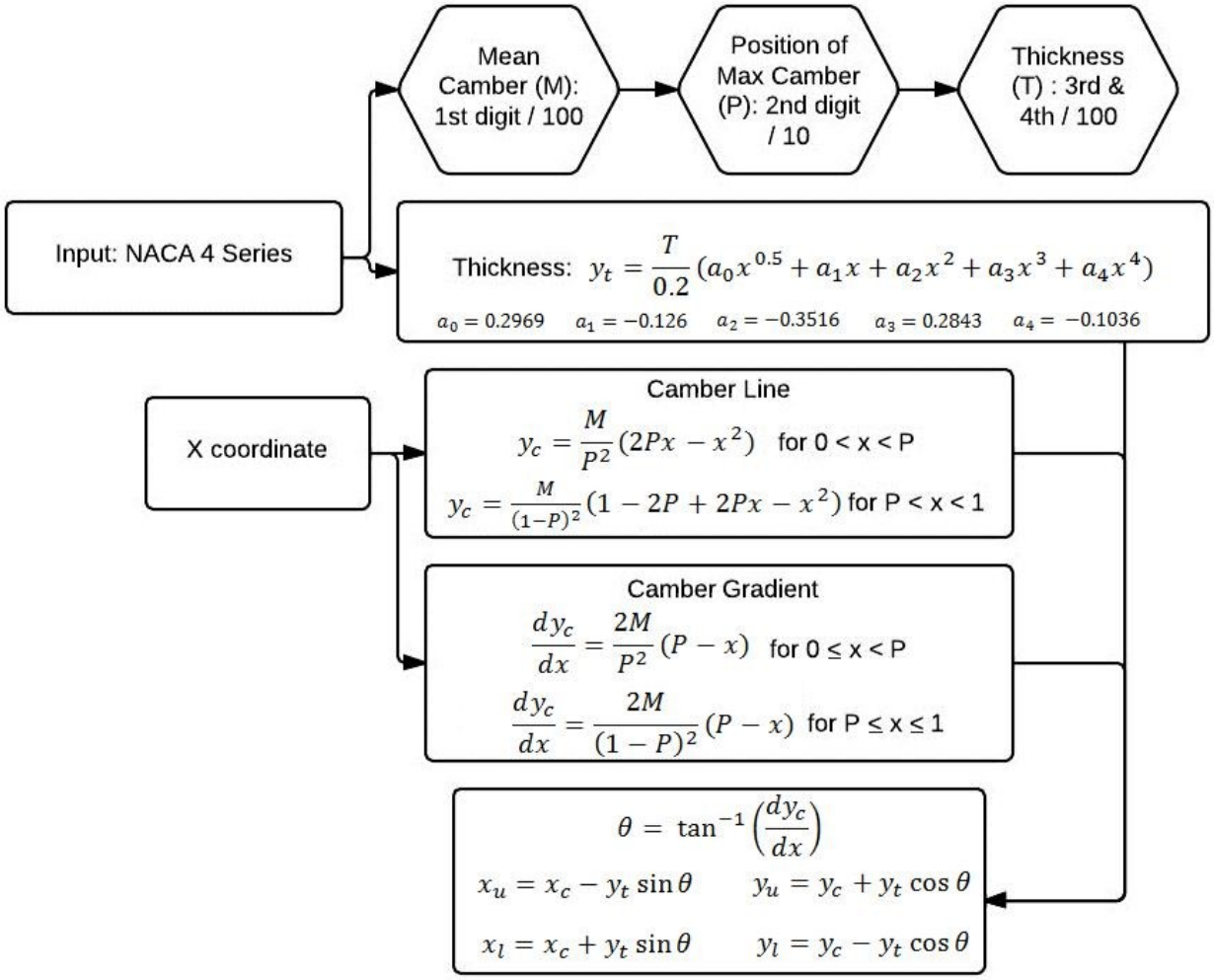


Figure 8: Flowchat describing NACA 4 series geometry

The flowchart above shows the logical process the program will use to obtain the geometry of the NACA 4 series airfoil. It will first consider the NACA 4 series values entered by the user. From that, it will determine the value of M, P and T. The program will then go into loop

process and for every chord value from o to P, it will calculate camber line values, camber gradient values and thickness values. It will then calculate the values from P to 1 using the back equations for camber and camber gradient. After that, for every chord value, it will store a value of x and y coordinates for upper and lower surfaces.

2.2 NACA 5 series:

The NACA 5 series have similar thickness distribution to the 4 series. However, they have a new camber line that allows it to concentrate camber near the leading edge. The reflexed camber line version is also designed to produce zero pitching moment. The NACA 5 series is represented by the following numbers: LPQXX, where:

- L: This digit controls the camber near the leading edge by identifying the designed coefficient of lift (Cl = 3L/20)
- P: Position of the max camber: P/20.
- Q: Identifies if the airfoil has normal camber line (0) or reflexed camber line (1). Reflexed camber line is a camber line that curves upward towards trailing edge to provide zero pitching moment.
- XX: Is thickness as percentage.

The geometry equation for NACA 5 series can be separated in to two types of airfoils: Normal Cambered and Reflexed Cambered. Once we have the camber line, the thickness equation used in the NACA 4 series can be used to obtain the x and y coordinates of the airfoil geometry.

2.2.1 NACA 5 Series: Normal Cambered: LP0QXX

The standard 5 series camber line is given in the table below [6,8]. Where m is not the position of the maximum camber but it is related to the maximum camber position by the equation [2.1.1]. The coefficient k1 is given by the following equation [2.1.2] and [2.1.3].

Table 1: NACA 5 series: Normal Cambered equations.

	Front (0≤x<m)	Back (m≤x≤1)
Camber	$= \frac{}{6}$ $(- 3 + (3 -))$	$= \frac{}{6} (1 -)$
Camber Gradient	$\frac{}{} \frac{}{}$ $- 6 + (3 -))$	$\frac{}{} \frac{}{}$ 6
$\frac{}{} = = (1 - \sqrt{} \frac{}{})$ [2.1.1]		
$= \frac{}{}, = \frac{}{}$ [2.1.2]		
$= \frac{(}{}) - \frac{}{} (1 - 2) (\frac{}{} - \sin (1 - 2))$ [2.1.3]		

2.2.2 NACA 5 Series: Reflexed Cambered: LP1QXX

The reflexed cambered 5 series is given by the equation in the table below [6,8]. Where m is given by the equation: [2.2.1]. Similarly to the normal cambered airfoil, k1 and Q are given by the equations: [2.2.2] and [2.2.3]. The k2/k1 ratio is given by the equation [2.2.4].

Table 2: NACA 5 series: Reflexed cambered equations

	(0 ≤ <)	(≤ ≤ 1)
Camber Reflexed	$= \frac{}{6}$ $((-) - \frac{}{} (1 -) - +)$	$= \frac{}{} \frac{}{6}$ $((-) - \frac{}{} (1 -) - +)$
Camber Gradient Reflexed	$\frac{}{} \frac{}{} \frac{}{}$ $= 6 (3 -) - (1 -) -$	$\frac{}{} \frac{}{} \frac{}{}$ $= 6 ((-) - (1 -) -$

[2.2.1]
[2.2.2]
[2.2.3]
[2.2.4]

Once the cambered equation is identified, we can follow the method established in the NACA 4 series to obtain the thickness equation and ultimately obtain the x and y coordinates of the NACA 5 series airfoil. Example below shows the calculated NACA 5 series airfoil and how it relates to the reflexed version of that airfoil.

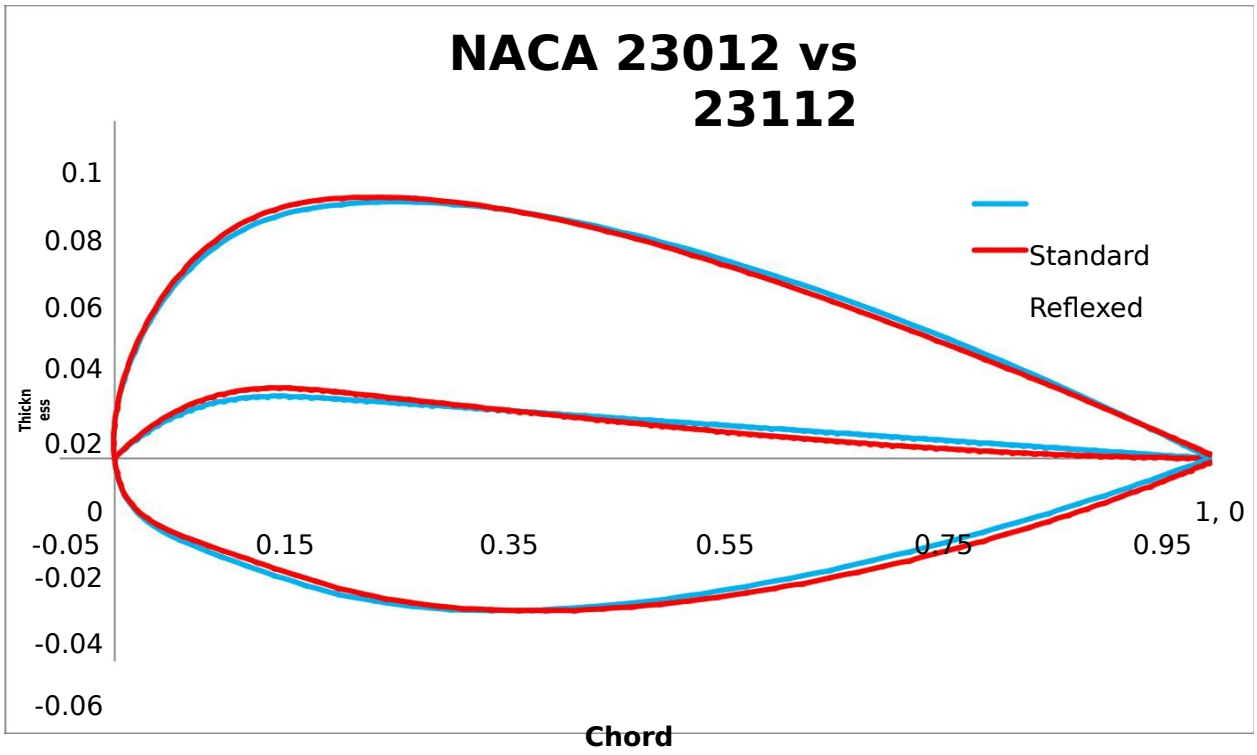


Figure 9: NACA 5 series: Normal & Reflexed cambered

2.3 Custom Airfoil:

One of the requirements for this project was to ensure that it can use custom airfoil geometry from the user and perform analysis on that geometry. MATLAB has a variety of functions that can read text files and store the information onto variables. The method used in this project is as follows:

```
load Airfoil.dat;
x = Airfoil(:,1);
y = Airfoil(:,2);
```

There are certain requirements that the input file must have inorder for it to be useful in this program:

1. The first requirement is that the file must be named “Airfoil.dat” and it must be in the same folder as the program.
2. The file must only contain the coordinates of the airfoil and any other information from the text file must be removed. If the airfoil coordinates are obtained from another source, they will include general information such as name of the airfoil and number of data points and they must be removed.
3. The file must have coordinates in two separate columns as shown below.
4. The file must have coordinates arranged properly to ensure they’re useful in creating pressure distribution since the pressure distribution method is using the panel method. The coordinates must start from the bottom trailing edge of the airfoil to the bottom leading edge and then follow the top leading edge to the top trailing edge. An example of this coordinate system is shown in the figure below.

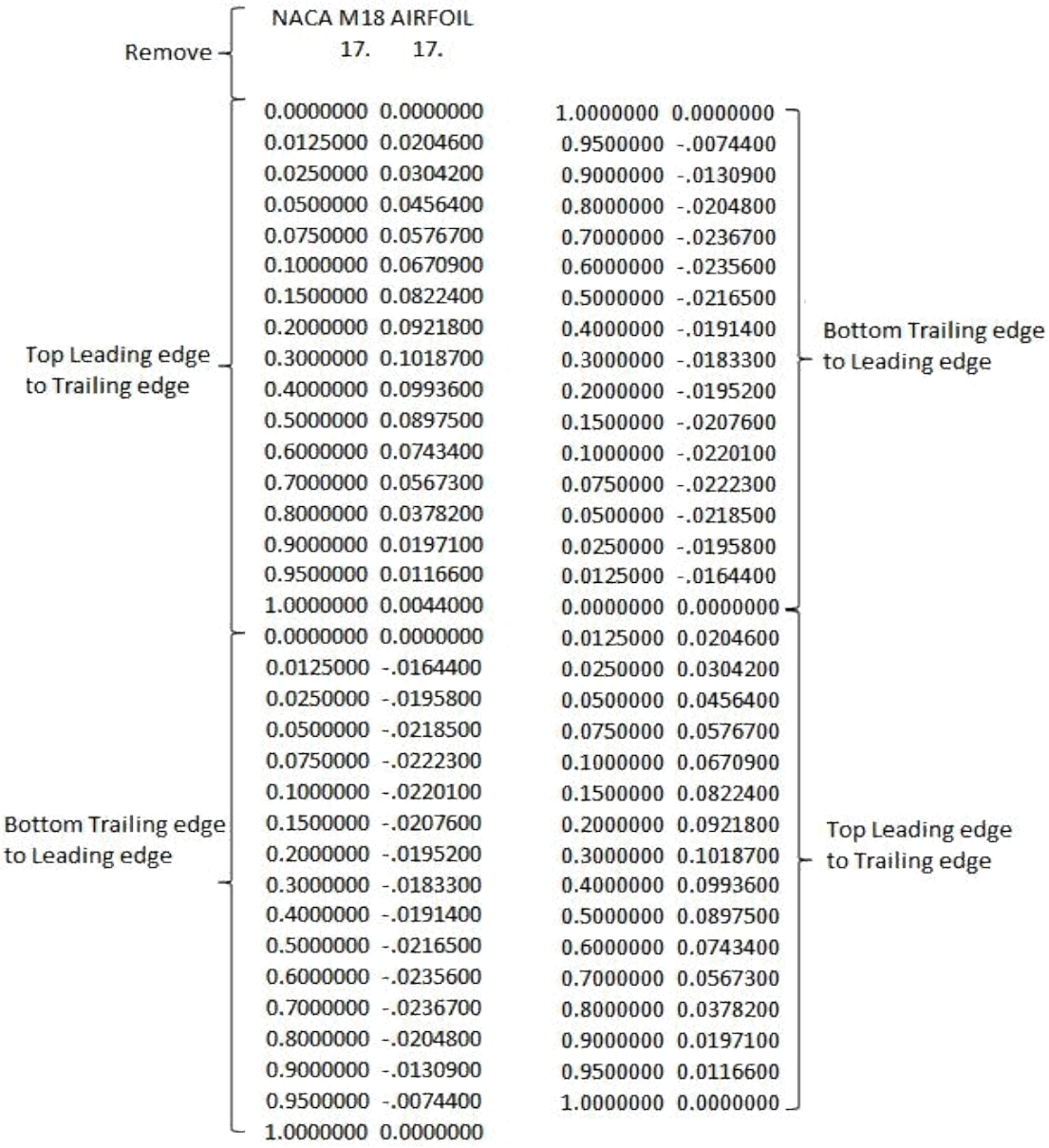


Figure 10: Example of Airfoil.dat file

3. Pressure distribution

The thin airfoil theory doesn't take into consideration of airfoil thickness and it has inaccurate pressure distribution near stagnation points. Also, airfoils with high camber and large thickness can't be used in thin airfoil theory due to the assumptions made in the derivation. Therefore, in order to provide accurate pressure distribution for a given geometry profile, we must consider Panel-Method. [5]

The Hess-Smith Panel Method was developed by Hess and Smith of the Douglas Aircraft in 1966. It uses combination of the solution of the Laplace's equation to source/sinks and vortices derived from the Potential Flow Theory to simulate the airfoil by discretizing the surface of the body into series of panels. The potential flow theory assumes inviscid, imcompressible and irrotational flow to solve Laplace's equation. [5]

$$= \quad + \quad + \qquad [3.1]$$

$$= \int ( ) \eta \qquad [3.2]$$

$$= - \int ( ) \qquad [3.3]$$

The equation above is the total potential function which contains the corresponding potential flow of the free stream velocity, source distribution and vortex distribution. The source and vortex distribution can contain varying strength throughout the airfoil. The Hess-Smith Panel Method assumes constant vortex strength over the entire airfoil and uses kutta condition to fix its value. It allows the source strength to vary from panel to panel. Together with constant vortex distribution, varying source strength and the kutta condition, the flow tangency boundary condition must be satisfied [3]. The equation one is then discretized to:

$$= \quad (\quad s + \sin) + \Sigma \qquad \int \qquad [( ) n - \quad] \qquad [3.4]$$

Using tangency boundary condition and kutta condition, the panel matrix can be set up as shown below. The tangency boundary conditions are used to determine the blue region of the matrix. The kutta condition represents the red region of the matrix. The full derivation of the process is described in reference [5].

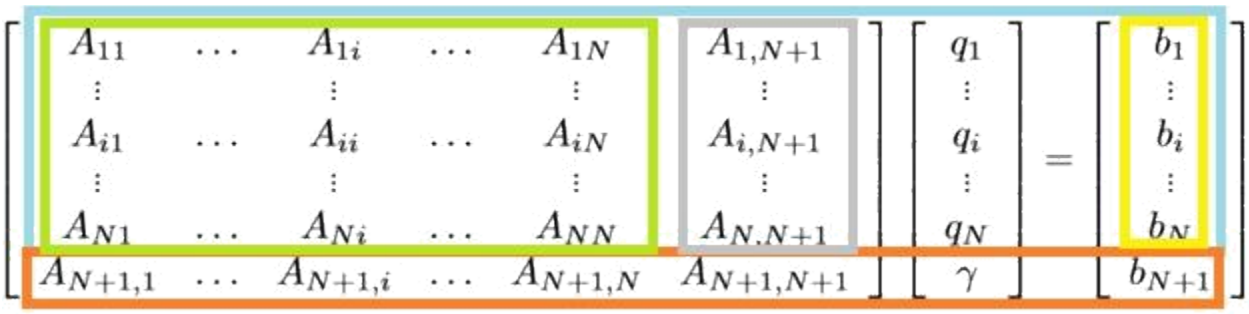


Figure 11: Matrix of the panel method [5]

The tangency condition represented by the blue region matrix is represented by the equation [3.5] stated below. The green region is represented by [3.6]. The grey region is represented by [3.7]. And the yellow region is represented by [3.8]. The r represents the distance from midpoint of panel i to the j^{th} node. β represents the angle subtended by the j^{th} panel at the midpoint of panel i.

Σ + = [3.5]

2 = sin(–) n ( ) + s(–) [3.6]

2 = ∑ s(–) n ( ) – sin(–) [3.7]

= sin(–) [3.8]

The kutta condition which represents the red region in the matrix is given by the following relationship:

Σ + = [3.9]

2 = ∑ sin(–) – s(–) n [3.10]

2 = ∑∑ sin(–) n ( ) + s(–) [3.11]

= – s(–) – s(–) [3.12]

Solving the matrix will result in the value of source strength q and the vortex strength γ. From that, the tangential velocity at the midpoint of each panel can be calculated to be:

$$= \quad s(\quad - \quad) + \sum \quad \text{—}[\sin(\quad - \quad) \quad - \quad s(\quad - \quad) \, n \, \text{——} \, + \,]$$

$$\text{—} \sum \quad [\sin(\quad - \quad) \, n \quad \text{———} \, + \quad s(\quad - \quad) \quad] \quad [3.13]$$

Therefore, the pressure coefficient at each panel can also be determined to be:

$$(\overline{\ }) = 1 - \qquad [3.14]$$

The program will integrate the Hess-Smith panel method described above and derived in the reference [3] to generate the pressure distribution on the given geometry of the airfoil. Once the pressure distribution is known, it can be used to identify the region of flow separation.

4. Separation Criteria

The program will use three different separation criteria: The Falkner-Skan Solution, The Stratford's Criterion for Laminar Boundary Layer (LBL), and The Stratford's Criterion for Turbulent Boundary Layer (TBL). Each of the criteria uses pressure distribution obtained from the panel method to predict the flow separation point on the airfoil. Once the separation point has been determined, the program will plot the result on the geometry of the airfoil.

4.1 Falkner-Skan Solution:

The Blasius solution developed by H. Blasius in 1908 is valid for zero pressure gradient problems. In 1930, Falkner and Skan developed the equation for incompressible boundary layer with pressure gradient. The derivation of this equation is highlighted in the reference [1,9]. The pressure term can be added to the x-momentum by the Euler's equation to obtain:

[4.1.1]

To obtain the self-similar solution, the independent variables (x,y) were transformed to (s,η) where

[4.1.3], and the new transformed stream function is

[4.1.4]

This results in the transformed stream function and its derivatives to be in terms of η and the x-momentum becomes an ordinary differential equation:

[4.1.5]

[4.1.6]

The boundary conditions for this equation are as follows:

Where β is the boundary layer separation criterion and for β = -0.1998, the

which indicates that the laminar boundary layer has separated. During the derivation of this method it was necessary to assume the flow is inviscid, and incompressible.

4.1.1 Program Approach: Falkner-Skan Solution:

The program will take the coefficient of pressure at each location to obtain the value for [4.1.2] which will be used to obtain the result of β as shown in equation [4.1.6] at each point on the top surface of the airfoil. This β value will be compared to -0.1998 to see if the laminar boundary layer has separated. If the β value is lower than -0.1998, the program will output the x-coordinate of that location as the result for Falkner-Skan Solution.

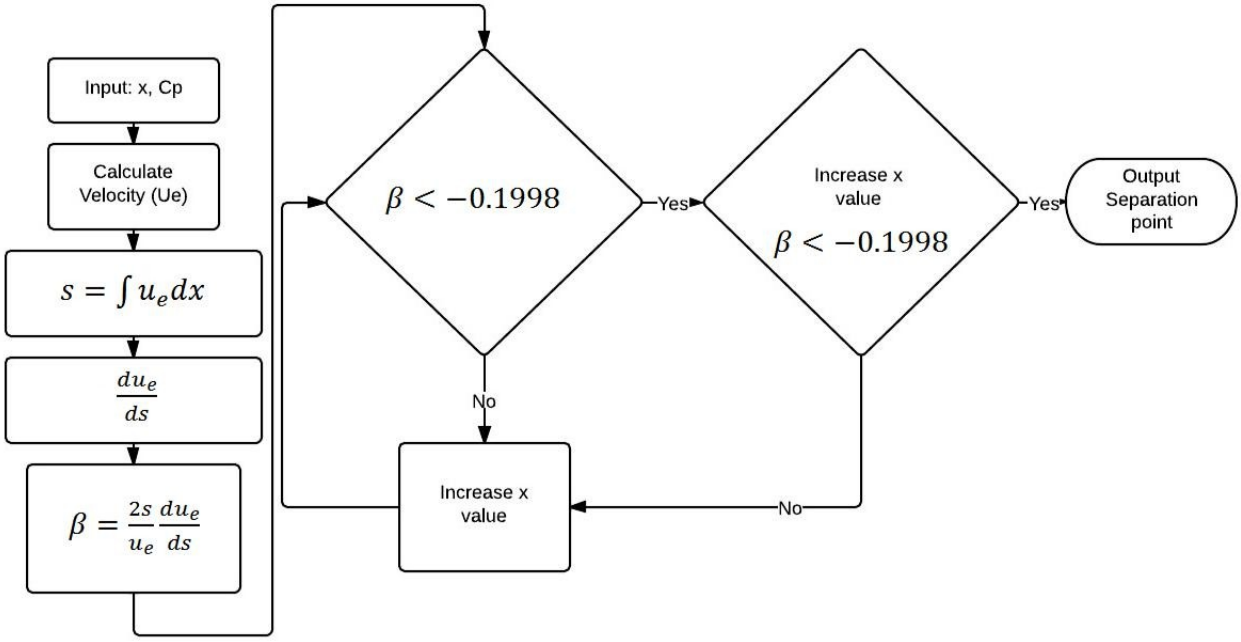


Figure 12: Flowchart of Falkner-Skan Solution in MatLab

4.2 Stratford's Criterion for 2-D laminar Separation

In 1957 B. S. Stratford derived a formula for separation of the laminar boundary layer that assumes the flow is inviscid and incompressible. The Stratford's criterion for 2-D laminar separation is given by [14,9]

$$\text{(illegible)} = 0.010 \qquad [4.2.1]$$

$$\text{(illegible)} = 1 - \text{(illegible)} \qquad [4.2.2]$$

$$\text{(illegible)} = \int \left(\text{(illegible)}\right) + \left(\text{(illegible)}\right) \qquad [4.2.3]$$

$$\text{(illegible)} = \int \left(\text{(illegible)}\right) \qquad [4.2.4]$$

(illegible) is the canonical pressure coefficient and (illegible) is effective boundary layer length. The canonical pressure coefficient varies from 1 at the stagnation point to 0 at the start of the pressure recovery region. The flow separation point occurs at x location where the [4.2.1] has value of

0.0104 or larger. If the pressure recovery region starts from the leading edge, then the effective boundary layer length is equal to the actual boundary layer length. However, in most of the problems this program will manage, there will be a region of favorable pressure gradient. In this region, the effective boundary layer length can be replaced as length of an equivalent constant pressure region with same location as the minimum pressure point as shown in [4.2.3].

4.2.1 Program Approach: Stratford's Criterion for 2-D Laminar Separation

The program will use the coefficient of pressure curve obtained from the panel method. It will first identify the location of the minimum coefficient of pressure and its x value. It will then integrate to solve equation [4.2.4] using trapezoid integration method highlighted in MatLab index. Then it will use that information along with location of minimum coefficient of pressure to obtain the value of . Then it will compute [4.2.2] for each x value. Once all required information is obtained it will compute equation [4.2.1] and compare each value starting from point of lowest coefficient of pressure to trailing edge to identify at which location the value exceeds 0.0104. It will output the x-coordinate of that value as a result.

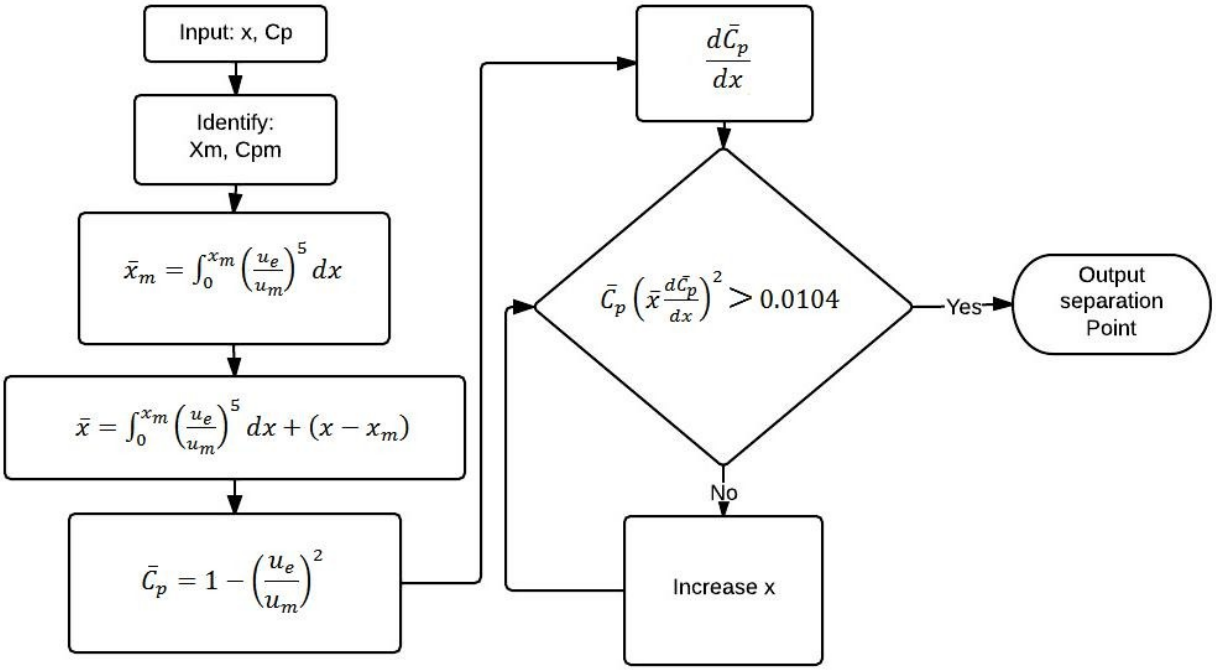


Figure 13: Flowchart of Stratford's LBL criterion in MatLab

4.3 Stratford's Criterion for 2-D Turbulent Separation:

The Stratford's criterion for turbulent separation is difficult to compute without additional assumptions due to the complexity of the equation. The general form of the equation is given by [14,9]:

[4.3.1]

Where β is a function of the shape of the pressure distribution in the region near the separation point. Upon further simplification and assuming N=0.53, beta =0.66 or 0.69; the right side of the equation can be simplified to:

[4.3.2]

Similar to the laminar criterion, the equation for and are given below.

[4.3.3]

Assuming we have a turbulent boundary layer from the stagnation point, then the would be determined by:

[4.3.4]

[4.3.5]

It should be noted that the criterion is invalid if is greater than 4/7.

4.3.1 Program Approach: Stratford's Criterion for 2-D Turbulent Separation:

The program will use the values of x and coefficient of pressure obtained from the panel method. It will first determine the location of minimum x value and the minimum coefficient of pressure. It will then determine by integrating equation [4.3.4]. It then computes equation [4.3.3] and [4.3.2]. The program then compares equation [4.3.2] to either 0.35 or 0.39 by checking if the pressure recovery region is concave or convex. It will also compare equation [4.3.2] to 4/7 to ensure if the solution obtained is valid.

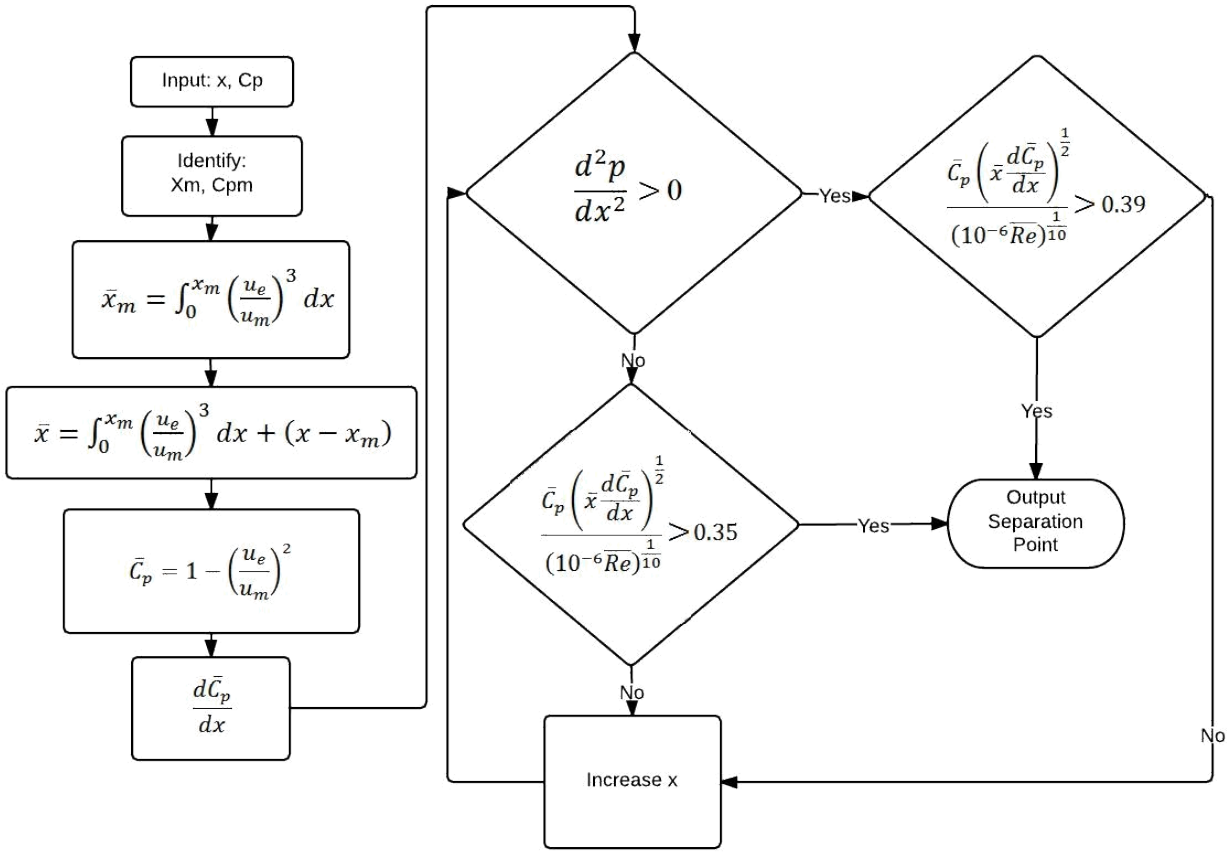


Figure 14: Flowchart of Stratford's TBL criterion in MatLab

5. MatLab Set Up and Integration

MatLab is a numerical computing environment that contains various built in functions to numerically solve mathematical expressions. Therefore it is can serve as an ideal platform for this project. The program generated in 2013a version of MatLab requires MatLab Complier Runtime (MCR) version 8.1 in order to execute the program. MCR can be downloaded for free at http://www.mathworks.com/products/compiler/mcr/. The Graphic User Interface for the program is shown in the figure below. The program is created to take the geometry as input value and it will output the flow separation region for various criteria and it will display the pressure distribution of the airfoil as well.

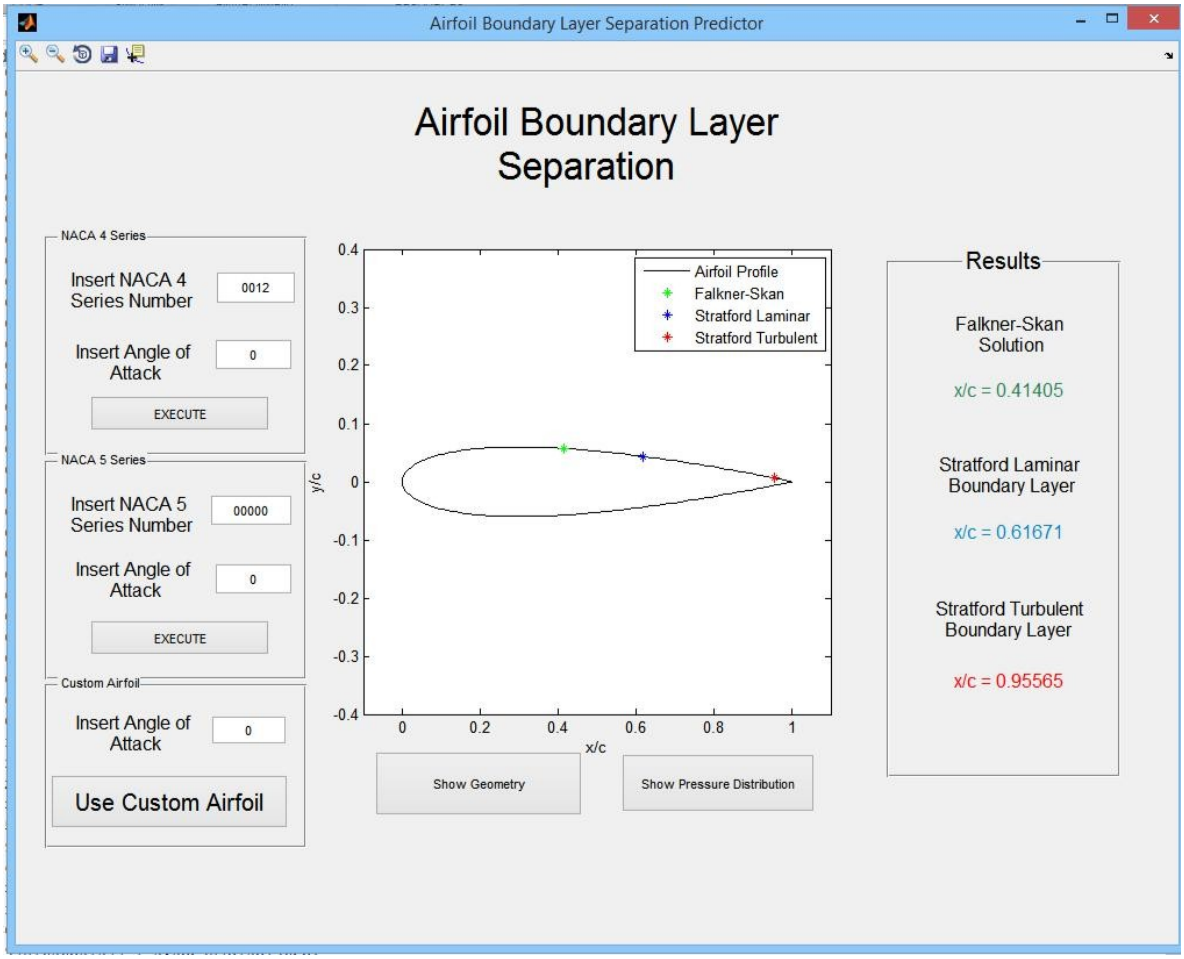


Figure 15: Example of the program's GUI

6. Results & Validation

The results obtained in this program will be validated using various methods. First, the geometry of NACA airfoils and custom airfoil used in the program needs to be validated with AirfoilTools.com. Second, the pressure distribution obtained by the panel method needs to be validated by comparing it to JavaFoil. Then the results obtained in this program from separation criteria will be compared to the results from Xfoil.

6.1 Geometry Validation:

The geometry obtained from NACA 4 and 5 series are shown above. The exact coordinates in MATLAB differs from the ones created by AirfoilTools. However, due to the exact nature of the equations used for the geometry of NACA series, the resultant geometry is nearly identical. Similar to AirfoilTools, the MatLab program created uses cosine method to generate more points at the leading and trailing edge of the airfoil. It can also be shown that the 201 points generated to create the airfoil geometry is sufficient enough to capture the airfoil profile. [10, 11]

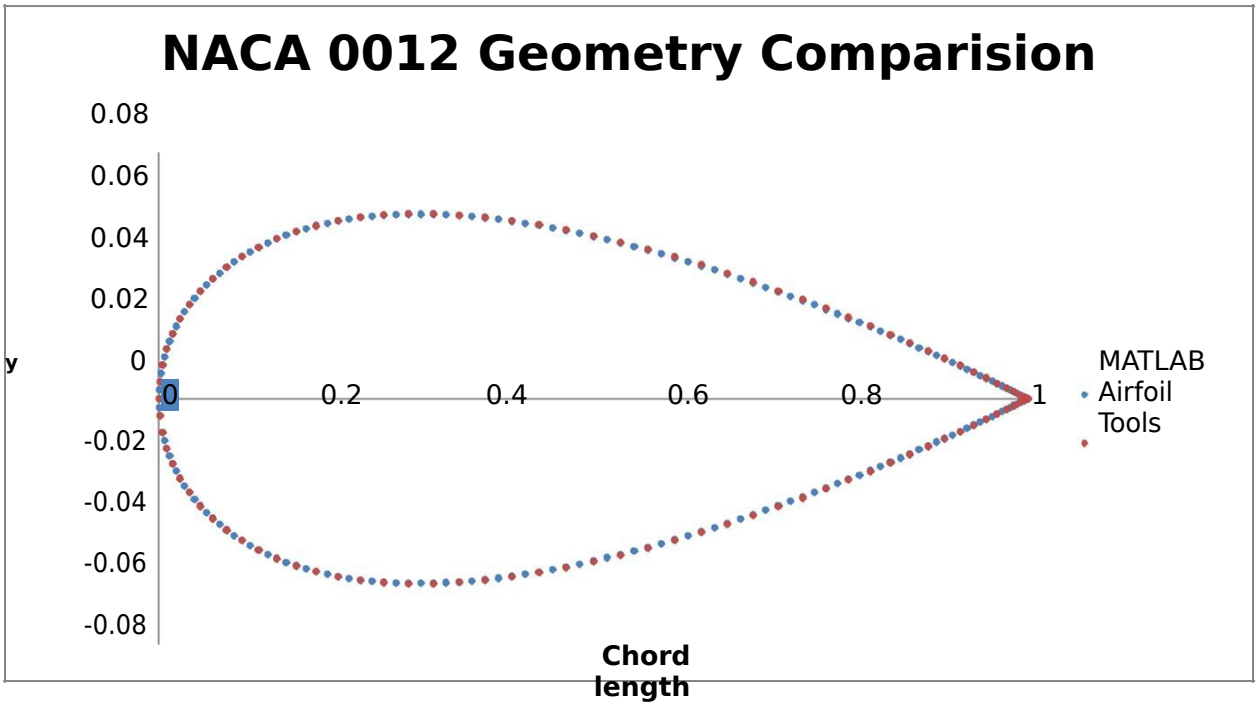
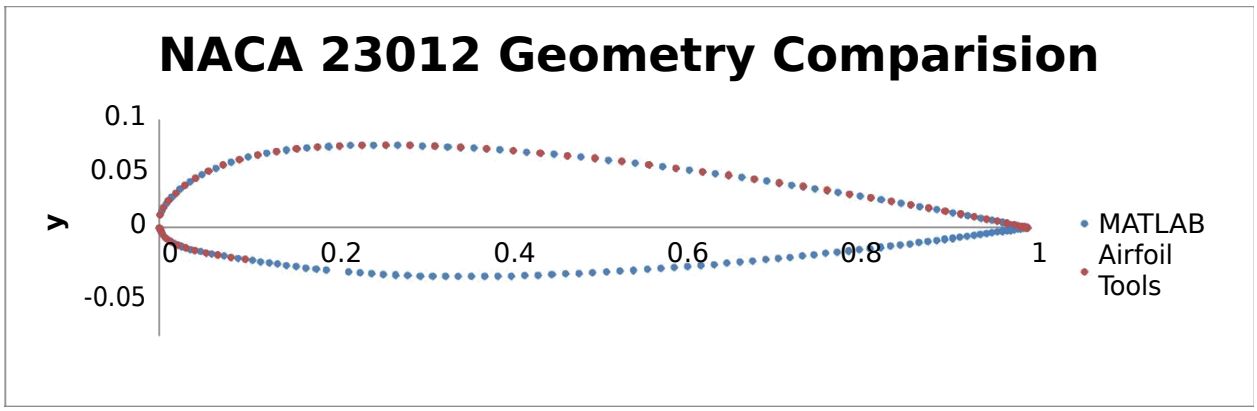


Figure 16: NACA 0012 geometry: MatLab vs AirfoilTools.com


NACA 23012 Geometry Comparision
0.1
0.05
0
-0.05
y
0
0.2
0.4
0.6
0.8
1
MATLAB
Airfoil
Tools

-0.1

Chord length

Figure 17: NACA 23012 Geometry: MatLab vs AirfoilTools.com

6.2 Pressure Distribution:

JavaFoil was used to compare the pressure distribution created in MATLAB program. JavaFoil uses the classic vortex panel method to determine the linear potential flow field around single and multi-element airfoils. In comparison, MatLab uses Hess-Smith panel method to obtain the pressure distribution. Both method uses numerical analysis to solve (# panel +1)^2 matrix to obtain the pressure distribution. The results are compared in the graph below. Figure [18] is for NACA 0012 airfoil with zero degree angle of attack and figure [19] is NACA 0012 airfoil with 10 degree angle of attack. The pressure distribution obtained by MatLab is very close to the results from JavaFoil. Therefore, the panel method used in the coding of this program provides accurate results that can be used to predict the flow separation point. [4]

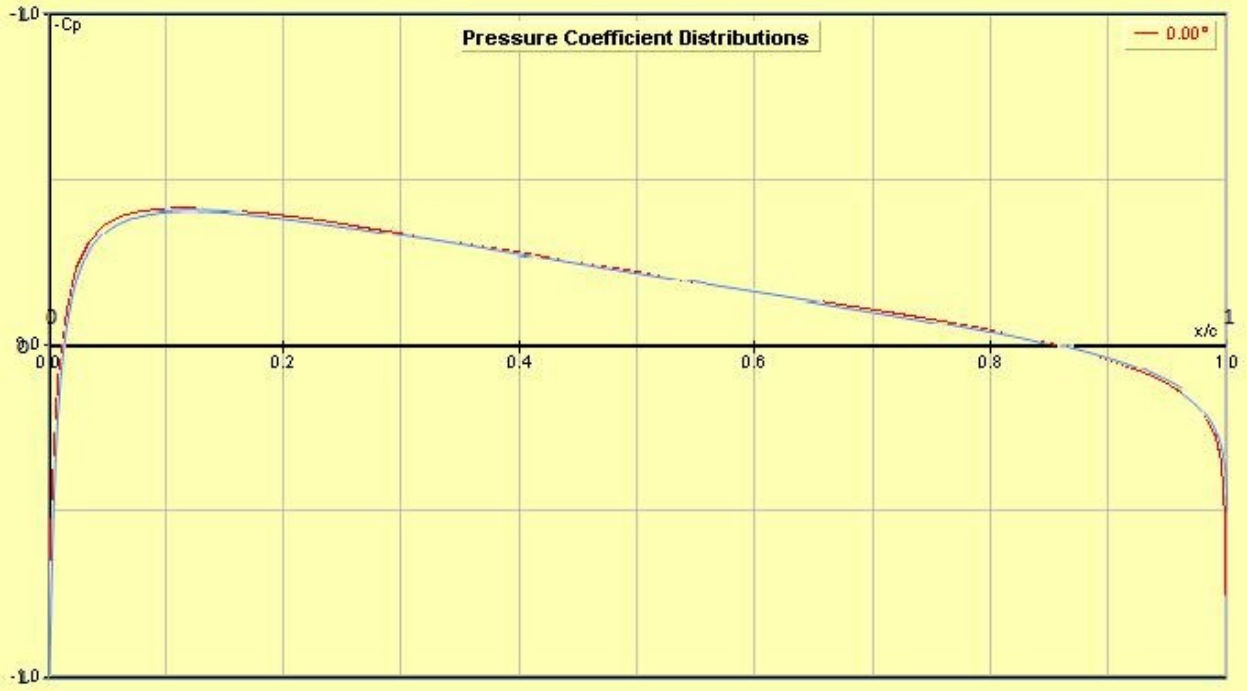


Figure 18: NACA 0012 Pressure Distribution: Matlab vs JavaFoil, AoA 0

Figure 19: NACA 0012 pressure distribution: MatLab vs JavaFoil, AoA = 10

6.3 Flow Separation:

In order to provide accurate benchmark for flow separation, the results from MATLAB program were compared with Xfoil at various Reynolds numbers. Xfoil is a 2-D interactive program that uses given geometry or airfoil number to analyze subsonic, incompressible flow. For a given airfoil and Reynolds number Xfoil calculates the pressure distribution. It has the option of solving viscous boundary layer numerically. From the resolved boundary layer, Xfoil can plot the coefficient of friction vs chord length and the flow separation point can be determined when Cf curve is less than or equal to zero. For NACA 0012, viscous boundary layer analysis was performed using Xfoil at various angles of attack and Reynolds number vary at $1x10^5$, $3x10^5$ and $5.5x10^5$. [3]

Table 3: Results from Program and Xfoil

AoA	Falkner - Skan	Stratford's LBL	Stratford's TBL	Xfoil 50k	Xfoil 300k	Xfoil 550k
0	0.41405	0.61671	0.95565	0.68493	.705119	1
1	0.353	0.53922	.94896	0.521678	.635656	1
2	0.29427	0.44514	.94896	0.427753	0.55754	1
3	0.22552	0.38329	.94183	0.334303	0.45660	1
4	0.15209	0.32328	N/A	0.236928	0.30888	1
5	0.073733	0.28006	N/A	0.131248	0.18214	1
6	0.032336	0.2523	N/A	0.058826	0.051501	0.049208
7	0.022162	0.23878	N/A	0.031729	0.027815	0.027217
8	0.013875	0.058172	N/A	0.020646	0.018736	0.018488
9	0.010449	0.027016	N/A	0.015646	.013559	0.013552
10	0.010449	0.017781	N/A	0.018154	0.010655	0.010725
11	0.0075061	0.0138775	N/A	0.01527	0.008525	0.008725
12	0.0075061	0.0138775	N/A	0.012226	0.00725	0.007612
13	0.0075061	0.010449	N/A	0.010431	0.006351	0.006571
14	0.007061	0.010449	N/A	0.008638	0.005875	0.006055
15	0.0050492	0.010449	N/A	0.006923	0.00545	0.005524

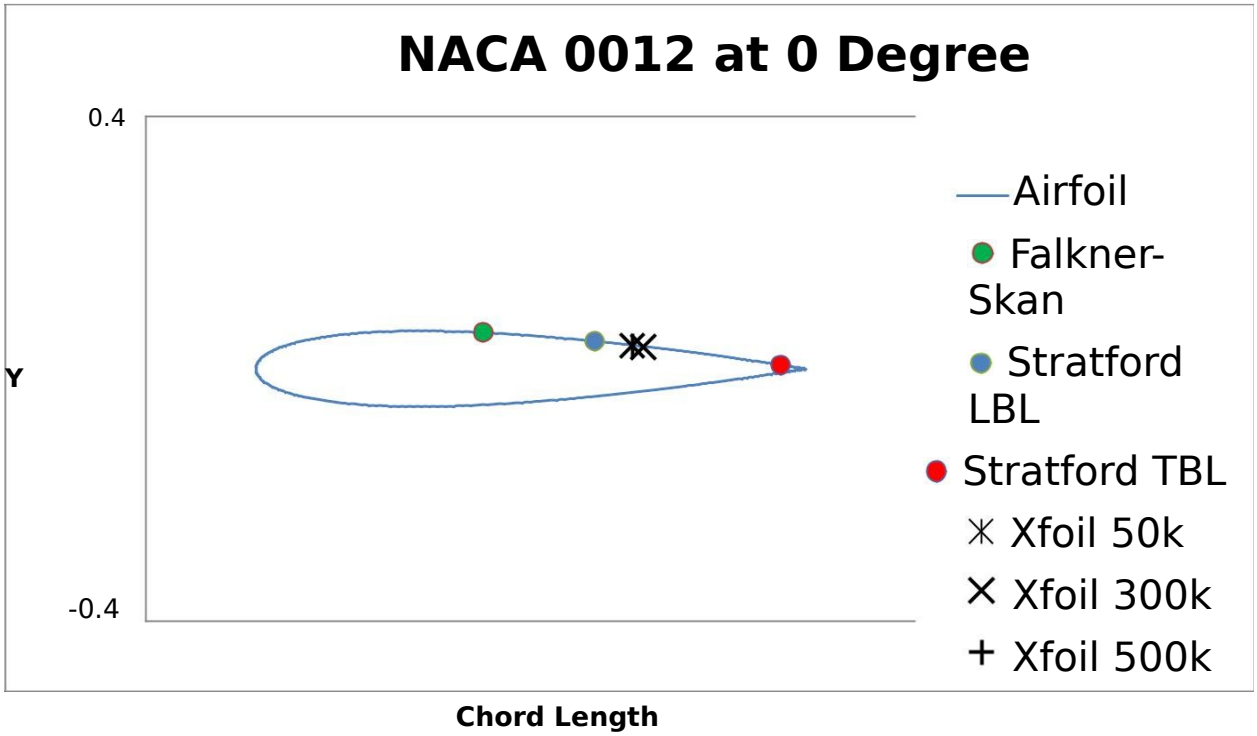


Figure 20: Results NACA 0012 at 0 degree

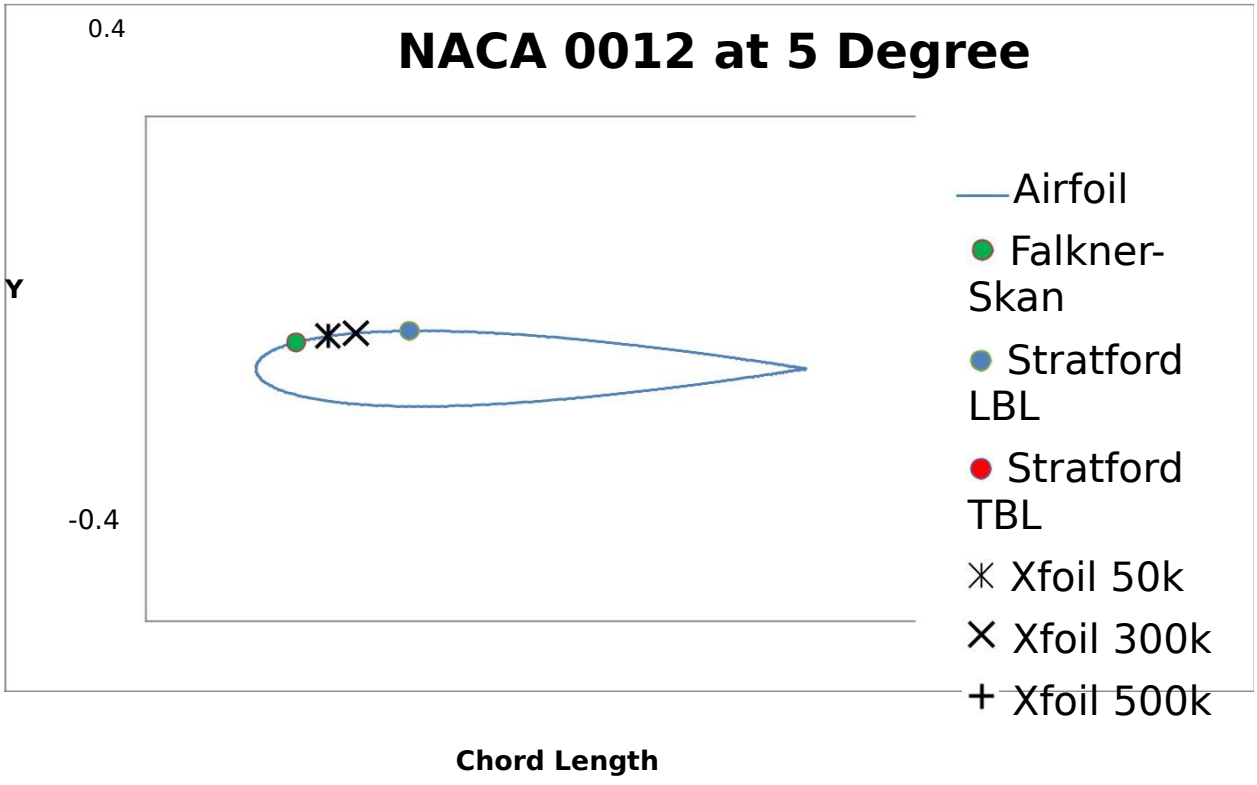


Figure 21: Results NACA 0012 at 5 degree

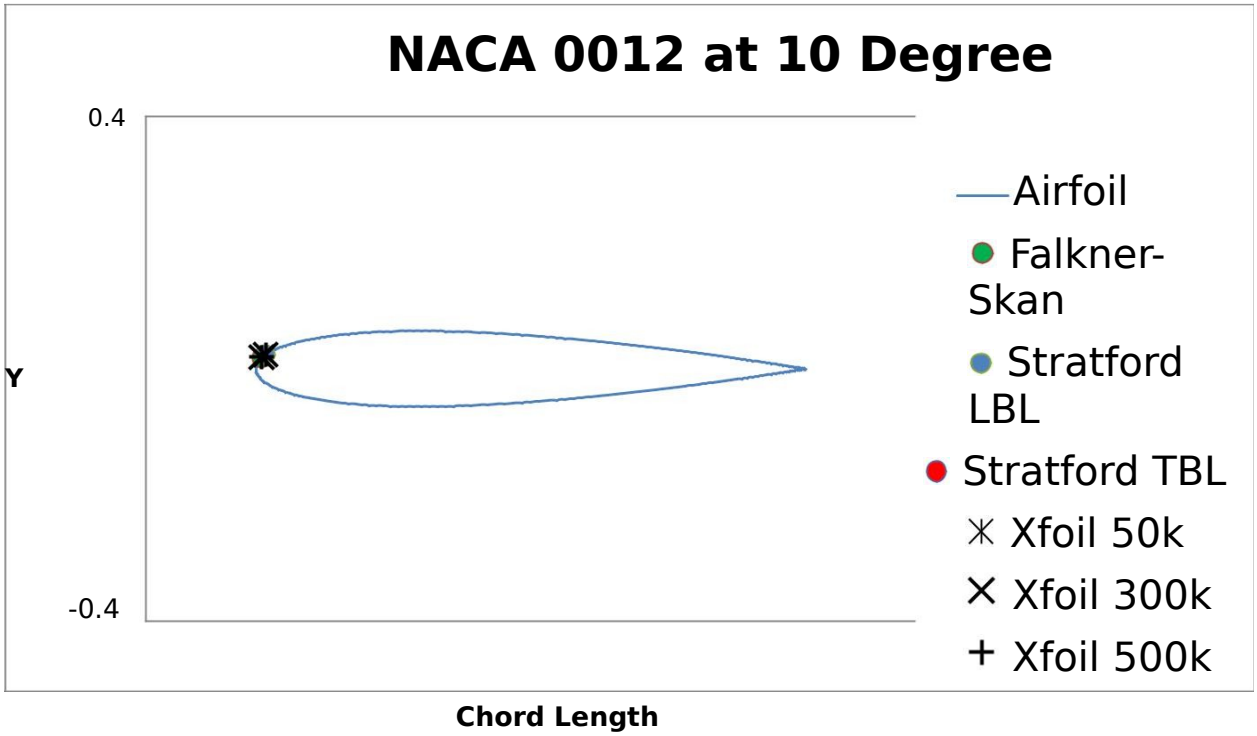


Figure 22: Results NACA 0012 at 10 degree

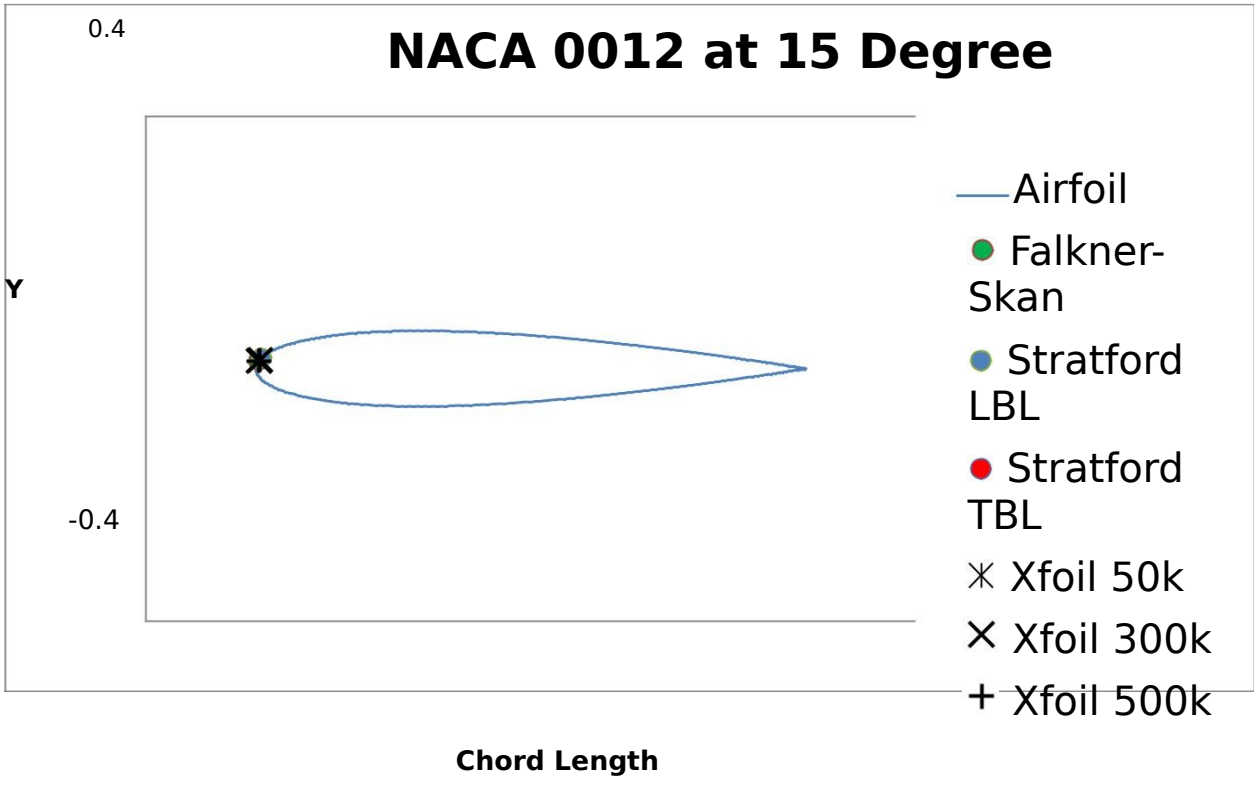


Figure 23: Results NACA 0012 at 15 degree

6.4 Results:

The results from MATLAB program and Xfoil are shown in the table above for NACA 0012. At low angles of attack of 0-3: for low Reynolds number the best criterion is Stratford's LBL criterion and for high Reynolds number the best criterion is Stratford's TBL criterion. At lower Reynolds number the results from Xfoil are very close to the results from Stratford's LBL criterion. As we increase the Reynolds number the results get closer to the Stratford's turbulent separation criterion. As we get into Reynolds number above 500,000 the results from Xfoil shows that there is no boundary layer separation. In comparison, the Stratford's criterion identifies the separation at the very trailing edge of the airfoil. The Falkner-Skan Solution in the region for low angle of attack does not represent an accurate solution to Xfoil results.

At medium angles of attack of 3-5 the Stratford's Laminar Boundary Layer criterion represents accurate result for all range of Reynolds Numbers. This result falls after Falkner-Skan Solution and before the Stratford's LBL criterion. As the Reynolds number increases, the results are closer to Stratford's laminar boundary layer criterion. At high Reynolds number, the Xfoil shows that the boundary layer is still attached. However, the Stratford's turbulent criterion is no longer valid because the is greater than 4/7 as detailed in the derivation section.

At high angles of attack of 6-10, the results from Falkner-Skan criterion show great accuracy when compared to the results from Xfoil. For both low and high Reynolds number the Falkner-Skan criterion is able to predict the flow separation point with very small margin of error.

The separation criterions used in this program will generally under predicts the flow separation region when compared to Xfoil which solve the boundary layer numerically. MatLab program generated for this project uses the pressure distribution profile obtained from Panel Method and various separation criterions to predict the flow separation. From the result comparison of NACA 0012, it is determined that for low angle of attacks the laminar flow separation point can be predicted from Stratford's LBL criterion and the turbulent flow separation point can be predicted from Stratford's TBL criterion. For high angle of attacks, the flow separation point can be predicted from Falkner-Skan Solution.

9. References

[1] Anderson, John. D. *Fundamentals of Aerodynamics*. 4. New York: McGraw-Hill, 2007. Print.

[2] "Blasius Boundary Layer." *Wikipedia*. Wikimedia Foundation, 29 Apr. 2014. Web. 13 December 2013. <http://en.wikipedia.org/wiki/Blasius_boundary_layer>.

[3] Drela, Mark, and Harold Youngren. "Xfoil." *Xfoil*. MIT. Web. 13 December 2013. <http://web.mit.edu/drela/Public/web/xfoil/>.

[4] Hepperle, Martin. "JavaFoil." *JavaFoil*. Web. 13 December 2013. <http://www.mh-aerotools.de/airfoils/javafoil.htm>.

[5] "Hess-Smith Panel Method." *Stanford*. Web. 13 December 2013. <http://canal.etsin.upm.es/CFDWORKSHOP/lect3-4.pdf>.

[6] Ladson, Charles, Cuyler Brooks, and Acquilla Hills. "Computer Program to Obtain Ordinates for NACA Airfoils." *NASA Technical Momorandum 4741*. Web. <https://www.unibw.de/lrt13_1/lehre/xtras/profildaten/nasa-tm-4741.pdf>.

[7] Le, H., Blackwell, T., & Dupzyk, I. (2009). *Determination of flow separation point on naca airfoils at different angles of attack by coupling the solution of panel method with three different separation criteria*. Unpublished manuscript, Aerospace Engineering, San Jose State University, San Jose, CA, .

[8] Mason, W. H. (1997). *Volume 1: Foundations and classical pre-cfd methods*. Unpublished manuscript, Department of Aerospace and Ocean Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, Retrieved from http://www.dept.aoe.vt.edu/~mason/Mason_f/CAtxtAppA.pdf

[9] Mourtos, Nikos. "Boundary Layers." *BL Notes*. San Jose State University. Web. 13 December 2013. <http://www.engr.sjsu.edu/nikos/courses/AE262/pdf/BL.notes.AE262.pdf>.

[10] "NACA 4 Digit Airfoil Generator (NACA 2412 AIRFOIL)." *NACA 4 Digit Airfoil Generator (NACA 2412 AIRFOIL)*. Web. 13 December 2013. <http://airfoiltools.com/airfoil/naca4digit>.

[11] "NACA 5 Digit Airfoil Generator (NACA24012 AIRFOIL)." *NACA 5 Digit Airfoil Generator (NACA24012 AIRFOIL)*. Web. 13 December 2013. <http://airfoiltools.com/airfoil/naca5digit>.

[12] Rudmin, D., & Benaissa, A. (2012). *Detection of laminar flow separation and transition on a naca-0012 airfoil using surface hot-films*. Unpublished manuscript, Department of Mechanical and Aerospace

Engineering, Royal Military College of Canada, Kingston, Ontario, , Available from Journal of Fluids Engineering. (Volume 135, Issue 10).

[13] Shan, H., Jiang, L., & Liu, C. (2003). *Direct numerical simulation of flow separation around a naca 0012 airfoil*. Unpublished manuscript, Department of Mathematics, University of Texas, Arlington, TX, , Available from Computers & Fluids. (Volume 34, Issue 9).

[14] Stratford, B. S. "Flow in Laminar Boundary Layer near Separation." *Aeronautical Research Council Reports and Memoranda* (1954). Web. <http://naca.central.cranfield.ac.uk/reports/arc/rm/3002.pdf>.

[15] "Stratford's Separation Criterion." *CFD-Wiki, the Free CFD Reference*. Web. 13 December 2013. <http://www.cfd-online.com/Wiki/Stratford%2527s_separation_criterion>.

APPENDIX A : MATLAB GUI CODE

```
function varargout = GUI_v1(varargin)
% GUI_V1 MATLAB code for GUI_v1.fig
%      GUI_V1, by itself, creates a new GUI_V1 or raises the existing
%      singleton*.
%
%      H = GUI_V1 returns the handle to a new GUI_V1 or the handle to
%      the existing singleton*.
%
%      GUI_V1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_V1.M with the given input arguments.
%
%      GUI_V1('Property','Value',...) creates a new GUI_V1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_v1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI_v1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_v1

% Last Modified by GUIDE v2.5 06-May-2014 14:00:53

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_v1_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_v1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback' ,  []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before GUI_v1 is made visible.
function GUI_v1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_v1 (see VARARGIN)
```

```
% Choose default command line output for GUI_v1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI_v1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI_v1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function FourDigit_Callback(hObject, eventdata, handles)
% hObject    handle to FourDigit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of FourDigit as text
%        str2double(get(hObject,'String')) returns contents of FourDigit as a
double


% --- Executes during object creation, after setting all properties.
function FourDigit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FourDigit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function FiveDigit_Callback(hObject, eventdata, handles)
% hObject    handle to FiveDigit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of FiveDigit as text
%        str2double(get(hObject,'String')) returns contents of FiveDigit as a
double


% --- Executes during object creation, after setting all properties.
function FiveDigit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FiveDigit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function N_Callback(hObject, eventdata,  handles)
% hObject    handle to N (see GCBO)
% eventdata  reserved - to be  defined in a future version of MATLAB
% handles    structure with handles and  user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of N as text
%        str2double(get(hObject,'String')) returns contents of N as a double


% --- Executes during object creation, after setting all properties.
function N_CreateFcn(hObject, eventdata, handles)
% hObject    handle to N (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in EXECUTE4.
function EXECUTE4_Callback(hObject, eventdata, handles)
% hObject    handle to EXECUTE4 (see GCBO)
% eventdata  reserved - to be defined in a  future version of MATLAB
% handles    structure  with handles and user data (see GUIDATA)
N = 100;
FourDigit = str2num(get(handles.FourDigit,'String'));

alpha = str2num(get(handles.alpha4,'String'));
global xFS;
```

```
global xSL;
global xST;
global yFS;
global ySL;
global yST;
global xU;
global CpU;
global x;
global y;
global xL;
global yL;
global CpL;
if FourDigit > 9999
    error = 'Invalid NACA 4 Series';
    set(handles.MSG, 'String', error);

elseif alpha > 90
    error = 'Invalid Angle of Attack';
        set(handles.MSG, 'String', error);

elseif alpha < 0
        error = 'Invalid Angle of Attack';
        set(handles.MSG, 'String', error);

else
    [x,y]= Airfoil4 (N,N,FourDigit);

    [xmid,ymid,Cp]=HessSmithPanel (x,y,alpha);

    [xU,yU,CpU]=UpperSurface (xmid,ymid,Cp,N);

    [xL,yL,CpL]=LowerSurface (xmid,ymid,Cp,N);
    if floor(FourDigit/1000)==0
        if (alpha ~= 0)==0
        xL =0;
        yL = 0;
        CpL = 0;
        end
    end
    [iU_FS,xU_FS] = FalknerSkan(xU,CpU);
    [iU_SL,xU_SL] = StratfordLBL(xU,CpU);
    [iU_ST,xU_ST] = StratfordTBL(xU,CpU);
    xFS=xU_FS;
    xSL=xU_SL;
    xST=xU_ST;

    yFS=yU(iU_FS);
    ySL=yU(iU_SL);
    yST=yU(iU_ST);

    FSx= num2str(xFS);
    SLx= num2str(xSL);
    STx= num2str(xST);
```

```
    set(handles.FSx, 'String', ['x/c = ',FSx]);
    set(handles.SLx, 'String', ['x/c = ',SLx]);
    set(handles.STx, 'String', ['x/c = ',STx]);
    axes (handles.axes1);
    plot(x,y,'k',xFS,yFS,'*g',xSL,ySL,'*b',xST,yST,'*r');
    axis([-.1 1.1 -.4 .4]);
    legend ('Airfoil Profile','Falkner-Skan', 'Stratford Laminar', 'Stratford
Turbulent');




    xlabel('x/c');
    ylabel('y/c' );
    error = '';
    set(handles.MSG, 'String', error);
    guidata (hObject, handles);

end




% --- Executes on button press in FigureG.
function FigureG_Callback(hObject, eventdata, handles)
% hObject    handle to FigureG (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global xFS;
global xSL;
global xST;
global yFS;
global ySL;
global yST;
global xU;
global CpU;
global x;
global y;

    axes (handles.axes1);
    plot(x,y,'k',xFS,yFS,'*g',xSL,ySL,'*b',xST,yST,'*r');
    axis([-.1 1.1 -.4 .4]);
    legend ('Airfoil Profile','Falkner-Skan', 'Stratford Laminar', 'Stratford
Turbulent');

    xlabel('x/c');
    ylabel('y/c');
    guidata (hObject, handles);
    global xFS;

% --- Executes on button press in FigureP.
function FigureP_Callback(hObject, eventdata, handles)
% hObject    handle to FigureP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
global xFS;
global xSL;
global xST;
global yFS;
global ySL;
global yST;
global xU;
global CpU;
global x;
global y;
global xL;
global CpL;


axes (handles.axes1);
plot(xU,CpU,'b',xL,CpL,'r');
set(gca, 'YDir', 'reverse');
axis auto
title('Pressure Distribution');
legend ('Top', 'Bottom');
xlabel('x/c');
ylabel('Cp');
guidata (hObject, handles);
% --- Executes on button press in EXECUTEC.
function EXECUTEC_Callback(hObject, eventdata, handles)
% hObject    handle to EXECUTEC (see GCBO)
% eventdata  reserved - to be defined in a  future version of MATLAB
% handles    structure  with handles and user data (see GUIDATA)
N = 100;
alpha = str2num(get(handles.alphaC,'String'));
global xFS;
global xSL;
global xST;
global yFS;
global ySL;
global yST;
global xU;
global CpU;
global x;
global y;
global xL;
global yL;
global CpL;
if alpha > 90
    error = 'Invalid Angle of Attack';
    set(handles.MSG, 'String', error);
elseif alpha < 0
    error = 'Invalid Angle of Attack';
    set(handles.MSG, 'String', error);
else
    [x,y,N]= AirfoilC ;
    [xmid,ymid,Cp]=HessSmithPanel (x,y,alpha);
    [xU,yU,CpU]=UpperSurface (xmid,ymid,Cp,N);
    [xL,yL,CpL]=LowerSurface (xmid,ymid,Cp,N);
    [iU_FS,xU_FS] = FalknerSkan(xU,CpU);
```

```
    [iU_SL,xU_SL] = StratfordLBL(xU,CpU);
    [iU_ST,xU_ST] = StratfordTBL(xU,CpU);
    xFS=xU_FS;
    xSL=xU_SL;
    xST=xU_ST;

    yFS=yU(iU_FS);
    ySL=yU(iU_SL);
    yST=yU(iU_ST);

    FSx= num2str(xFS);
    SLx= num2str(xSL);
    STx= num2str(xST);

    set(handles.FSx, 'String', ['x/c = ',FSx]);
    set(handles.SLx, 'String', ['x/c = ',SLx]);
    set(handles.STx, 'String', ['x/c = ',STx]);
    axes (handles.axes1);
    plot(x,y,'k',xFS,yFS,'*g',xSL,ySL,'*b',xST,yST,'*r');
    axis([-.1 1.1 -.4 .4]);
    legend ('Airfoil Profile','Falkner-Skan', 'Stratford Laminar', 'Stratford
Turbulent');

    xlabel('x/c');
    ylabel('y/c' );
    error = '';
    set(handles.MSG, 'String', error);
    guidata (hObject, handles);
end


function alphaC_Callback(hObject, eventdata, handles)
% hObject    handle to alphaC (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of alphaC as text
%        str2double(get(hObject,'String')) returns contents of alphaC as a
double



% --- Executes during object creation, after setting all properties.
function alphaC_CreateFcn(hObject, eventdata, handles)
% hObject    handle to alphaC (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in EXECUTE5.
function EXECUTE5_Callback(hObject, eventdata, handles)
% hObject    handle to EXECUTE5 (see GCBO)
% eventdata  reserved - to be defined in a  future version of MATLAB
% handles    structure  with handles and user data (see GUIDATA)

N = 100;
FiveDigit = str2num(get(handles.FiveDigit,'String'));
alpha = str2num(get(handles.alpha5,'String'));
global xFS;
global xSL;
global xST;
global yFS;
global ySL;
global yST;
global xU;
global CpU;
global x;
global y;
global xL;
global yL;
global CpL;
global yU;
 L=floor(FiveDigit/10000);
 P=floor(FiveDigit/1000)-L*10;
 R=floor(FiveDigit/100)-L*100-P*10;
 XX=FiveDigit-L*10000-P*1000-R*100;

 if FiveDigit > 99999
     error = 'Invalid NACA 5 Series: Value must not exceed 5 digits.';
     set(handles.MSG, 'String', error);
 elseif FiveDigit < 10000
     error = 'Invalid NACA 5 Series';
     set(handles.MSG, 'String', error);
 elseif R > 1
     error = 'Invalid NACA 5 Series: Third Digit Must be 0 or 1.';
     set(handles.MSG, 'String', error);
 elseif alpha <0
      error = 'Invalid Angle of Attack.' ;
     set(handles.MSG, 'String', error);
 elseif alpha > 90
      error = 'Invalid Angle of Attack.' ;
     set(handles.MSG, 'String', error);
 elseif P>5
     error = 'Invalid NACA 5 Series: Second Digit Can Not Exceed 5.';
     set(handles.MSG, 'String', error);

 else
    [x,y]= Airfoil5 (N,N,FiveDigit);
    [xmid,ymid,Cp]=HessSmithPanel (x,y,alpha);
    [xU,yU,CpU]=UpperSurface (xmid,ymid,Cp,N);
    [xL,yL,CpL]=LowerSurface (xmid,ymid,Cp,N);

    [iU_FS,xU_FS] = FalknerSkan(xU,CpU);
    [iU_SL,xU_SL] = StratfordLBL(xU,CpU);
```

```
    [iU_ST,xU_ST] = StratfordTBL(xU,CpU);
    xFS=xU_FS;
    xSL=xU_SL;
    xST=xU_ST;

    yFS=yU(iU_FS);
    ySL=yU(iU_SL);
    yST=yU(iU_ST);


    FSx= num2str(xFS);
    SLx= num2str(xSL);
    STx= num2str(xST);

    set(handles.FSx, 'String', ['x/c = ',FSx]);
    set(handles.SLx, 'String', ['x/c = ',SLx]);
    set(handles.STx, 'String', ['x/c = ',STx]);
    axes (handles.axes1);
    plot(x,y,'k',xFS,yFS,'*g',xSL,ySL,'*b',xST,yST,'*r');
    axis([-.1 1.1 -.4 .4]);
    legend ('Airfoil Profile','Falkner-Skan', 'Stratford Laminar', 'Stratford
Turbulent');

    xlabel('x/c');
    ylabel('y/c');
   error = '';
   set(handles.MSG, 'String', error);
   guidata (hObject, handles);
end

function alpha5_Callback(hObject, eventdata, handles)
% hObject    handle to alpha5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of alpha5 as text
%        str2double(get(hObject,'String')) returns contents of alpha5 as a
double


% --- Executes during object creation, after setting all properties.
function alpha5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to alpha5  (see GCBO)
% eventdata  reserved - to be  defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function alpha4_Callback(hObject, eventdata, handles)
% hObject    handle to alpha4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of alpha4 as text
%        str2double(get(hObject,'String')) returns contents of alpha4 as a
double


% --- Executes during object creation, after setting all properties.
function alpha4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to alpha4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in inst.
function inst_Callback(hObject, eventdata, handles)
% hObject    handle to inst (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
msgbox('Operation Complete');
```