INVERSE DESIGN OF AIRFOILS USING A FLEXIBLE MEMBRANE METHOD

by

KAMON THINSURAT

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN AEROSPACE ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2010

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Brian H. Dennis who gave me the knowledge of Computational Fluid Dynamic which importantly helps me to conduct my thesis. He also gave me lots of useful knowledge about programming and using commercial software, FLUENT which I had never used before. Not only for my thesis but that knowledge is also very valuable for me in the future work. Without his support and guidance, this thesis would not been accomplished.

I also would like to thank Dr. Don Wilson and Dr. Zhen Xue Han for spending time to be my thesis committee. Thank to their comments and suggestions for my thesis as well.

I wish to thank my friends in CFD lab who help me passed through problems using Linux operation in the lab and helping me about English language in my thesis. Without them, my life would be much harder to get through those problems. I would like to thank all of my Thai friends who always encourage me during the time I was conducting my thesis. Without them in the US, my life would not been meaningful and happy as I always am.

Lastly, I particularly thank my parents and my family who always give love ,and encouragement to me. Especially, thank my parents who have been supporting for all of my studying expense. Without their attentiveness, my life would not been great like this.

November 24, 2010

ABSTRACT

INVERSE DESIGN OF AIRFOIL USING
A FLEXIBLE MEMBRANE METHOD

KAMON THINSURAT, M.S.

The University of Texas at Arlington, 2010

Supervising Professor: Brian Dennis

The Modified Garabedian Mc-Fadden (MGM) method is used to inversely design airfoils. The Finite Difference Method (FDM) for Non-Uniform Grids was developed to discretize the MGM equation for numerical solving. The Finite Difference Method (FDM) for Non-Uniform Grids has the advantage of being used flexibly with an unstructured grids airfoil.

The commercial software FLUENT is being used as the flow solver. Several conditions are set in FLUENT such as subsonic inviscid flow, subsonic viscous flow, transonic inviscid flow, and transonic viscous flow to test the inverse design code for each condition. A moving grid program is used to create a mesh for new airfoils prior to importing meshes into FLUENT for the analysis of flows.

For validation, an iterative process is used so the Cp distribution of the initial airfoil, the NACA0011, achieves the Cp distribution of the target airfoil, the NACA2315, for the subsonic inviscid case at M=0.2. Three other cases were carried out to validate the code. After the code validations, the inverse design method was used to design a shock free airfoil in the transonic condition and to design a separation free airfoil at a high angle of attack in the subsonic condition.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

1.1 Background

Inverse design method has become an effective tool for airfoil design after Computational Fluid Dynamics field had been developed. Before the inverse approach, direct method has to be used in order to design airfoils by evaluating the performance of an actual geometry and modify it with either empirical rules or their own experience to obtain the target shape. Doing direct approach is time consuming and inefficient [1]. Inverse approach is a more powerful method where the airfoil shape is iteratively modified to achieve a certain objective pressure distribution. The inverse method needs much lesser cost and time to design a desired airfoil corresponding to the target Cp distribution [2]. There are two main types of inverse airfoil design: decoupled and coupled techniques [2, 3].

The coupled technique is the technique that simultaneously finds the solution of the flow-field and the unknown part of the boundary [2, 5]. The examples of the coupled technique are the indirect transpiration technique [18, 19] ,which obtains the target shape from exchanging no-slip wall boundary conditions, stream-function-as-coordinate approaches, characteristic boundary condition approaches, and adjoint operator/control theory approaches. Ashrafizadeh et al. [6] demonstrated the successful case of using coupled technique to design a 2-D ducts using flexible string algorithm. On the other hand, decoupled technique does not require the modification to a flow-field with the physical boundaries and iteration is needed instead [2]. Flow analysis is separated from the inverse function; therefore, any commercial flow solver software or data from wind tunnel can be easily used to design airfoils. Flexible Membrane Technique is one of decoupled technique, airfoil surface is considered as a membrane which will change the shape when the pressure distribution applied on the surface.

Flexible Membrane technique has an MGM equation as an inverse function to calculate the shape update of the airfoil. MGM equation [7, 8, 9] was developed by Garabedian and McFadden. The MGM equation is an ordinary differential equation; therefore, it is easy to implement. The input of the equation is Cp distribution and the output is the updated shape of an airfoil. Flexible Membrane technique can be widely applied to any design perspectives. For example, Wu et al. [13] used this method to design multipoint inverse shape which can effectively use in various flight conditions.

Not only in airfoil design field but any other fields also use the Inverse design effectively. For example, Bonaiuti et al. [17] applied inverse design method for determining the effect of designed pump parameters to pump efficiency so they can reduce head loss in pump machine. Henriques et al. [12] applied inverse design method for designing a new turbine blade section which has increased maximum lift and reduced the adverse pressure gradient on the suction side.

1.2 Objective

This thesis is to inversely design airfoils using flexible membrane technique and choosing FLUENT as a flow solver. The flexible membrane technique is modified in order to apply to the moving grid program which required a non-uniform grid airfoil; therefore, Finite Difference Method for non-uniform grid need to be developed to discretize to flexible membrane equation. The techniques can be applied in both subsonic and transonic cases.

1.3 Thesis Overview

The theories which are needed will be presented in Chapter 2, including Flexible Membrane Technique, Finite Difference Method for Non-Uniform Grid, NACA 4-digit series, Thomas's Algorithm and mesh generating program which is moving grid program. After that, the methodology will be presented in Chapter 3. Chapter 3 shows how to apply the theories to get the result. Then the result will be presented in Chapter 4. Then the conclusion will be discussed in Chapter 5.

CHAPTER 2

THEOREM

Aerodynamic airfoil shape design can basically categorize into two groups which are shape optimization and inverse shape design. The shape optimization method retrieves the best properties of the designed airfoil experimentally; consequently, it is very time consuming and expensive to operate the wind tunnel repeatedly. On the contrary, the inverse shape design does not require any experiment. Instead, a computer program will be used in place of the experiment which benefits by reducing time and cost. An aerodynamic airfoil shape will be iteratively obtained when the desired pressure distribution is given. In this paper, inverse airfoil design method will be used.

There are two main categories of inverse airfoil design which are methods with coupled analysis and shape modification, and methods with uncoupled analysis and shape modification. The coupled analysis methods are more complicated in writing a program than the uncoupled analysis. Some examples of this type are indirect transpiration technique, stream-function-as-coordinate approaches, characteristic boundary condition approaches, and adjoint operator/control theory approaches. On the other hand, the uncoupled analysis methods are simpler than the coupled method and easier in writing a program. Some examples of this type are elastic membrane techniques (Modified Garabedian-McFadden Method or MGM Method), and the DISC technique [2]. In this paper, the MGM method will be used.

2.1 Modified Garabedian-McFadden (MGM) Method

Modified Garabedian-McFadden Method is the elastic membrane technique which is based on the linearized small disturbance potential flow solution of the 2D wavy-wall problem on the basis of the difference between an actual pressure coefficient distribution (C_p^{actual}) and a target pressure coefficient distribution (C_P^{target}) [4,10,11,14,16]. The equation can be expressed as

$$\beta_0 \Delta y + \beta_1 \frac{d\Delta y}{dx} + \beta_2 \frac{d^2 \Delta y}{dx^2} = C_P^{target} - C_P^{actual} \qquad (2.1)$$

where

- x is the coordinate which runs along the airfoil chord
- Δy represents the airfoil contour changes in Cartesian coordinate
- β_0, β_1, and β_2 are the coefficients which are arbitrary constants. They are chosen to get stable cycles and to accelerate convergence [4]

2.1.1. Central Difference Discretization

Now, the derivative terms in equation (1) need to be discretized in order to solve it numerically. Taylor expansions are the most standard way to obtain the substitution of derivative terms with finite difference. For the one dimensional case, the expansion of function u(x) about point (i) can be expressed as

$$u_{i+1} = u_i + \left(\frac{\partial u}{\partial x}\right)_i \Delta x + \left(\frac{\partial^2 u}{\partial x^2}\right)_i \frac{(\Delta x)^2}{2} + \left(\frac{\partial^3 u}{\partial x^3}\right)_i \frac{(\Delta x)^3}{6} + \cdots \qquad (2.2)$$

and

$$u_{i-1} = u_i - \left(\frac{\partial u}{\partial x}\right)_i \Delta x + \left(\frac{\partial^2 u}{\partial x^2}\right)_i \frac{(\Delta x)^2}{2} - \left(\frac{\partial^3 u}{\partial x^3}\right)_i \frac{(\Delta x)^3}{6} + \cdots \qquad (2.3)$$

A first-order derivative finite-difference form of second-order accuracy can be obtained by subtracting equation (3) from equation (2); then, we get

$$u_{i+1} - u_{i-1} = 2\left(\frac{\partial u}{\partial x}\right)_i \Delta x + 2\left(\frac{\partial^3 u}{\partial x^3}\right)_i \frac{(\Delta x)^3}{6} + \cdots \qquad (2.4)$$

If only the first three terms of equation (2) and (3) are considered and leave others as truncation error (O$(\Delta x)^2$), we can rearrange equation (4) as

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1}-u_{i-1}}{2\Delta x} + O(\Delta x)^2 \qquad (2.5)$$

Similarly, we can get a second-order derivative finite-difference form of second-order accuracy by adding equation (2) and equation (3); then, we get

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_{i+1}-2u_i+u_{i-1}}{(\Delta x)^2} + O(\Delta x)^2 \qquad (2.6)$$

By considering function $\Delta y(x)$ in the MGM equation (equation (1)) as function u(x) in equation (5) and (6) then substituting eqn. (5) and eqn. (6) into the MGM equation, we get

$$\beta_0 \Delta y_i + \beta_1\left[\frac{\Delta y_{i+1}-\Delta y_{i-1}}{2\Delta x}\right] + \beta_2\left[\frac{\Delta y_{i+1}-2\Delta y_i+\Delta y_{i-1}}{(\Delta x)^2}\right] = C_P^{target} - C_P^{actual} \qquad (2.7)$$

Rearranging equation (7), we obtain

$$\left[\frac{\beta_2}{(\Delta x)^2} - \frac{\beta_1}{2\Delta x}\right]\Delta y_{i-1} + \left[\beta_0 - \frac{2\beta_2}{(\Delta x)^2}\right]\Delta y_i + \left[\frac{\beta_1}{2\Delta x} + \frac{\beta_2}{(\Delta x)^2}\right]\Delta y_{i+1} + O(\Delta x)^2 = C_P^{target} - C_P^{actual} \qquad (2.8)$$

Note that by using equation (8), Δx has to be constant. Therefore, it is inconvenient to use equation (8) when we get the non-uniform coordinates of the initial airfoil. The other method of discretization is presented next in order to avoid the mentioned difficulty.

2.1.2. Finite Difference for Non-Uniform Grid

Considering figure 2.1 , Taylor's series can be used to expand the solution (Δy) about a=x_i from point "i" to point "i-1" and "i+1" as

$$\Delta y_{i+1} = \Delta y_i + \frac{d\Delta y}{dx}\frac{(x_{i+1}-x_i)}{1!} + \frac{d^2\Delta y}{dx^2}\frac{(x_{i+1}-x_i)^2}{2!} + \frac{d^3\Delta y}{dx^3}\frac{(x_{i+1}-x_i)^3}{3!} + O(x^3) \qquad (2.9.1)$$

$$\Delta y_{i-1} = \Delta y_i + \frac{d\Delta y}{dx}\frac{(x_{i-1}-x_i)}{1!} + \frac{d^2\Delta y}{dx^2}\frac{(x_{i-1}-x_i)^2}{2!} + \frac{d^3\Delta y}{dx^3}\frac{(x_{i-1}-x_i)^3}{3!} + O(x^3) \qquad (2.10.1)$$

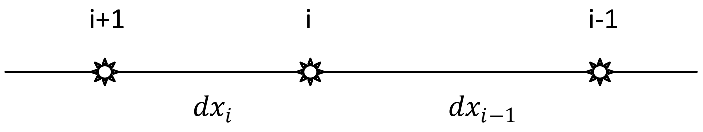


Figure 2.1 Non-Uniform 1D grids where $dx_i = x_{i+1} - x_i$ and $dx_{i-1} = x_i - x_{i-1}$

Rewriting equation (2.9) and (2.10), we obtain

$$\Delta y_{i+1} = \Delta y_i + \frac{d\Delta y}{dx}(dx_i) + \frac{d^2\Delta y}{dx^2}\frac{(dx_i)^2}{2} + \frac{d^3\Delta y}{dx^3}\frac{(dx_i)^3}{6} + O(x^3) \qquad (2.9.2)$$

$$\Delta y_{i-1} = \Delta y_i - \frac{d\Delta y}{dx}(dx_{i-1}) + \frac{d^2\Delta y}{dx^2}\frac{(dx_{i-1})^2}{2} - \frac{d^3\Delta y}{dx^3}\frac{(dx_{i-1})^3}{6} + O(x^3) \qquad (2.10.2)$$

Multiplying equation (2.9.2) by γ_1 and equation (2.10.2) by γ_2 where γ_1 and γ_2 can be any number, we get

$$\gamma_1\Delta y_{i+1} = \gamma_1\Delta y_i + \gamma_1\frac{d\Delta y}{dx}(dx_i) + \gamma_1\frac{d^2\Delta y}{dx^2}\frac{(dx_i)^2}{2} + \gamma_1\frac{d^3\Delta y}{dx^3}\frac{(dx_i)^3}{6} + O(x^3) \qquad (2.9.3)$$

$$\gamma_2\Delta y_{i-1} = \gamma_2\Delta y_i - \gamma_2\frac{d\Delta y}{dx}(dx_{i-1}) + \gamma_2\frac{d^2\Delta y}{dx^2}\frac{(dx_{i-1})^2}{2} - \gamma_2\frac{d^3\Delta y}{dx^3}\frac{(dx_{i-1})^3}{6} + O(x^3) \qquad (2.10.3)$$

Subtracting equation (2.10.3) from equation (2.9.3), we have

$$\begin{aligned}\gamma_1\Delta y_{i+1} - \gamma_2\Delta y_{i-1} = (\gamma_1 - \gamma_2)\Delta y_i + (\gamma_1 dx_i + \gamma_2 dx_{i-1})\frac{d\Delta y}{dx} + \\ \left(\frac{\gamma_1(dx_i)^2 - \gamma_2(dx_{i-1})^2}{2}\right)\frac{d^2\Delta y}{dx^2} + \\ \left(\frac{\gamma_1(dx_i)^3 + \gamma_2(dx_{i-1})^3}{6}\right)\frac{d^3\Delta y}{dx^3} + O(x^3)\end{aligned} \qquad (2.11)$$

Adding equation (2.9.3) and (2.10.3), we get

$$\gamma_1\Delta y_{i+1} + \gamma_2\Delta y_{i-1} = (\gamma_1+\gamma_2)\Delta y_i + (\gamma_1 dx_i - \gamma_2 dx_{i-1})\frac{d\Delta y}{dx} + \left(\frac{\gamma_1(dx_i)^2+\gamma_2(dx_{i-1})^2}{2}\right)\frac{d^2\Delta y}{dx^2} + \left(\frac{\gamma_1(dx_i)^3-\gamma_2(dx_{i-1})^3}{6}\right)\frac{d^3\Delta y}{dx^3} + O(x^3) \quad (2.12)$$

By choosing γ_1=$(dx_{i-1})^2$ and γ_2=$(dx_i)^2$, then neglecting the third order derivative in equation (2.11), we get

$$(dx_{i-1})^2\Delta y_{i+1} - (dx_i)^2\Delta y_{i-1} = ((dx_{i-1})^2-(dx_i)^2)\Delta y_i + ((dx_{i-1})^2 dx_i + (dx_i)^2 dx_{i-1})\frac{d\Delta y}{dx} + \left(\frac{(dx_{i-1})^2(dx_i)^2-(dx_i)^2(dx_{i-1})^2}{2}\right)\frac{d^2\Delta y}{dx^2} + O(x^2) \quad (2.11.1)$$

Therefore, the second order derivative term is canceled out then we obtain

$$\frac{d\Delta y}{dx} = \frac{(-dx_i^2)\Delta y_{i-1} + (dx_i^2 - dx_{i-1}^2)\Delta y_i + (dx_{i-1}^2)\Delta y_{i+1}}{S_1} + O(x^2) \quad (2.13)$$
$$\text{where } S_1 = dx_{i-1}^2 dx_i + dx_i^2 dx_{i-1}$$

By choosing γ_1=dx_{i-1} and γ_2=dx_i, then neglecting the third order derivative in equation (2.12), we get

$$dx_{i-1}\Delta y_{i+1} + dx_i\Delta y_{i-1} = (dx_{i-1}+dx_i)\Delta y_i + (dx_{i-1}dx_i - dx_i dx_{i-1})\frac{d\Delta y}{dx} + \left(\frac{dx_{i-1}(dx_i)^2 + dx_i(dx_{i-1})^2}{2}\right)\frac{d^2\Delta y}{dx^2} + O(x^2) \quad (2.12.1)$$

Therefore, the first derivative term is canceled out then we obtain

$$\frac{d^2\Delta y}{dx^2} = \frac{(dx_i)\Delta y_{i-1} - (dx_{i-1} + dx_i)\Delta y_i + (dx_{i-1})\Delta y_{i+1}}{S_2} + O(x^2)$$
$$\text{where} \quad S_2 = \frac{dx_{i-1}^2 dx_i + dx_i^2 dx_{i-1}}{2} = \frac{S_1}{2} \tag{2.14}$$

Substituting equation (2.13) and (2.14) into equation (2.1), we obtain

$$\beta_0(\Delta y_i) + \beta_1\left(\frac{(-dx_i^2)\Delta y_{i-1} + (dx_i^2 - dx_{i-1}^2)\Delta y_i + (dx_{i-1}^2)\Delta y_{i+1}}{S_1}\right) + \beta_2\left(\frac{(dx_i)\Delta y_{i-1} - (dx_{i-1} + dx_i)\Delta y_i + (dx_{i-1})\Delta y_{i+1}}{S_2}\right) + O(x^2) = C_P^{target} - C_P^{actual} \tag{2.15}$$

Rearranging equation (2.15), we get

$$A_i\Delta y_{i-1} + B_i\Delta y_i + C_i\Delta y_{i+1} = C_P^{target} - C_P^{actual} \tag{2.16}$$

where,

$$A_i = -\beta_1\frac{dx_i^2}{S_1} + \beta_2\frac{dx_i}{S_2}$$

$$B_i = \beta_0 - \beta_1\frac{dx_{i-1}^2 - dx_i^2}{S_1} - \beta_2\frac{dx_{i-1} + dx_i}{S_2}$$

$$C_i = \beta_1\frac{dx_{i-1}^2}{S_1} + \beta_2\frac{dx_{i-1}}{S_2}$$

when

$S_1 = dx_{i-1}^2 dx_i + dx_i^2 dx_{i-1}$ and $S_2 = \frac{dx_{i-1}^2 dx_i + dx_i^2 dx_{i-1}}{2} = \frac{S_1}{2}$

By using the discretization for a non-uniform grid, Δx does not need to be constant. Therefore, it can be applied to a more sophisticated mesh than using regular central different discretization, which can only be applied to the uniform grid.

2.2 Tri-Diagonal Matrix Algorithms

After applying one of the discretization methods, it usually comes out with a system of linear equations

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

where $a_1 = 0$ and $c_n = 0$ which can be rewritten in a metric form as

$$\begin{bmatrix} b_1 & c_1 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & & \vdots \\ 0 & a_3 & b_3 & \ddots & 0 \\ \vdots & & \ddots & \ddots & c_{n-1} \\ 0 & \dots & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}$$

Modifying the coefficients b_i and c_i in order to eliminate the coefficients a_i, we get

$$c_i' = \begin{cases} \dfrac{c_1}{b_1} & ; i = 1 \\ \dfrac{c_i}{b_i - c_{i-1}' a_i} & ; i = 2, 3, \dots, n-1 \end{cases}$$

and

$$d_i' = \begin{cases} \dfrac{d_1}{b_1} & ; i = 1 \\ \dfrac{d_i - d_{i-1}' a_i}{b_i - c_{i-1}' a_i} & ; i = 2, 3, \dots, n-1 \end{cases}$$

Therefore, we get

$$\begin{bmatrix} b_1 & c_1' & & & 0 \\ & b_2 & c_2' & & \\ & & b_3 & \ddots & \\ & & & \ddots & c_{n-1}' \\ 0 & & & & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1' \\ d_2' \\ d_3' \\ \vdots \\ d_n' \end{bmatrix}$$

By doing backward sweep, we get

$$x_n = d_n'$$

$$x_i = d_i' - c_i' x_{i+1} \quad ; i = n-1, n-2, \ldots, 1.$$

2.3 NACA Four-Digit Series Airfoils

The National Advisory Committee for Aeronautics (NACA) developed the NACA airfoil series. There are many types of airfoils which are used with different methods to generate airfoil coordinates. For example, the 4-digit airfoil and 5-digit airfoil are generated based on analytical equations where one is for expressing the camber of the mean-line of the airfoil section and the other one is for describing the section's thickness distribution along the chord of the airfoil. For newer airfoil series like the 6-digit airfoil, more sophisticated shapes are introduced which require theoretical methods rather than geometrical methods. For convenience, the NACA 4-digit series will be used in this thesis.

The NACA Four-digit Series has 4 numbers in its name. The first digit indicates the maximum camber (m) in percentage of the chord, the second digit designates the position of the maximum camber (p) in tenths of chord, and the other two digits specify the maximum thickness (t) of the airfoil in percentage of chord. For illustration, NACA 4315 airfoil is a 4% camber airfoil which the maximum camber is situated 30% back from the leading edge with a maximum thickness of 15%. The coordinate of the 4-digit airfoils are generated from the following procedure.

First, the values of x from 0 to the maximum chord c are chosen; the number of x depends on how smooth of the airfoil one requires. Second, substituting the values of m and p into the following equations at every x coordinates; the mean camber line coordinates are presented

$$y_c = \frac{m}{p^2}(2px - x^2) \qquad \text{from } x = 0 \ \ to \ \ x = p$$

$$y_c = \frac{m}{(1-p)^2}((1-2p) + 2px - x^2) \qquad \text{from } x = p \ \ to \ \ x = c$$

Third, substituting the maximum thickness (t) into the following equation, the thickness distribution is obtained

$$y_t = \frac{t}{0.2}(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4)$$

Last, the final airfoil coordinates are calculated from

$$X_U = x - y_t sin\theta$$

$$Y_U = y_c + y_t cos\theta$$

$$X_L = x + y_t sin\theta$$

$$Y_L = y_c - y_t cos\theta$$

where (X_U, Y_U) are the coordinates of the upper airfoil

(X_L, Y_L) are the coordinates of the lower airfoil

$\theta = arctan\left(\frac{dy_c}{dx}\right)$

2.4 Flow Analysis

2.4.1. Mesh Generating

The moving grid program developed by Dennis [15], called "mvg", is the program for generating 2-D meshes both structured and unstructured mesh which is based on the grid deformation technique. The grid deformation technique bases on the spring analogy which treats mesh as linear springs as shown in Figure 2.2.

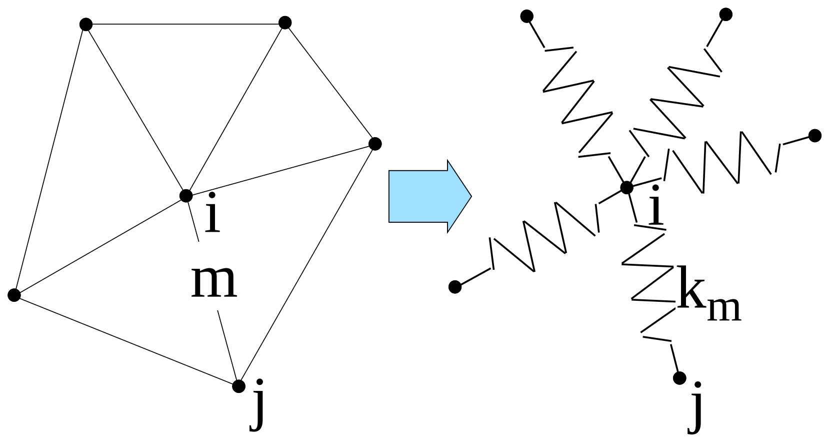


Figure 2.2 Mesh when treated as linear springs

New mesh is obtained when static equilibrium is achieved by minimizing the energy in spring system. The program is effective when using with small deformations of boundary.

The program required a base mesh and a modified boundary where the base mesh should be in qgrid, fgrid, or ugrid (2D) format and the airfoil boundary maker should be 1. In order to use the "mvg" program in LINUX, simply type "./mvg airfoil.dat inputgrid outputgrid". The airfoil.dat file is the modified boundary, the base mesh is inputgrid, and the outputgrid is the output file which can be in fgrid, tec (Tecplot), or neu (Gambit neutral file) format. The program is very fast in generating grids of the desired airfoil because only airfoil coordinates are needed as the modified boundary, then the mesh is generated. Compared to Gambit, we have to take so much more time to make all edges, then mesh, and specify the boundary conditions before we can export the mesh to FLUEN. However, the user must be aware of the number of points in the airfoil.dat file. It must have the same number of points as the airfoil in the base grid and the ordering (clockwise, counterclockwise, starting point, etc.) of the points should be the same as well.

2.4.2. Flow Solver (FLUENT)

FLUENT is software for simulating fluid flows in complicated geometries [20]. A mesh of any geometry needs to be generated outside of FLUENT then import to the solver. In order to properly solve a problem using FLUENT, there are three important points to be considered; definition of the modeling goals, choice of the computational model, choice of physical models, and determination of the solution procedure. For the definition of the modeling goals, proper specific results and degree of accuracy are considered. For the choice of the computational model, computational domain, and boundary conditions should be set properly. For the choice of physical models, user should know the fundamental physical behavior of flows. For example, is the flow inviscid, laminar, or turbulent?, is the flow unsteady or steady?, is heat transfer important?, or will you treat the flow as incompressible or compressible? For the determination of the solution procedure, the formulation and solution parameters should be set in order to be suitable to a problem so convergence will be obtained or accelerated.

To solve a problem using FLUENT, there are fourteen steps to follow. First of all, create the model geometry and mesh by using mesh generating software or program outside FLUENT, then start the appropriate solver for 2D or 3D modeling, then import the mesh, then check the mesh, then select the solver formulation. After that, choose the basic equations to be solved such as laminar or turbulent, then specify material properties, then specify the boundary conditions. Next, adjust the solution control parameters, then initialize the flow field, then calculate the solution, then examine the results, then save the results, and then, if necessary, refine the grid or revise the numerical or physical model.

In FLUENT there are three solver formulations to use; segregated, coupled implicit, and coupled explicit. All three solver formulations give accurate solutions for a wide range of flows. However, in some case, one formulation may be better than others when consider the rate of convergence. The segregated solver sequentially solves the continuity, momentum, and energy equations. On the other hand, the coupled solver will solve the continuity, momentum, and energy equations equations simultaneously. The segregated solver is suitable for solving low-speed incompressible flow and the coupled implicit solver is appropriate when solving high-speed compressible flow. When coupled implicit is desirable but memory of a machine is not enough, the coupled explicit solver may be used instead.

CHAPTER 3

METHODOLOGY

According to the MGM equation,

$$\beta_0 \Delta y + \beta_1 \frac{d\Delta y}{dx} + \beta_2 \frac{d^2 \Delta y}{dx^2} = C_P^{target} - C_P^{actual}$$

one can see that the pressure coefficient is the input to the equation in order to get Δy, which changes the y-coordinates of the initial airfoil. There are two C_P distributions to input into the equation. First, is the target C_P distribution which is the C_P distribution that we want for the desired airfoil. The target C_P distribution is set to be constant all over the process. Second, is the actual C_P distributions that will change every iteration. One can choose the C_P distributions of any airfoil at the first iteration and the MGM will yield Δy at each x-coordinate; then, the new airfoil coordinates are obtained. We will use GAMBIT to generate the mesh of the new airfoil, then import the mesh to FLUENT to analyze the flow. Finally, the actual C_P distribution is sent out from FLUENT and it serves as the new input of MGM equation for the next iteration. Once the actual C_P distribution approaches the target C_P distribution, the C_P distributions of the target airfoil is what we want. Therefore, the airfoil coordinates of the last iteration is the target airfoil. For illustration, the process is shown in Figure (3.1).

3.1 Initial and Target Airfoil data collecting

3.1.1. Airfoil Coordinate Generating

In order to validate the programming of MGM equation, the target C_P distribution is chosen from the known coordinate airfoil. In this thesis, the NACA 2315 airfoil is used. For the initial airfoil, we can arbitrarily choose one; however, symmetrical airfoils are recommended in order to prevent the divergence in some cases. Therefore, the symmetrical airfoil NACA0011 is selected to be the initial airfoil. From section 2.3, we can write a program in MATLAB to generate the desired 4-digit airfoil coordinates, with the input being the 4 numbers from the name of the airfoils. The output has 3 columns in [X Y Z] form. Copy the output from MATLAB, then save it as a text data file. The program is presented in Appendix A. Once we get the coordinates of NACA0011 and NACA2315, GAMBIT is used to generate the mesh grid of both airfoils.

3.1.2. Mesh Generating

Now, we have airfoil coordinates of NACA0011 and NACA2315 in a ".txt" file. The text file are imported to GAMBIT as vertex data by clicking "File", then choose "import", then "Vertex Data", then browse the text file you already have. The vertex data of the airfoil will be presented in the screen of GAMBIT with some additional needed vertexes for creating the outer boundary, which produced from the NACA 4-digit generator program shown in Appendix A. Figure 3.1 shows the vertexes data in GAMBIT. Now the needed vertexes are imported to GAMBIT, then edges will be created. For the airfoil vertices, the surfaces will be created base on an interpolation method. First, in operating the process is to choose the Geometry command button, then choose the edge command button, and then choose the create edge from vertices. Create upper surface and lower surface separately then the airfoil surfaces are created as shown in figure 3.3.

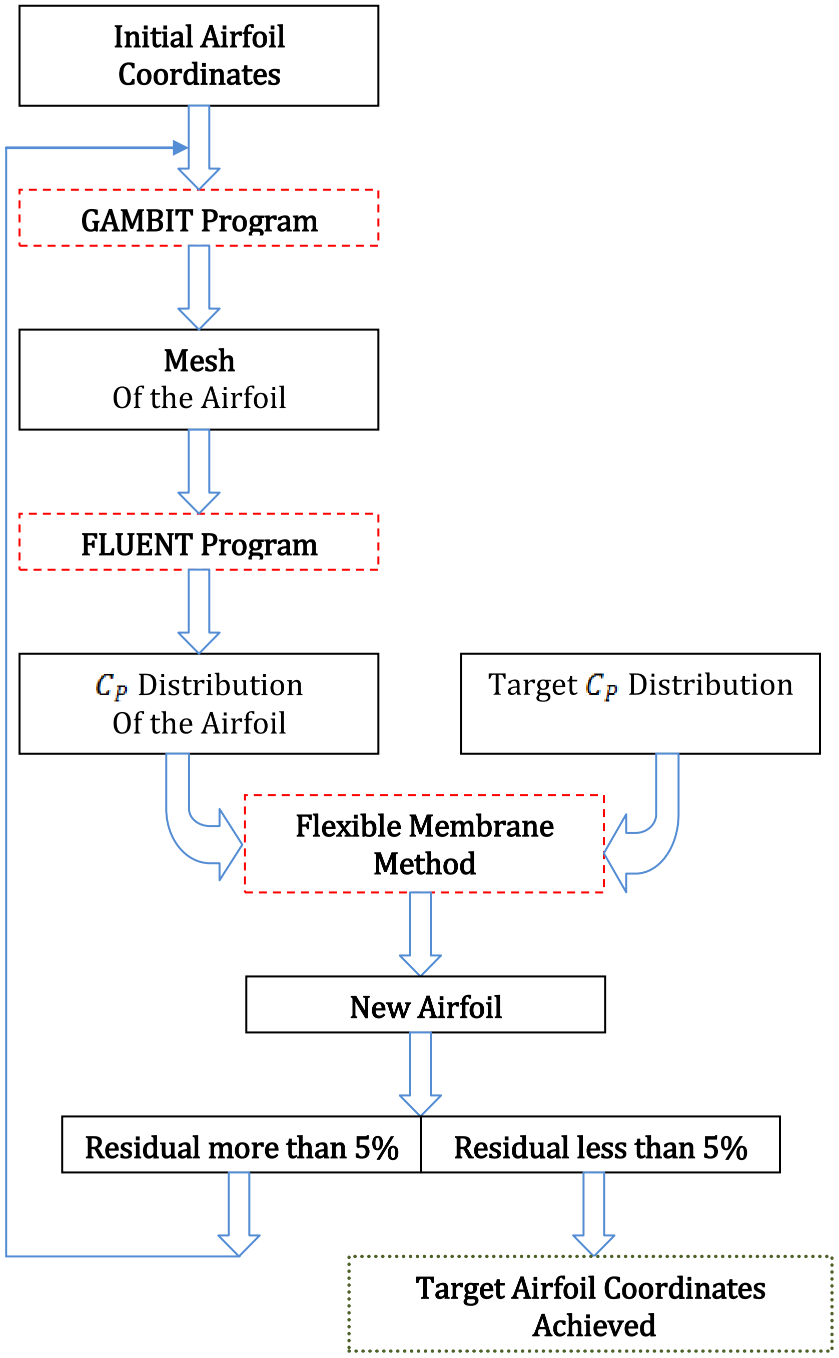


Figure 3.1 Inverse Airfoil Design Methodology

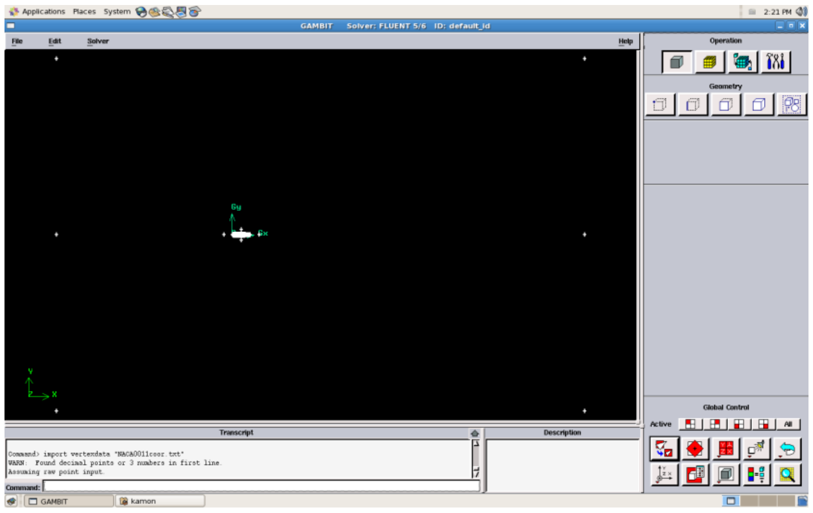

```
Command> import vertexdata "NACA0011coor.txt"
WARN:  Found decimal points or 3 numbers in first line.
Assuming raw point input.
```

Figure 3.2 Vertex data after import into GAMBIT

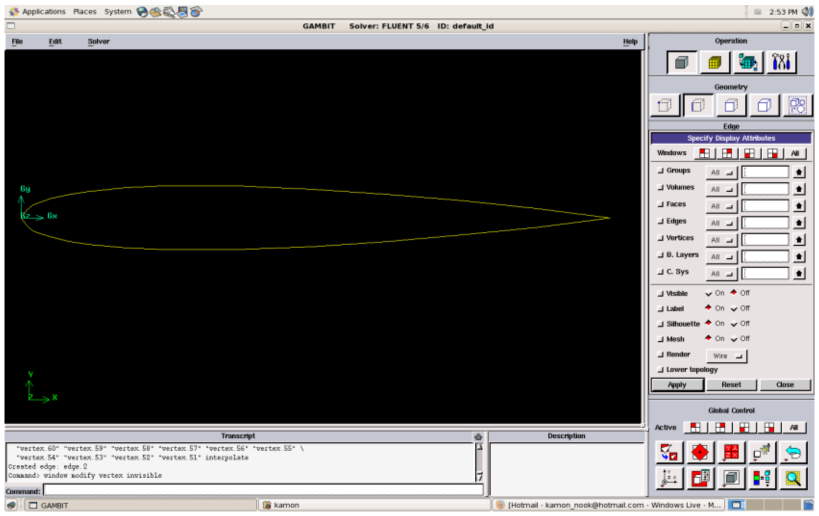
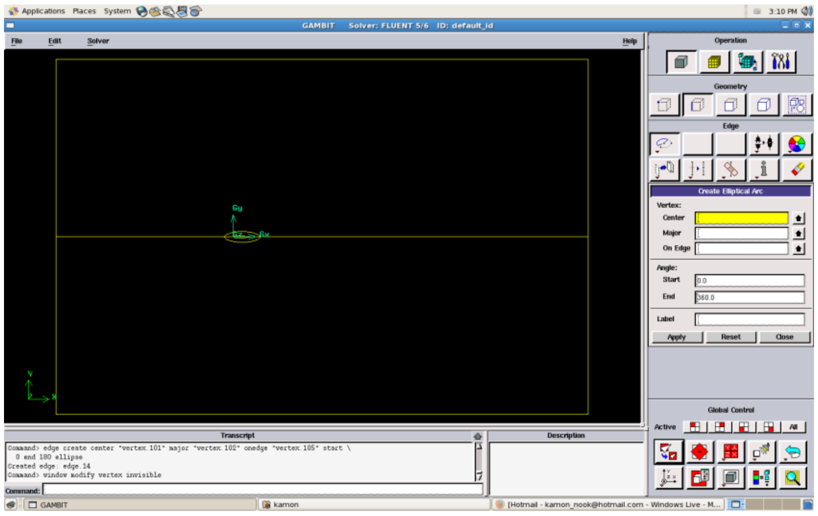

```
  "vertex.60" "vertex.59" "vertex.58" "vertex.57" "vertex.56" "vertex.55" \
  "vertex.54" "vertex.53" "vertex.52" "vertex.51" interpolate
Created edge: edge.2
Command> window modify vertex invisible
```

Figure 3.3 NACA0011 Airfoil after edges creating

Use four vertices around airfoil and one vertex inside the airfoil to create elliptic boundary around the airfoil and then use six other vertices to create the outer boundary. Then create straight lines to half cut the domain into two sections, upper and lower one, which is easier to mesh grids in the entire domain. The result is shown in figure 3.4. Note that the six outer vertices have to be created far enough from the airfoil in order to get the correct solution from

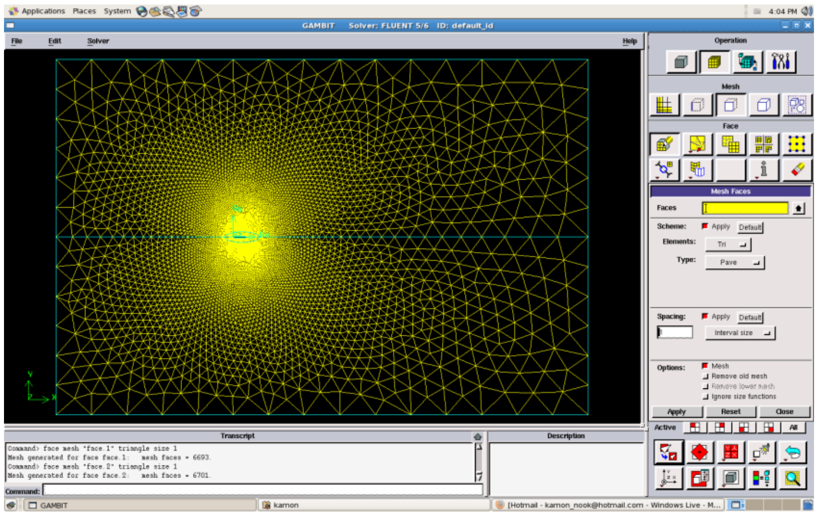
FLUENT. According to this thesis, only subsonic and transonic will be considered. Therefore, it is alright to set the outer domain in front of the airfoil ten times of airfoil's chord and also for the upper and lower parts. However, behind the airfoil, turbulence usually occurs; therefore, the outer domain behind the airfoil should be further. In this case, twenty times the chord length is used.



```
Command> edge create center "vertex.101" major "vertex.102" onedge "vertex.105" start \
  0 end 180 ellipse
Created edge: edge.14
Command> window modify vertex invisible
```

Figure 3.4 Edges of entire domain

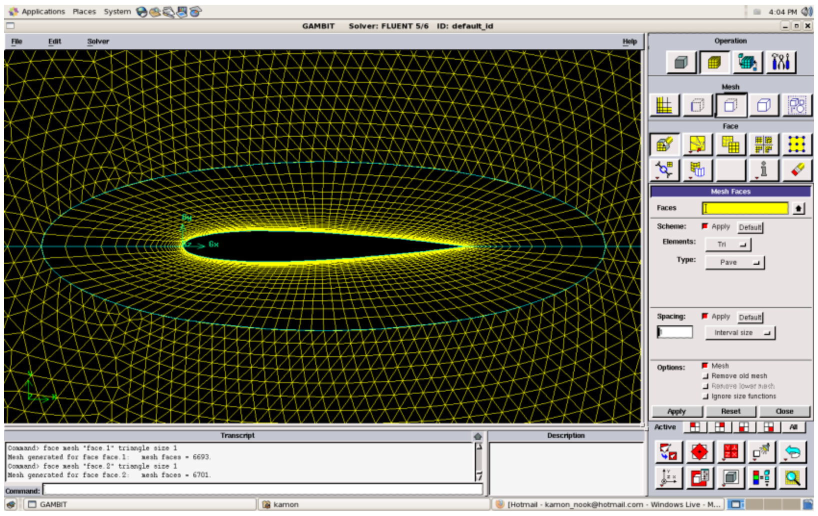
Next, all edges are meshed in such a way that small grids will be created in the area near the airfoil, and bigger grids will be created for the region far away from the airfoil. Then create four faces, which are the upper and lower face between airfoil surface and elliptic edges, the upper and lower face of outer edges and elliptic edges. Then link the two horizontal outer edges to be periodic. After that, mesh two faces which are close to the airfoil as quadrilateral structured grids and the other outer faces as triangular unstructured grids. The result is presented in figure 3.5 and figure 3.6. Now, all edges need to be specified for boundary conditions. Velocity inlet is set for the outer domain edges, in front of the airfoil. Periodic is set for the outer domain edges, upper and lower of the airfoil. Pressure outlet is set for the outer domain edges behind the airfoil. Wall is set for the airfoil surfaces. The others are set to be Interior. Then the mesh is exported for FLUENT.



Elements: Tri
Type: Pave
Spacing: Interval size

```
Command> face mesh "face.1" triangle size 1
Mesh generated for face face.1:   mesh faces = 6693.
Command> face mesh "face.2" triangle size 1
Mesh generated for face face.2:   mesh faces = 6701.
```

Figure 3.5 Mesh for NACA0011 of entire domain



Elements: Tri
Type: Pave
Spacing: Interval size

```
Command> face mesh "face.1" triangle size 1
Mesh generated for face face.1:   mesh faces = 6693.
Command> face mesh "face.2" triangle size 1
Mesh generated for face face.2:   mesh faces = 6701.
```

Figure 3.6 Mesh for NACA0011 close to the airfoil surface

3.1.3 Flow Solver

After a mesh of an airfoil is created, now FLUENT is the software for analyzing the flow over the airfoil. All information of the flow, such as Pressure, Velocity, Density, etc., will be stored at each node which is meshed from GAMBIT. In this thesis, the only information needed is Pressure Coefficient (Cp) Distribution along the airfoil surfaces. In order to get the Cp Distribution, start from “read case”, then open the “.msh” file which is exported from GAMBIT.

Then check the grid and if there is no error found, it can be analyzed. Then define model to be inviscid or laminar as desired. After that the boundary conditions which are previously defined from GAMBIT are rechecked again. Actually, there is nothing to do with the boundary conditions except the velocity inlet boundary, which has to be set the magnitude of the inlet air velocity and direction. The referent value of velocity is needed to be changed to the same magnitude as it is set in the boundary condition in order to get the real value of any output parameters. After that, a solution is needed to be initialized. Normally, the X-velocity is initialized to be the same amount as the velocity inlet. Then we can choose what method would we like FLUENT to use for solving the solution. Generally, the SIMPLE method is used for Pressure-Velocity Coupling. For Discretization, the standard method is used for Pressure and First Order Upwind is used for Momentum. The under-Relaxation Factors may be reduced in order to get the faster conversion. However, if the solution still seems to converge slowly, changing the method may be a good way to do it. Now the solution is ready to be solved, the only thing to do is monitor it. The first one is Residual Monitors where the absolute criteria should set to 1e-06 to ensure that the solution does converge. However, it is difficult that the continuity residual is going to be less than 1e-06 because the real flow might not flow smoothly. Vertex or turbulent may happen behind the airfoil which make the solution not unique and it causes the continuity residual stay higher than the criteria. Therefore, Force Monitors may be needed to monitor the convergence of the solution. In this thesis, the Lift Coefficient is used because by monitoring CL of the airfoil, CP at every node are all considered. Therefore, when CL converges, it means that the solution is obtained. For illustration, the Residual and Lift Coefficient Monitors of NACA0011 are presented on figure 3.7 and figure 3.8.

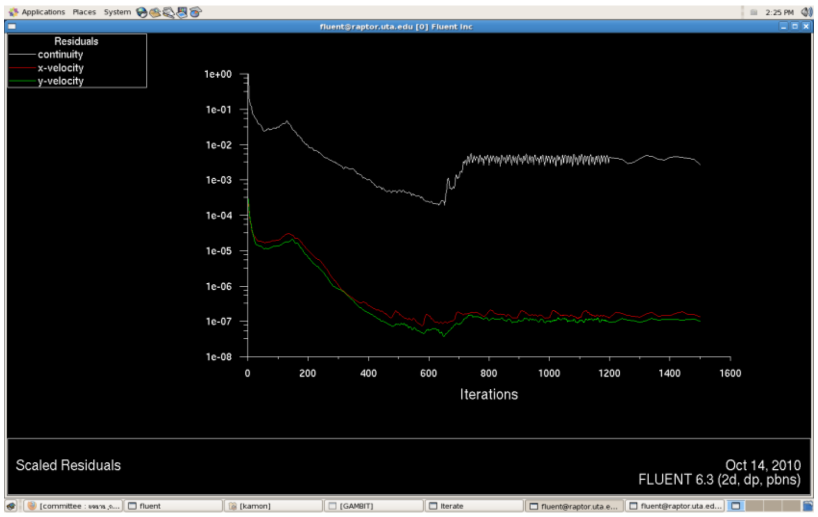


Figure 3.7 Residual from solving the flow of NACA0011

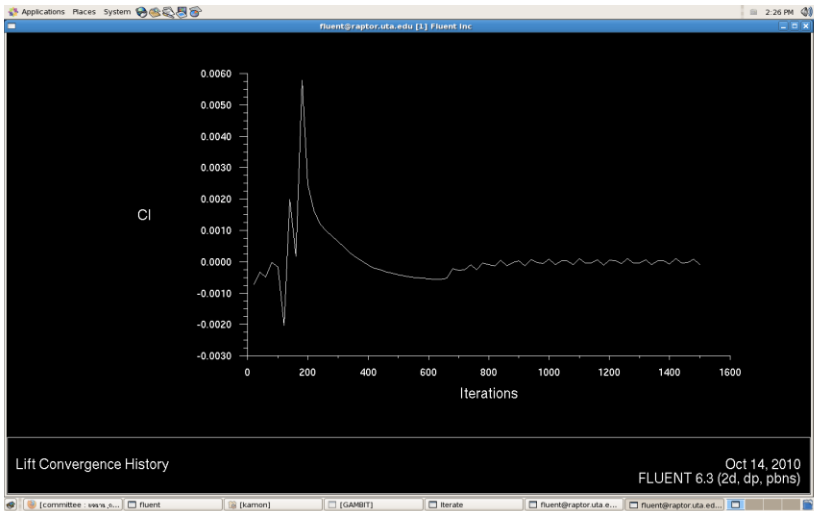


Figure 3.8 CL monitor of NACA0011

<u>3.2 Flexible Membrane Method</u>

From the MGM equation,

$$\beta_0 \Delta y + \beta_1 \frac{d\Delta y}{dx} + \beta_2 \frac{d^2 \Delta y}{dx^2} = C_P^{target} - C_P^{actual}$$

using the central difference method to discretize the MGM equation, we got equation (2.8)

$$\left[\frac{\beta_2}{(\Delta x)^2}-\frac{\beta_1}{2\Delta x}\right]\Delta y_{i-1}+\left[\beta_0-\frac{2\beta_2}{(\Delta x)^2}\right]\Delta y_i+\left[\frac{\beta_1}{2\Delta x}+\frac{\beta_2}{(\Delta x)^2}\right]\Delta y_{i+1}+O(\Delta x)^2 = C_P^{target}-C_P^{actual} \qquad (2.8)$$

Equation (2.8) can be rewritten with neglecting the truncation error as

$$\left[\frac{\beta_2}{(\Delta x)^2}-\frac{\beta_1}{2\Delta x}\right]\Delta y_{i-1}+\left[\beta_0-\frac{2\beta_2}{(\Delta x)^2}\right]\Delta y_i+\left[\frac{\beta_1}{2\Delta x}+\frac{\beta_2}{(\Delta x)^2}\right]\Delta y_{i+1}=C_P^{target}-C_P^{actual} \qquad (3.1)$$

By using the central difference discretization, Δx has to be constant for all grids along the airfoil. However, the C_P distribution of NACA0011 and NACA2315 which were exported from FLUENT do not give the C_P values at the coordinates that we need. The C_P values are at the grid points which were automatically discreted from GAMBIT using interval count. Although it is meshed with constant distant between two grid points, the surfaces of the airfoil are not along the X-axis; therefore, the coordinates, when projected onto X-axis, do not give a constant values ofΔx. In order to use equation (3.1), all data has to be done curve fittings then get a new set of data which is at the coordinates giving constantΔx. In this thesis, 'polyfit' function in MATLAB is used to fit the C_P data then 'polyval' function is used to get a new set of data at the desired points. $\Delta x = 0.004854369$is used then the chord of the unit chord length airfoil is cut into 206 elements with constantΔx. After fitting the C_P data of NACA0011 and NACA2315, the C_P data is used to produce the system of equations. From equation (3.1), we can first arbitrarily chooseβ_0, β_1, and β_2 for the equation then after solving the system of equations, β_0, β_1, and β_2 can be adapted in order that the new C_P distribution (C_P^{actual}) result is obtained closest to the target C_P. Nevertheless, there is no theoretical method to analyze whether it is closest to the target C_P or not. The only way to do is approximately observing. After obtaining the suitable constantsβ_0, β_1, and β_2, those values will be used for other iteration steps. From equation (8.1), becauseΔx, β_0, β_1, and β_2 are constant, we can rewrite equation (3.1) as

$$A\Delta y_{i-1} + B\Delta y_i + C\Delta y_{i+1} = \left(C_P^{target} - C_P^{actual}\right)_i \qquad (3.2)$$

where $A = \left[\frac{\beta_2}{(\Delta x)^2} - \frac{\beta_1}{2\Delta x}\right]$, B= $\left[\beta_0 - \frac{2\beta_2}{(\Delta x)^2}\right]$, and C=$\left[\frac{\beta_1}{2\Delta x} + \frac{\beta_2}{(\Delta x)^2}\right]$ are constants

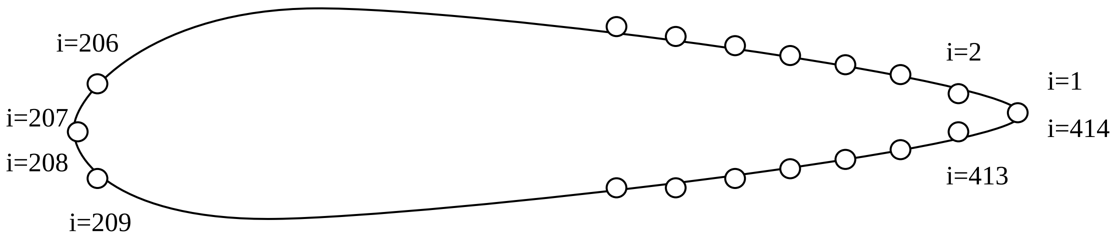


Figure 3.9 Grids along airfoil surfaces

To construct the system of equations, there are 3 boundary conditions to be considered:

1. No displacement at the trailing edge of the upper airfoil (Δy_1 = 0).
2. No displacement at the trailing edge of the lower airfoil (Δy_{414} = 0).
3. Δy at the leading edges of the upper airfoil and the lower airfoil (Δy_{207} = Δy_{208}) are equal.

The airfoil is divided into two parts, upper airfoil and lower airfoil. The grids of the upper airfoil are from i=1 to i=207, which are 206 elements. In the same manner, there are 50 elements as well for the lower airfoil starting from i=208 to i=414. The coordinate of each grid point is automatically produced by GAMBIT, so the coordinates change every time when the shape of airfoil changes. Getting C_P distribution of airfoils at desired coordinates is stated before. By starting from i=2 to i=101, the system of equations are presented below

$$A\Delta y_1 + B\Delta y_2 + C\Delta y_3 = \left(C_P^{target} - C_P^{actual}\right)_2$$

$$A\Delta y_2 + B\Delta y_3 + C\Delta y_4 = \left(C_P^{target} - C_P^{actual}\right)_3$$

so on and so forth until

$$A\Delta y_{411} + B\Delta y_{412} + C\Delta y_{413} = \left(C_P^{target} - C_P^{actual}\right)_{412}$$

$$A\Delta y_{412} + B\Delta y_{413} + C\Delta y_{414} = (C_P^{target} - C_P^{actual})_{413}$$

Applying the boundary conditions and rewriting the system of equations into matrix form, we obtain

$$[M]\{dY\} = \{dC_P\}$$

where

$$[M] = \left[\begin{array}{ccccccc} A & B & C & & & & 0 \\ & A & B & C & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & A & B & C & \\ 0 & & & & A & B & C \end{array}\right]_{412\times414}$$

$$\{dY\} = \left\{\begin{array}{c} \Delta y_1 \\ \Delta y_2 \\ \Delta y_3 \\ \vdots \\ \Delta y_{414} \end{array}\right\}_{414\times1}$$

and

$$\{dC_P\} = \left\{\begin{array}{c} (C_P^{target} - C_P^{actual})_2 \\ (C_P^{target} - C_P^{actual})_3 \\ (C_P^{target} - C_P^{actual})_4 \\ \vdots \\ (C_P^{target} - C_P^{actual})_{413} \end{array}\right\}_{412\times1}$$

Applying the boundary conditions 1 and 2, we get

$$\begin{bmatrix} A & B & C & & & & 0 \\ & A & B & C & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & A & B & C & \\ 0 & & & & A & B & C \end{bmatrix}_{412\times414} \begin{Bmatrix} 0 \\ \Delta y_2 \\ \vdots \\ \Delta y_{413} \\ 0 \end{Bmatrix}_{414\times1} = \begin{Bmatrix} (C_P^{target} - C_P^{actual})_2 \\ (C_P^{target} - C_P^{actual})_3 \\ (C_P^{target} - C_P^{actual})_4 \\ \vdots \\ (C_P^{target} - C_P^{actual})_{413} \end{Bmatrix}_{412\times1}$$

Then, matrix [M] will be reduced into two columns and matrices $\{dY\}$and$\{dC_P\}$ are also reduced to the first and the last row to make the programming easier. Therefore, we obtain

$$\begin{bmatrix} B & C & & & & 0 \\ A & B & C & & & \\ & \ddots & \ddots & \ddots & & \\ & & A & B & C & \\ & & & A & B & C \\ 0 & & & & A & B \end{bmatrix}_{412\times412} \begin{Bmatrix} \Delta y_2 \\ \Delta y_3 \\ \vdots \\ \Delta y_{412} \\ \Delta y_{413} \end{Bmatrix}_{412\times1} = \begin{Bmatrix} (C_P^{target} - C_P^{actual})_2 \\ (C_P^{target} - C_P^{actual})_3 \\ (C_P^{target} - C_P^{actual})_4 \\ \vdots \\ (C_P^{target} - C_P^{actual})_{413} \end{Bmatrix}_{414\times1}$$

By using the Thomas Algorithm to solve the system of equations, we first compute the new coefficient "C" in [M] in order to remove coefficients "A" then computes the values of$\{dC_P\}$. After that, the solution $\{dY\}$ is obtained by backward sweep the matrix as presented in section 2.2. The new airfoil Y-coordinates can then be obtained from

$$\{Y_{new}\} = \{Y_{initial}\} + \{dY\}$$

Finally, the coordinates of the new airfoil [X Y] are obtained and the first iteration finishes.

The new airfoil coordinates which have been obtained from the first iteration are going to be meshed by GAMBIT then analyzed by FLUENT again in order to get the $C_P$distribution. Then the $C_P$distribution of the new airfoil is used as initial information for the next iteration. The process is repeated until the solution converges; that is, when the error of the different between the target C_P distribution and new C_P distribution is less than 5 percent.

However, according to the results from FLUENT which will be provided in the next chapter, one can see that residuals are not less than the criteria. Although the number of iteration is enormous especially when the airfoil shapes are not symmetrical. Therefore, the process of making grids and setting parameters in FLUENT should be enhanced. Moreover, making grids in GAMBIT and setting parameters in FLUENT for every new airfoil is time consuming; therefore, using an adaptive grid program instead of GAMBIT is very useful. The improved methodology is presented in the next section.

3.3 Improved Methodology

3.3.1 Mesh Generating Improvement

In this section, the moving grid program is used instead of making grids of every new airfoil. However, in order to use the moving grid program, x-coordinates have to be fixed at the position of the x-coordinates of the base grid from the moving grid program. Therefore, coordinates of the initial airfoil, NACA0011, and target airfoil, NACA2315, have to be regenerated from the specific program called “naca” that will produce the compatible airfoil coordinates to the moving grid program called “mvg”. GAMBIT is not needed for the new methodology. The new methodology is shown in Figure 3.10. It is pretty much the same as the old one, just only moving grid program is used instead of GAMBIT and MGM equation is modified for a non-uniform grid.

In order to get the coordinates of an airfoil from “naca” program, type “./naca XXXX airfoil.dat” in the terminal window of LINUX operation which “naca” program is installed, where XXXX are the number of airfoil 4-digit and airfoil.dat is the name of the output file. It can be changed to any prefered name but has to be .dat file in order to be used in moving grid program. After obtaining the coordinates of an airfoil, the mesh is obtained by using “mvg” program by type “./mvg airfoil.dat inputgrid outputgrid”. In this thesis, inputgrid is already provided, named base.qgrid, and the outputgrid is in neutral file, file.neu, in order to be used in FLUENT. For more information about moving grid program, section 2.4.2 should be reviewed.

FLUENT is used for analyzing the flow in order to get pressure distribution on the airfoil surface. Journal file is written to save the procedure of setting all condition and parameters in FLUENT so that we can just only open this file to run for analyzing a new airfoil with the same conditions. In this thesis, inviscid subsonic, viscous subsonic, inviscid transonic and viscous transonic flows are considered in order to validate the inverse airfoil design program. In this section, viscous subsonic flow is considered to demonstrate the use of FLUENT. First, import the gambit mesh format (.neu file) which is from the output of “mvg” program. Check the grid and if there is no error, the mesh is good to use. In Define-Materials, the density is changed from constant to ideal-gas to make it more realistic. Other properties are reasonable so we do not need to change. In Define-Boundary Conditions, set airfoil boundary to be wall and the outer domain to be pressure far field then change Mach number to 0.2 since we are considering the flow in subsonic. In Define-Model-Energy, click the energy equation if it is not chosen yet. Then choose Laminar flow in Define-Model-Viscous for viscous flow analysis. Then initialize the solution with an x-velocity of 69.41754, as it is the velocity of Mach number 0.2 at standard condition. In Solve-Monitors-Residual, set all absolute Criteria to 10^{-6} in order to obtain a finer solution. Now the flow is ready to be solved. In Solve-Iterate, set the number of Iterations to 1500 as it is usually converged before so that the iteration will stop automatically when the errors go below the criteria. When the solution converges, export a data explorer file of pressure coefficient of the airfoil surface as the text file (.dx).The file is read by inverse airfoil design program written in MATLAB to calculate the new airfoil coordinates from the difference of pressure distribution. The analysis is demonstrated in the next section.

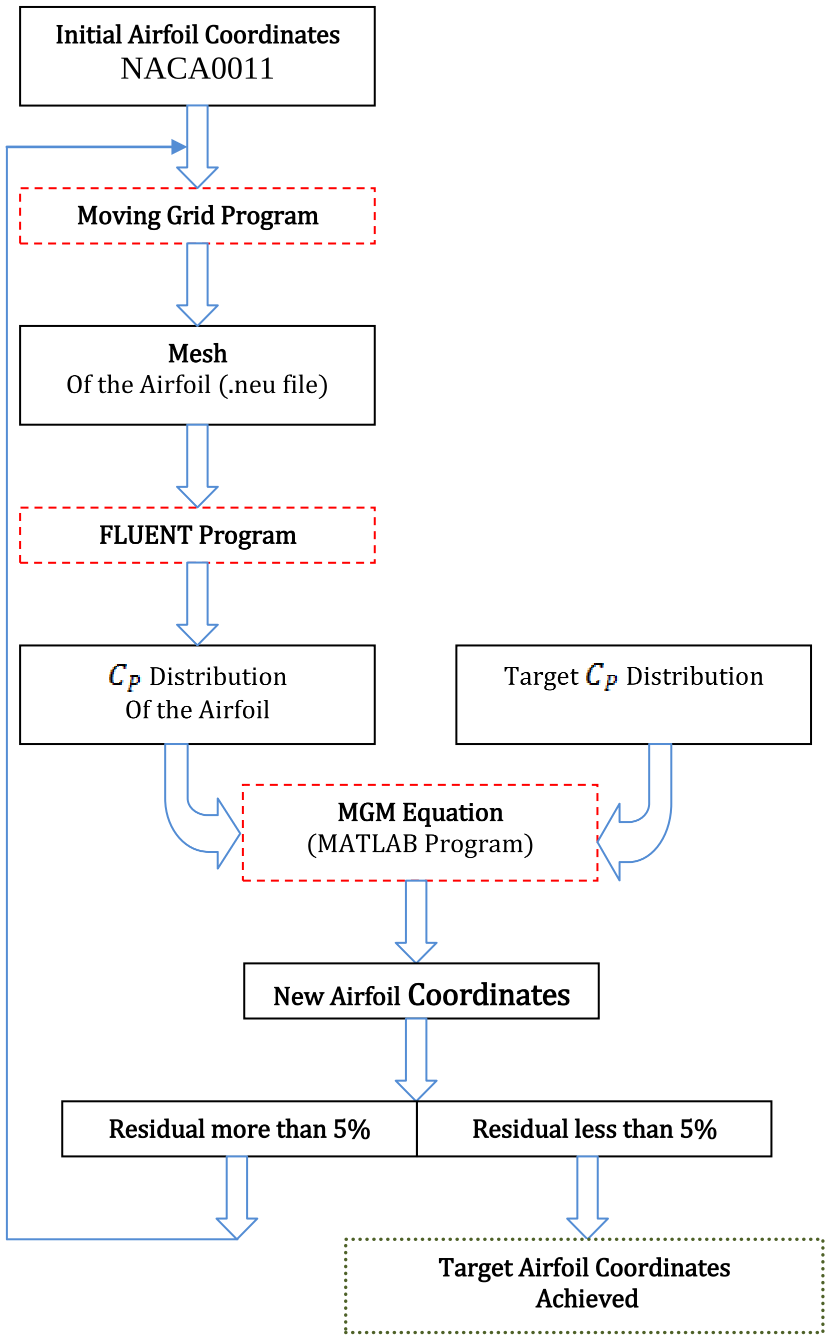


Figure 3.10 Improved Methodology; moving grid program is used instead of GAMBIT

3.3.2 Finite Difference Method for Non-Uniform Grid

Repeating equation (2.16) from section 2.1.2, we have

$$A_i \Delta y_{i-1} + B_i \Delta y_i + C_i \Delta y_{i+1} = C_P^{target} - C_P^{actual} \tag{2.16}$$

where,

$$A_i = -\beta_1 \frac{dx_i^2}{S_1} + \beta_2 \frac{dx_i}{S_2}$$

$$B_i = \beta_0 - \beta_1 \frac{dx_{i-1}^2 - dx_i^2}{S_1} - \beta_2 \frac{dx_{i-1} + dx_i}{S_2}$$

$$C_i = \beta_1 \frac{dx_{i-1}^2}{S_1} + \beta_2 \frac{dx_{i-1}}{S_2}$$

when

$$S_1 = dx_{i-1}^2 dx_i + dx_i^2 dx_{i-1} \text{ and } S_2 = \frac{dx_{i-1}^2 dx_i + dx_i^2 dx_{i-1}}{2} = \frac{S_1}{2}$$

Consider the upper airfoil from figure 3.9 which has 207 points where the leading edge is at point “i=207” and the trailing edge is at point “i=1”. We are trying to get the solution Δy_i from point i=2 to i=206 and then extrapolate the solution at point i=207 from the 2 previous consecutive points which are i=206 and i=205. The solution at i=1 is fixed to be $\Delta y_1 = 0$ as a boundary condition. A_i, B_i, and C_i are computed from point i=2 to i=206 using dx_i and dx_{i-1} where $dx_i = x_{i+1} - x_i$ and $dx_{i-1} = x_i - x_{i-1}$. Then, the matrix of a system of linear equations is presented below

$$\begin{bmatrix} A_2 & B_2 & C_2 & & & & 0 \\ & A_3 & B_3 & C_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & A_{205} & B_{205} & C_{205} & \\ 0 & & & & A_{206} & B_{206} & C_{206} \end{bmatrix}_{205\times207} \begin{Bmatrix} \Delta y_1 \\ \Delta y_2 \\ \Delta y_3 \\ \vdots \\ \Delta y_{207} \end{Bmatrix}_{207\times1} = \begin{Bmatrix} \Delta C_{p2} \\ \Delta C_{p3} \\ \Delta C_{p4} \\ \vdots \\ \Delta C_{p206} \end{Bmatrix}_{205\times1}$$

Canceling out the first column since $\Delta y_1 = 0$ and modifying the last row by extrapolating

$$\Delta y_{207} = \left(1 - \frac{x_{207} - x_{205}}{x_{206} - x_{205}}\right) \Delta y_{205} + \left(\frac{x_{207} - x_{205}}{x_{206} - x_{205}}\right) \Delta y_{206}$$, we get

$$\begin{bmatrix} B_2 & C_2 & & & 0 \\ A_3 & B_3 & C_3 & & \\ & \ddots & \ddots & \ddots & \\ & & A_{205} & B_{205} & C_{205} \\ 0 & & & A'_{206} & B'_{206} \end{bmatrix}_{205\times205} \begin{Bmatrix} \Delta y_2 \\ \Delta y_3 \\ \Delta y_4 \\ \vdots \\ \Delta y_{206} \end{Bmatrix}_{205\times1} = \begin{Bmatrix} \Delta C_{p2} \\ \Delta C_{p3} \\ \Delta C_{p4} \\ \vdots \\ \Delta C_{p206} \end{Bmatrix}_{205\times1}$$

where $$A'_{206} = A_{206} + C_{206}\left(1 - \frac{x_{207} - x_{205}}{x_{206} - x_{205}}\right)$$

and $$B'_{206} = B_{206} + C_{206} \frac{x_{207} - x_{205}}{x_{206} - x_{205}}$$

Using Thomas's Algorithm, we obtain the solution Δy_2 to Δy_{206} then extrapolating to Δy_{207} from$\Delta y_{207} = \left(1 - \frac{x_{207} - x_{205}}{x_{206} - x_{205}}\right) \Delta y_{205} + \left(\frac{x_{207} - x_{205}}{x_{206} - x_{205}}\right) \Delta y_{206}$. Therefore, we have the solution Δy for all points of the top airfoil.

Now, consider the bottom airfoil from point i=208 to i=414. There are two boundary conditions. First, the leading edge has to be continuous so we have to set $\Delta y_{208} = \Delta y_{207}$. Second, the trailing edge is fixed as the top airfoil so$\Delta y_{414} = \Delta y_1 = 0$. Applying the boundary conditions to the first and last equation of the bottom airfoil, we get

$$\begin{bmatrix} B_{208} & C_{208} & & & \\ A_{209} & B_{209} & C_{209} & & \\ & \ddots & \ddots & \ddots & \\ & & A_{412} & B_{412} & C_{412} \\ & & & A_{413} & B_{413} \end{bmatrix}_{205\times205} \begin{Bmatrix} \Delta y_{208} \\ \Delta y_{209} \\ \Delta y_{210} \\ \vdots \\ \Delta y_{413} \end{Bmatrix}_{205\times1} = \begin{Bmatrix} \Delta C_{p208} - A_{208}\Delta y_{207} \\ \Delta C_{p209} \\ \Delta C_{p210} \\ \vdots \\ \Delta C_{p413} \end{Bmatrix}_{205\times1}$$

Using Thomas's Algorithm, we obtain the solution Δy_{208} toΔy_{413}. Now, we know the solution Δy for all points of the airfoil. Δy are added to the y-coordinates of initial airfoil which is NACA0011, then we get the new airfoil coordinates. The new airfoil is used as an initial airfoil in the next iteration until the new airfoil produces the same C_P distribution as the target airfoil.

For all other cases which are subsonic inviscid flow, transonic inviscid flow, and transonic viscous flow, the methodology is the same. The only two things that need to be changed are the Mach number and the flow solver in FLUENT part.

CHAPTER 4

RESULT

There are two parts which are going to be presented. First, the different results between using regular finite difference method and finite difference method for non-uniform grid are compared. The effect from the constants of MGM equations is also showed in this part. The other part is the validation of the program using finite difference method for non-uniform grid. There are three cases to be demonstrated. The first two cases are subsonic inviscid and subsonic viscous flow at Mach number 0.2 by using the pressure coefficient of NACA0011 as an initial input and the pressure coefficient of NACA2315 as a target. The other case is subsonic viscous flow at Mach number 0.6 by using the pressure coefficient of NACA0012 as an initial input and the pressure coefficient of NACA4415 as a target. After all the results, the transonic viscous flow at Mach number 0.8 is presented.

4.1 The comparison results between two methodologies

By using the regular finite difference method, grids must be divided equally. Therefore, at the leading edge, the shape is not smooth if there is not enough number of grid points along airfoil surface. In order to make the shape of the leading edge smooth, many more grids need to be added for entire airfoil surface, grids cannot be added only on the leading edge zone because Δx needs to be constant. This results in the unnecessary fine grids at the other zone of airfoil surface, which increase the computational time in FLUENT. Figure 4.1 shows the unnecessary fine grids in order to make leading edge smooth. On the other hand, by using the finite difference method for a non-uniform grid, the grids added at the leading edge zone only to make the smooth leading edge. Therefore, there are fewer grid points in the entire airfoil surface such that less computational time is consumed. Figure 4.2 shows the mesh using un-uniform grids along the airfoil surface.

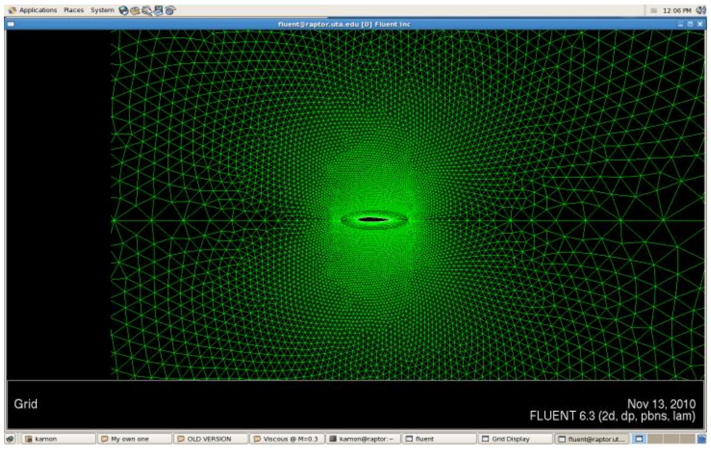


Figure 4.1 Unnecessary fine mesh in order to make leading edge smooth

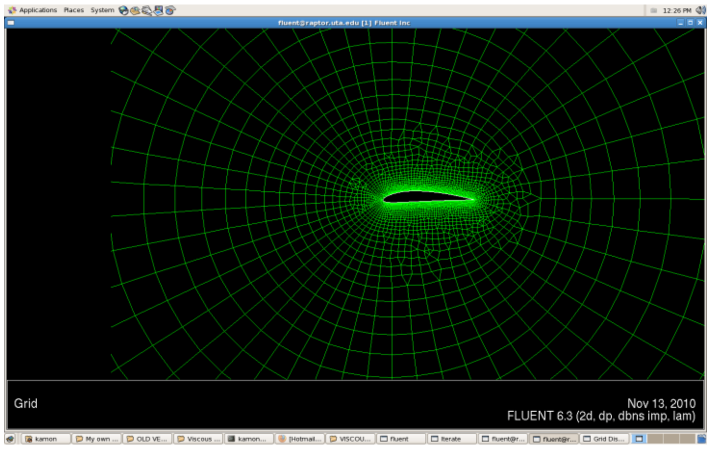


Figure 4.2 A mesh using non-uniform grids along the airfoil surface

Generally, airfoil coordinates are not uniform when using an airfoil coordinate generator. To use the regular finite difference method, fitting curve for uniform coordinates is required. By fitting curve, we are definitely adding some error to the calculation. It is possible to make the method diverge if the error is big enough, especially when the pressure distribution has a sophisticated feature. On the contrary, there is no any problem using finite difference method for a non-uniform grid. Figure 103 illustrates the divergence when the error from fitting curve is too much. The divergence occurs with the upper airfoil because the Cp distribution of the upper airfoil is a sharp curve; therefore, it is difficult to fit the curve perfectly. However, the lower airfoil has a smooth Cp distribution so the error from fitting the curve is not too much to make the iteration diverge.

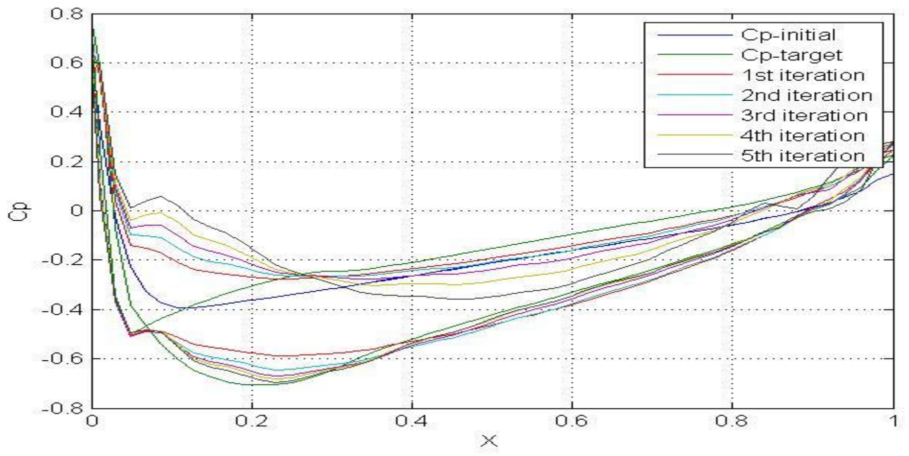


Figure 4.3 Illustration of divergence occurring when the error from fitting curve is too big

4.2 The Validation

In this section, the Inverse Design of Airfoils Using a Flexible Membrane Method is tested in three cases. The first two cases are subsonic inviscid and subsonic viscous flow at Mach number 0.2 by using the pressure coefficient of NACA0011 as an initial input and the pressure coefficient of NACA2315 as a target. The other case is subsonic viscous flow at Mach number 0.6 by using the pressure coefficient of NACA0012 as an initial input and the pressure coefficient of NACA4412 as a target.

4.2.1 The inviscid Subsonic Case at Mach 0.2

In this case, NACA0011 is chosen as an initial airfoil to use its Cp distribution as an input to the program, and Cp distribution of NACA2315 is used as a target to validate the program. The Cp distributions of both airfoils are presented in Figure 4.4.

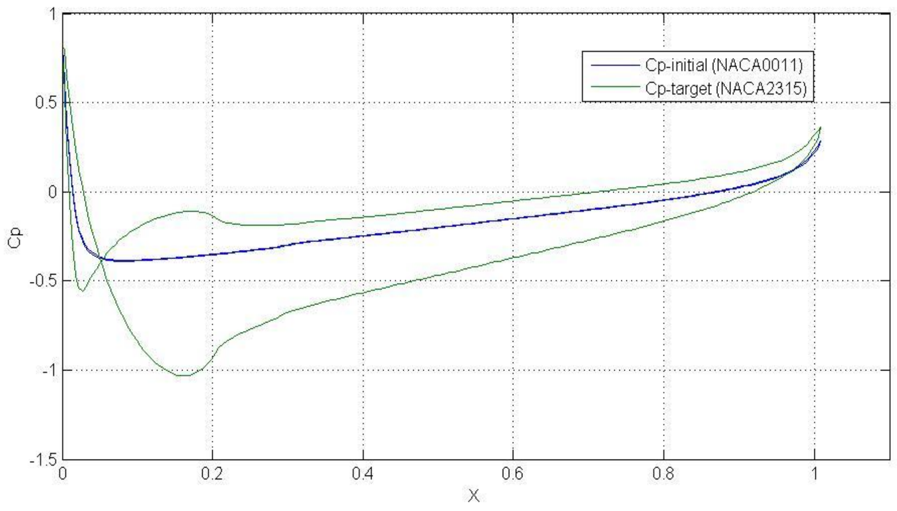


Figure 4.4 Cp Distributions of NACA0011 and NACA2315 for inviscid flow at M=0.2

Figure 4.5 shows the grid generated from the moving grid program which has 4795 nodes, 4826 cells, 9621 faces. The number of node, cell, and faces of grids for any airfoil will be the same when using “mvg” program. By using FLUENT as a flow solver, choosing pressure base solver and ideal gas, setting inviscid flow with Mach number equals 0.2, setting pressure-velocity coupling to SIMPLE, using first order upwind to discretize the flow equations with Courant Number equals 5, and using the coefficients $\beta_0 = 30, \beta_1 = 0.1, \beta_2 = 4$ in MGM equation, 45 iterations are used to get the good target airfoil with error at all grid points are less than 5 percent. The absolute criteria of residuals in FLUENT are set to be 10e-6. When residuals reach the absolute criteria, a solution of the flow is obtained. Figure 4.6 shows the monitor of residuals in FLUENT for the inviscid at M=0.2 case. The Cp distribution at the first, tenth, twentieth, thirtieth, and forty fifth iteration are presented in Figure 4.7. Then the airfoil shapes at those iterations are also shown in Figure 4.8. Noticeably, after 15 iterations, the Cp distribution changes much slower than the beginning of the iterations. At this state, most of the points have already converged to the target values except at the sharp turning parts of Cp distribution as shown in Figure 4.9 at the fifteenth iteration. Therefore, in order to get a faster rate of convergence, multiplier should be applied to the force vector ($[\Delta C_P]$). In this case, ten is applied. After multiplying the force vector once, the multiplier should be removed to let the solution stabilize again. Then if Cp distribution seems to converge slowly, multiplier may be needed occasionally. If the multiplier is not removed after applied, divergence will occur. Figure 4.10 shows the comparison of Cp distribution at the fifteenth iteration between applying multiplier and not.

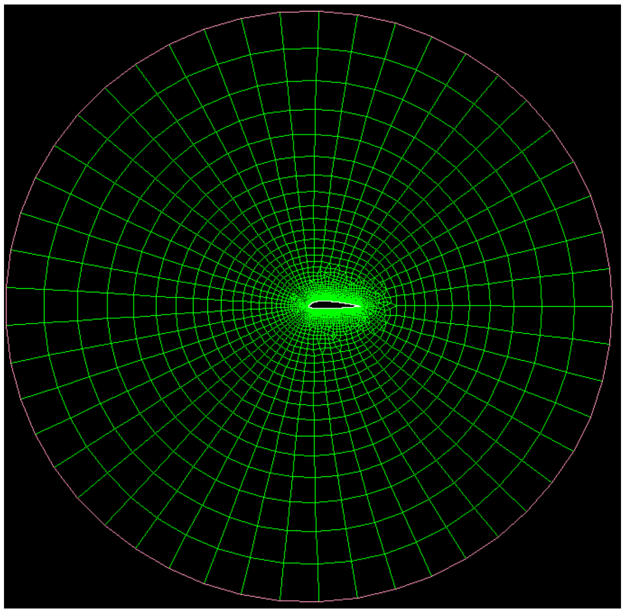


Figure 4.5 A grid generated from the moving grid program

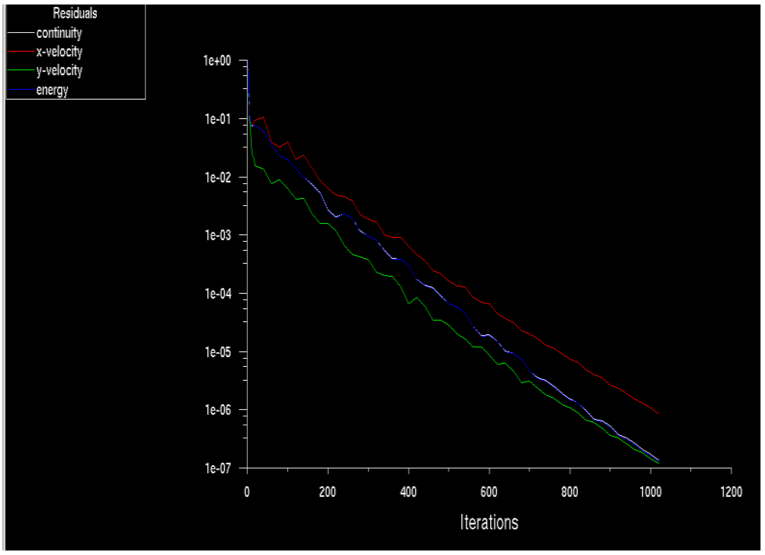


Figure 4.6 Residual monitor for inviscid at M=0.2 case

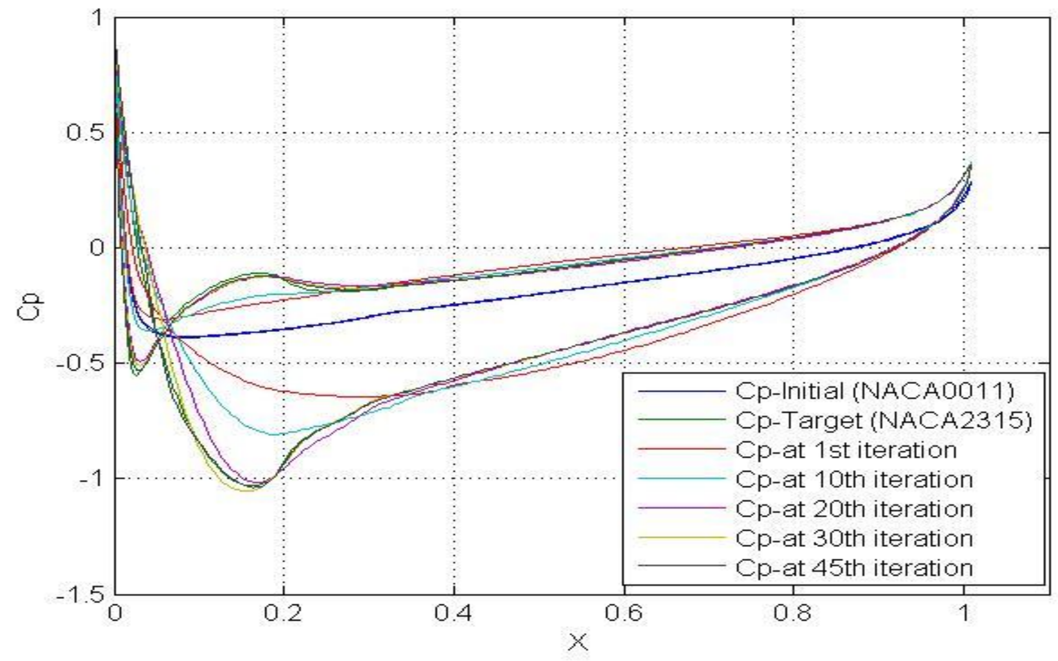


Figure 4.7 Cp Distributions of airfoils during iteration of inviscid M=0.2 case

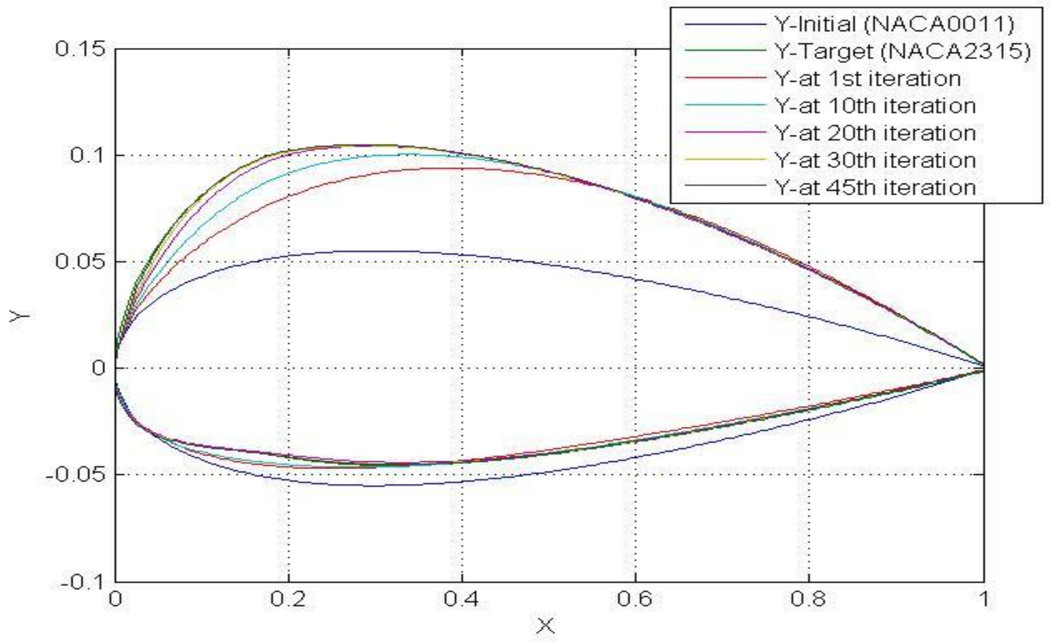


Figure 4.8 Airfoil Shapes during iteration of inviscid M=0.2 case

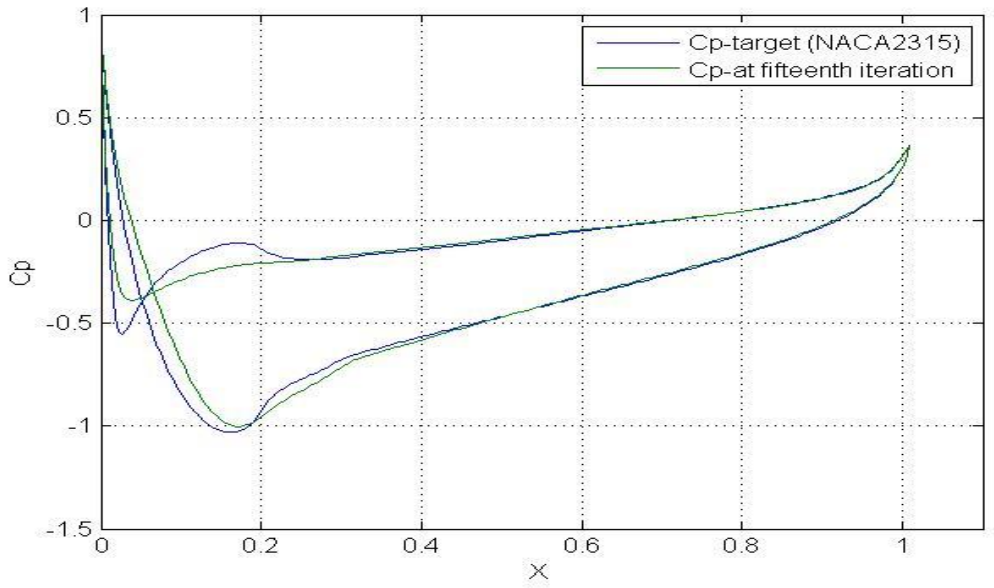


Figure 4.9 Cp distributions of target airfoil and airfoil at fifteenth iteration

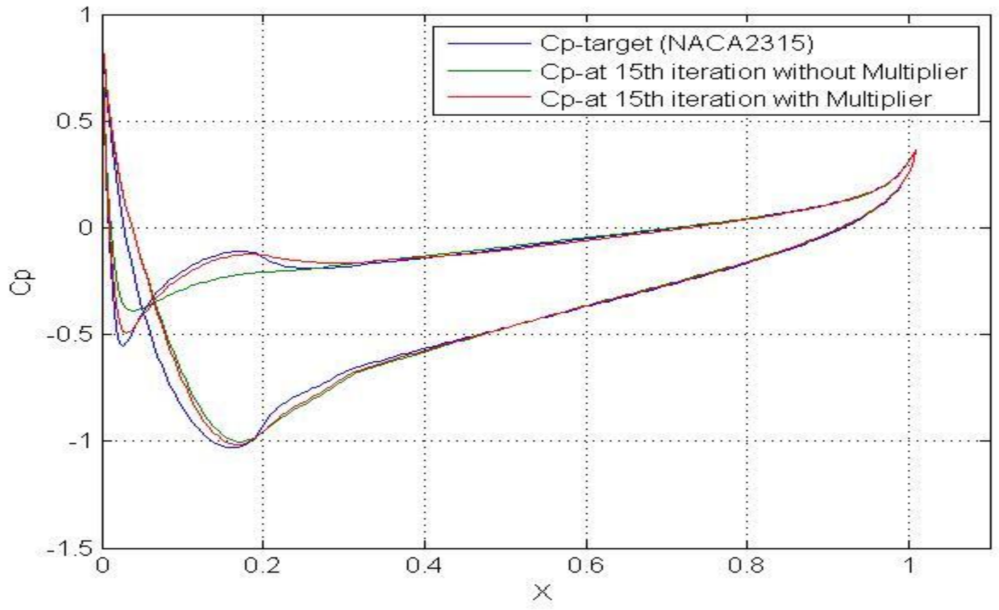


Figure 4.10 Cp distributions of airfoil at fifteenth iteration with and without multiplier

After the 45th iteration, the errors of all points are less than 5 percent. Therefore, the designed goal is reached. Figure 4.11 shows the errors of all points at the 45th iteration. Figure 4.12 shows the shape of the designed airfoil compared with the target airfoil NACA2315 and Figure 4.13 shows the Cp distributions of the designed airfoil compared with the target Cp distribution of NACA2315. Figure 4.14 shows the norm error which indicates that the error decreases when the number of iteration increase. It means that the system is converging. Note that, sometimes, the error drops immediately from the previous step that is because of the multiplier which proved to increase the convergent rate. Figure 4.15 shows the velocity flow field of the initial airfoil and target airfoil.

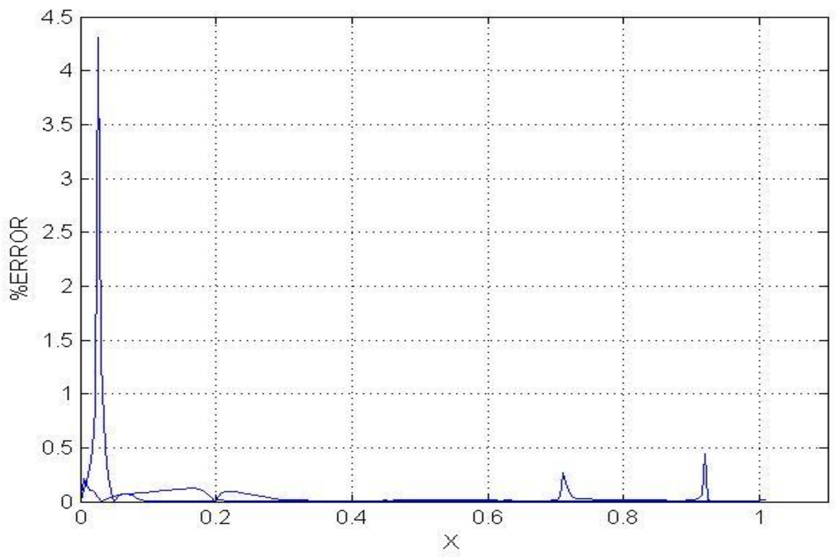


Figure 4.11 The errors at all points at 45th iteration

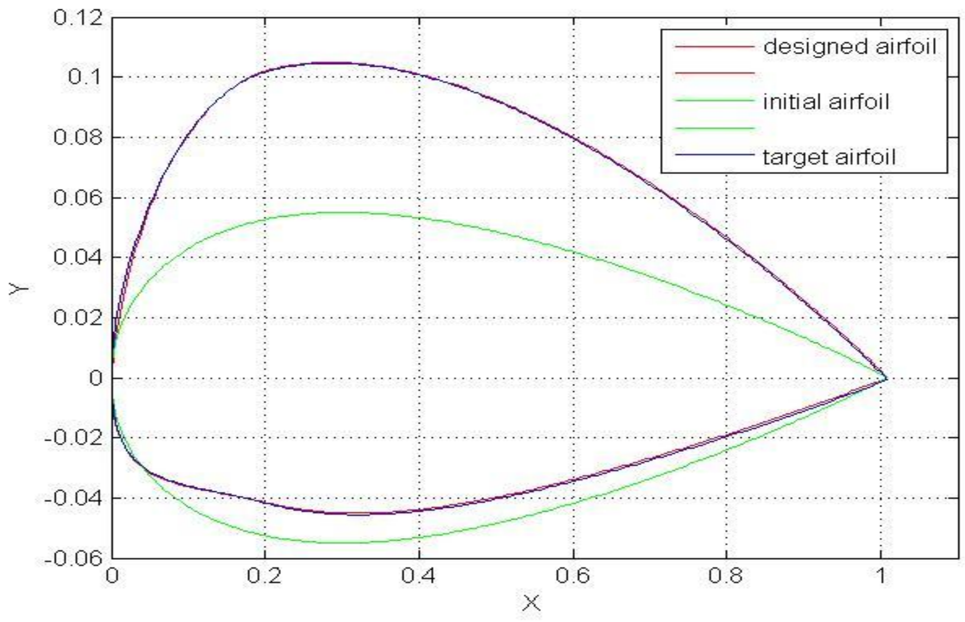


Figure 4.12 The shape of the designed airfoil compared with the target airfoil(NACA2315) and initial airfoil (NACA0011)

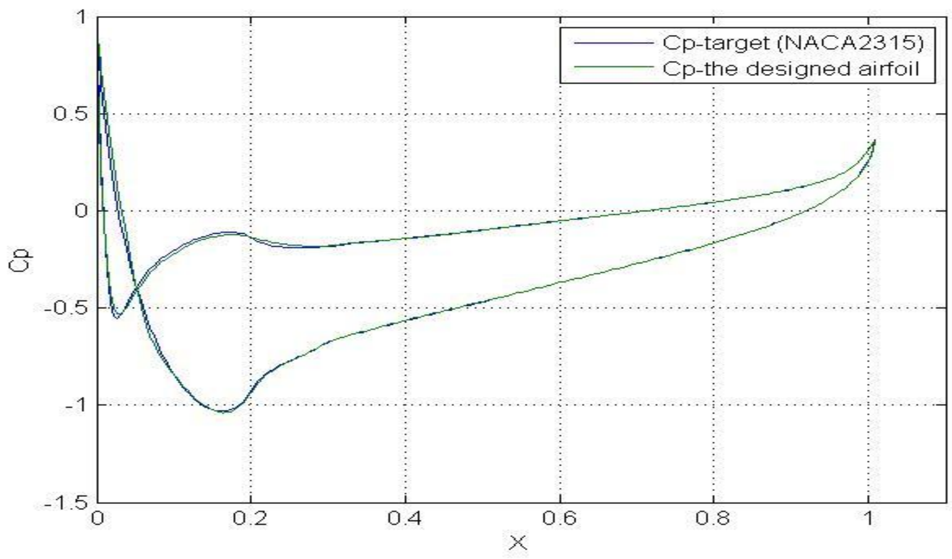


Figure 4.13 The Cp distributions of the designed airfoil compared with the target (NACA2315)

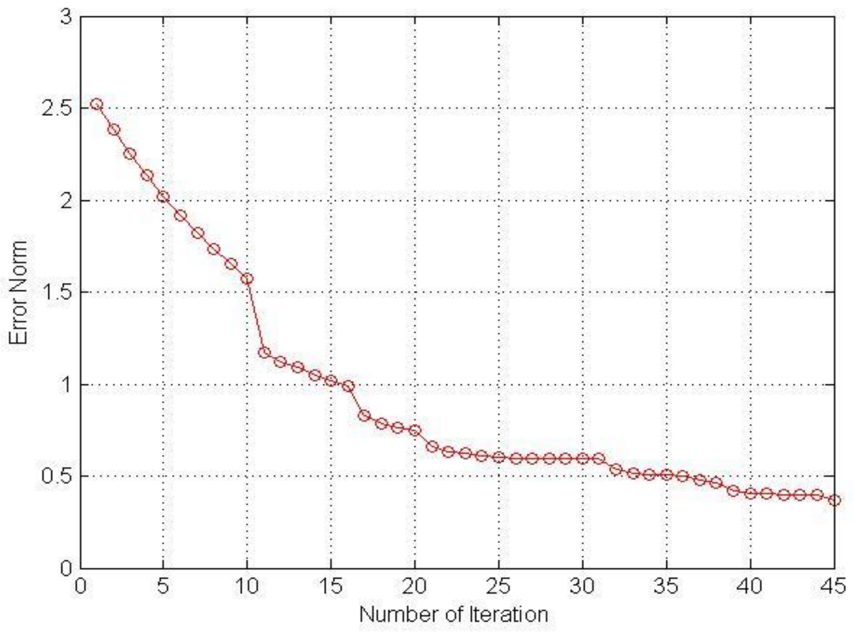
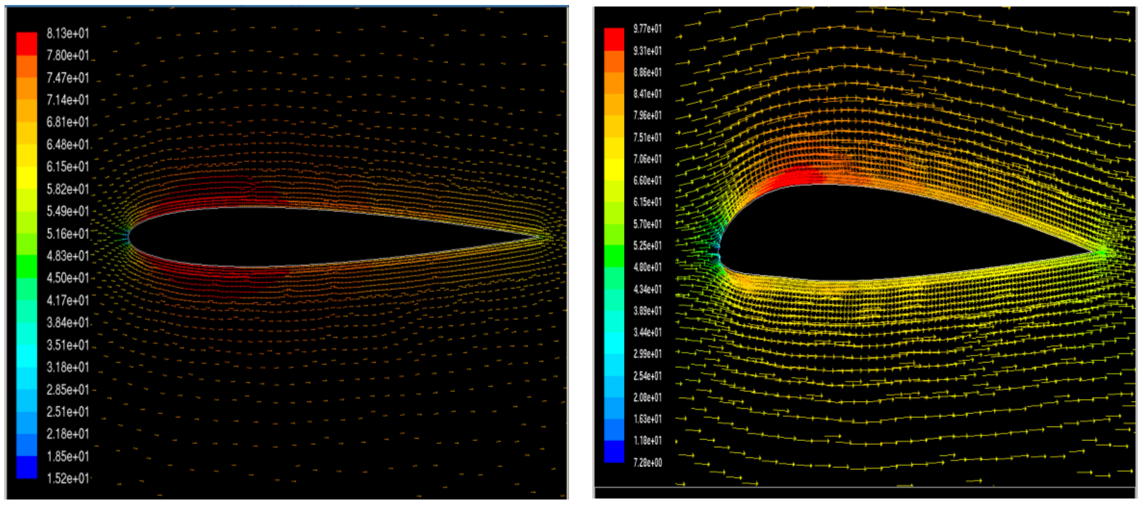


Figure 4.14 The norm error for inviscid at M=0.2 case



(a) (b)

Figure 4.15 (a) Velocity flow field of the initial airfoil NACA0011 for inviscid @M=0.2
(b) Velocity flow field of the target airfoil NACA2315 for inviscid @M=0.2

4.2.2 The viscous Subsonic Case at Mach 0.2

In this case, an initial airfoil is also NACA0011 and the pressure coefficient of NACA2315 is used as a target. The Cp distributions of both initial and target airfoils are presented in Figure 4.16. By using FLUENT as a flow solver, choosing pressure base solver and ideal gas, setting viscous (laminar) flow with Mach number equals 0.2, setting pressure-velocity coupling to SIMPLE, using first order upwind to discretize the flow equations with Courant Number equals 5, and using the coefficients $\beta_0 = 30, \beta_1 = 0.1, \beta_2 = 4$ in MGM equation as the first case. Forty six iterations are used to get the good target airfoil with error at all grid points that are less than 5 percent. The Cp distribution at the first, tenth, twentieth, thirtieth, and forty sixth iteration are presented in Figure 4.17. Then the airfoil shapes at those iterations are also shown in Figure 4.18. In this case, the error is less than the inviscid case. The Cp distribution of the designed airfoil is shown in Figure 4.19 and the shape of the designed airfoil is shown in Figure 4.20. The errors at each grid points are indicated in Figure 4.21.

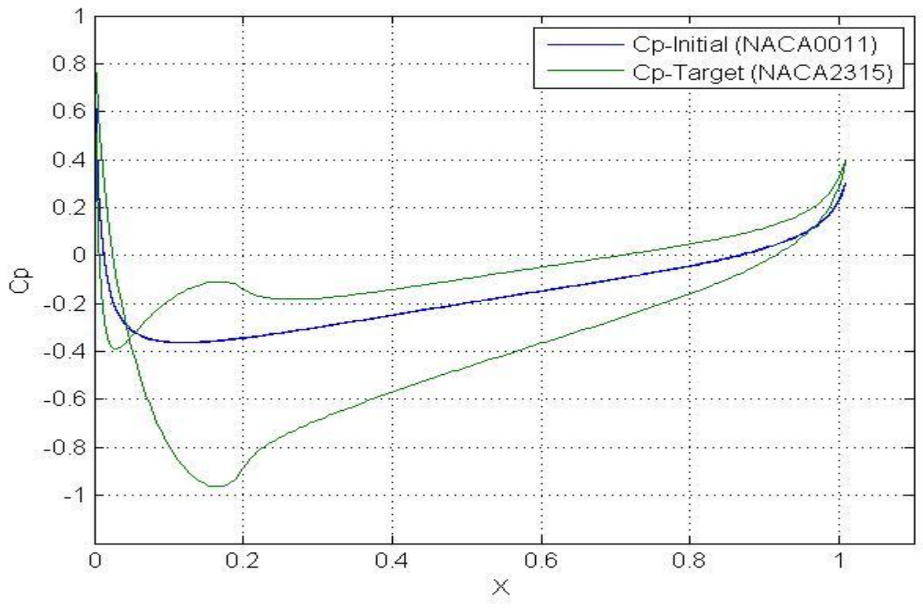


Figure 4.16 Cp Distributions of NACA0011 and NACA2315 for viscous flow at M=0.2

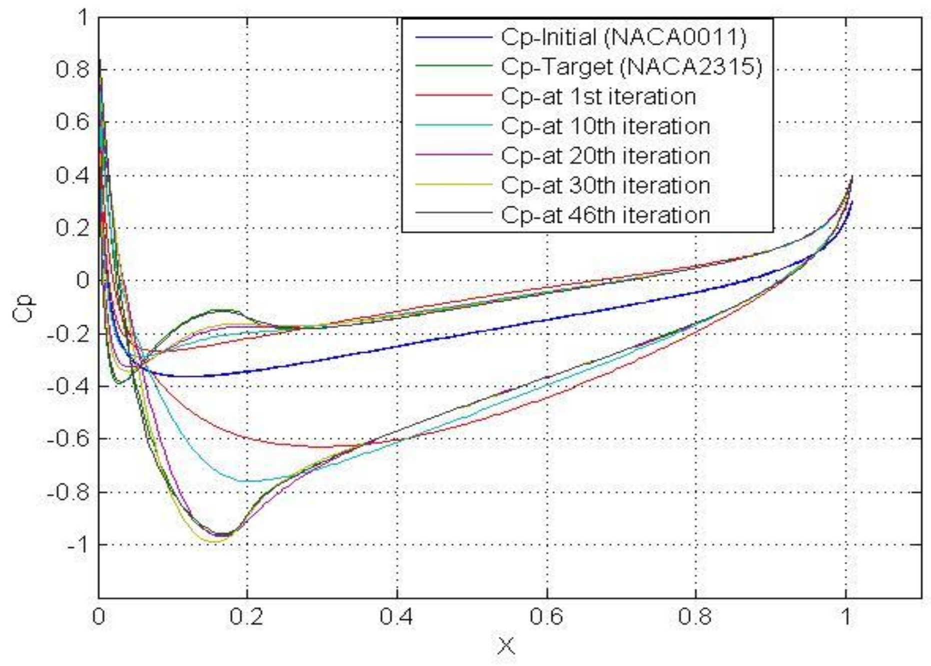


Figure 4.17 Cp Distributions of airfoils during iteration of viscous M=0.2 case

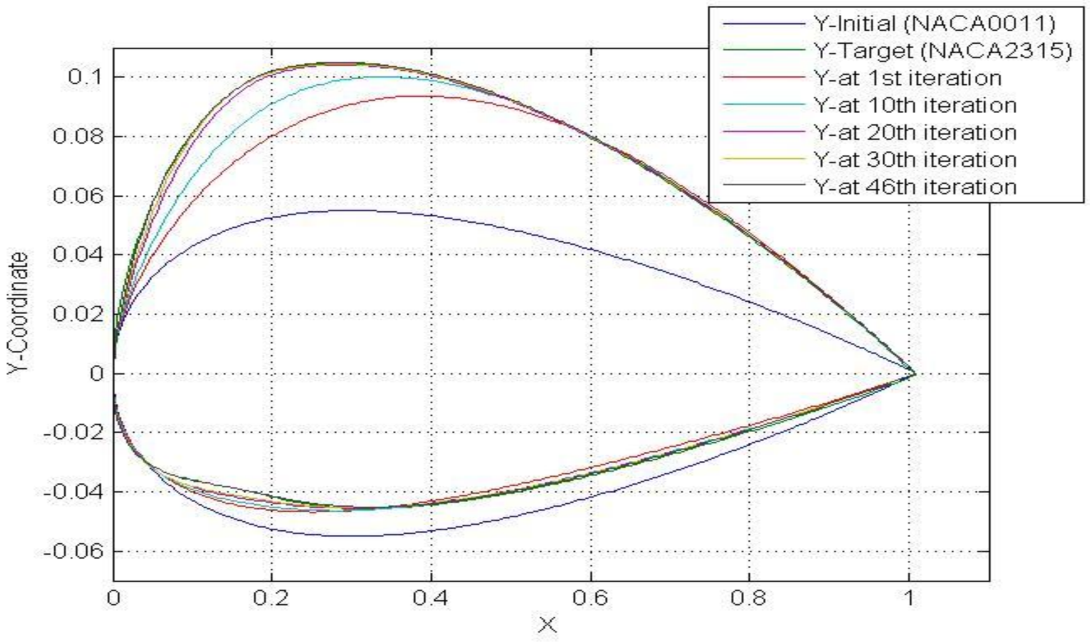


Figure 4.18 Airfoil Shapes during iteration of viscous M=0.2 case

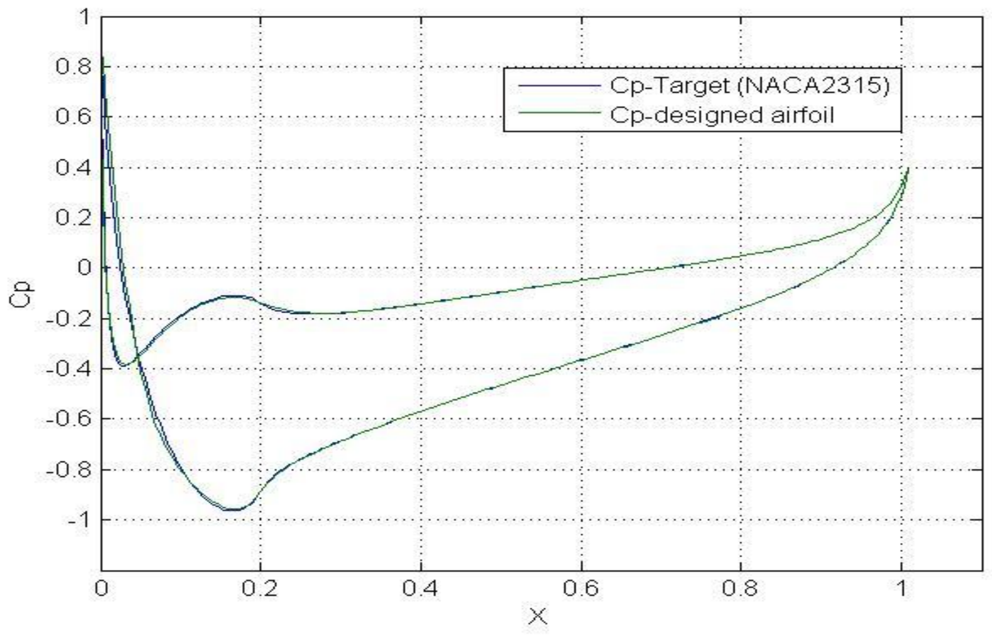


Figure 4.19 The Cp distributions of the designed airfoil compared with the target (NACA2315) airfoil for viscous M=0.2 case

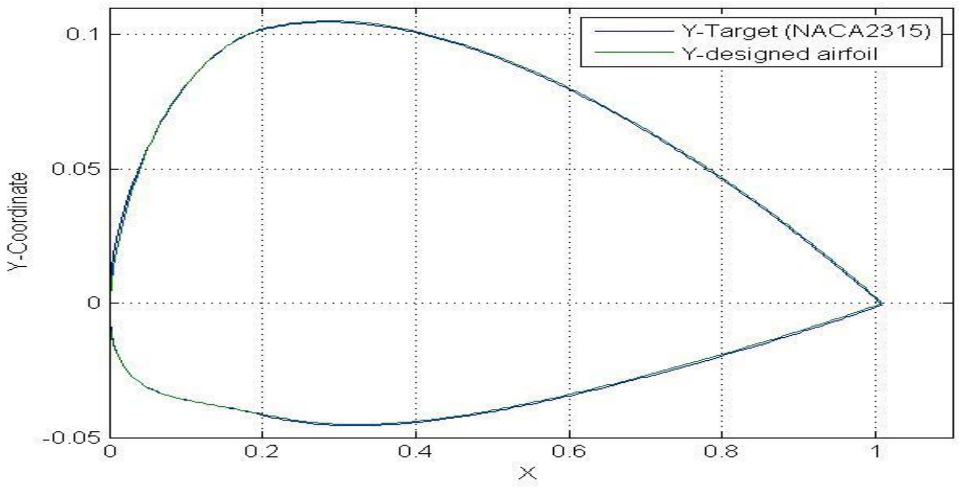


Figure 4.20 The shape of the designed airfoil compared with the target (NACA2315) airfoil for viscous M=0.2 case

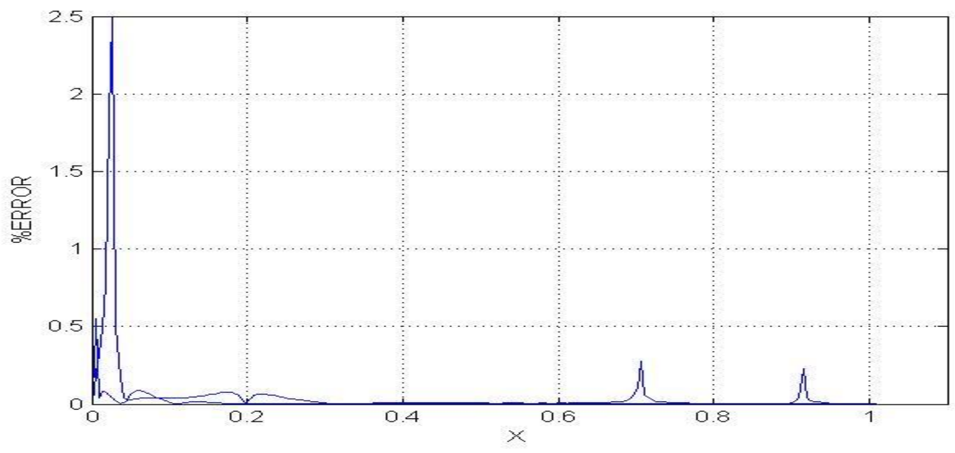


Figure 4.21 The errors of all grid points of the designed result for viscous M=0.2 case

Figure 4.22 shows the norm error in the viscous at M=0.2 case. The error decreases when the number of iteration increase in almost the same rate as inviscid M=0.2 case. According to Figure 4.11 and 4.21, there are some points that the error is much higher than the others. The reason is at those points, the reference value which is the divider is very close to zero; therefore, even the residual is acceptable, error is still high.

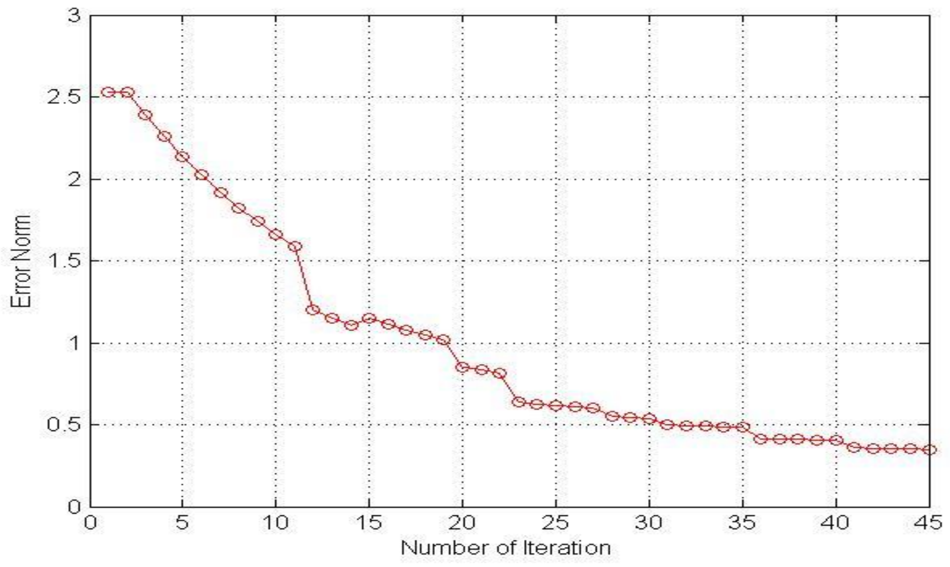


Figure 4.22 The norm error for viscous at M=0.2 case

4.2.3 The viscous Subsonic Case at Mach 0.6

In this case, an initial airfoil is changed from NACA0011 to NACA0012 and the pressure coefficient of NACA4412 is used as a target. The Cp distributions of both airfoils are presented in Figure 4.23.

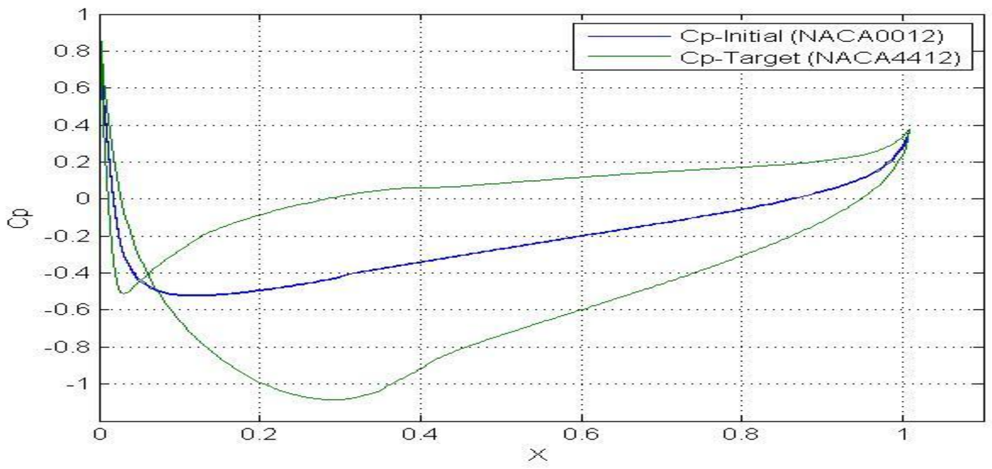


Figure 4.23 Cp Distributions of NACA0012 and NACA4412 in viscous M=0.6 case

By using FLUENT as a flow solver, setting viscous flow with Mach number equals 0.6 and using the coefficients $\beta_0 = 25, \beta_1 = 0.1, \beta_2 = 4$ in MGM equation. Twenty one iterations are used to get the good target airfoil with error at all grid points are less than 5 percent. The Cp distribution at the first, fifth, tenth, fifteenth, and twenty first iteration are presented in Figure 4.24. Then the airfoil shapes at those iterations are also shown in Figure 4.25. Notice that in this case, 21 iterations were used to get the target result. It used less iteration time than the first two cases. One of the reasons is that the Cp distribution of NACA4412, which is the target, is smoother than the Cp distribution of NACA2315. The Cp distribution of the designed airfoil is shown in Figure 4.26 and the shape of the designed airfoil is shown in Figure 4.27. The errors at each grid points are indicated in Figure 4.28.

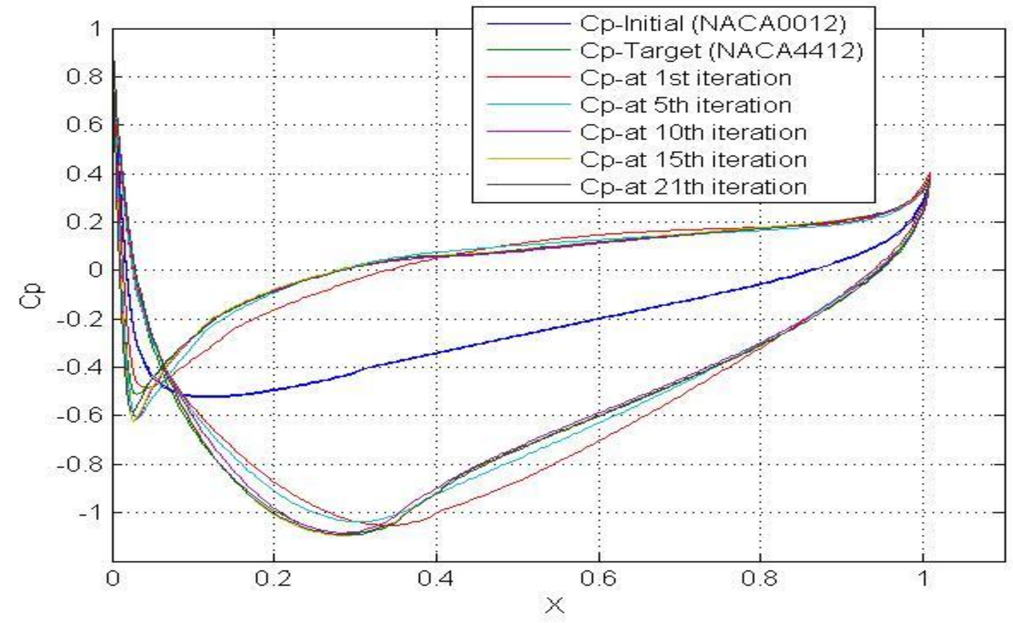


Figure 4.24 Cp Distributions of airfoils during iteration of viscous M=0.6 case

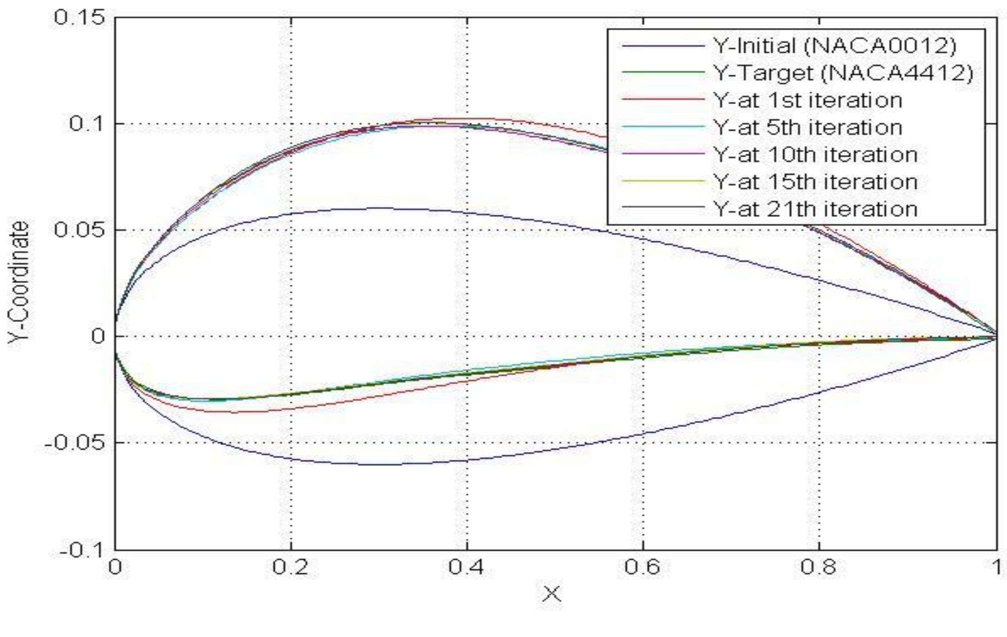


Figure 4.25 Shape of airfoils during iteration of viscous M=0.6 case

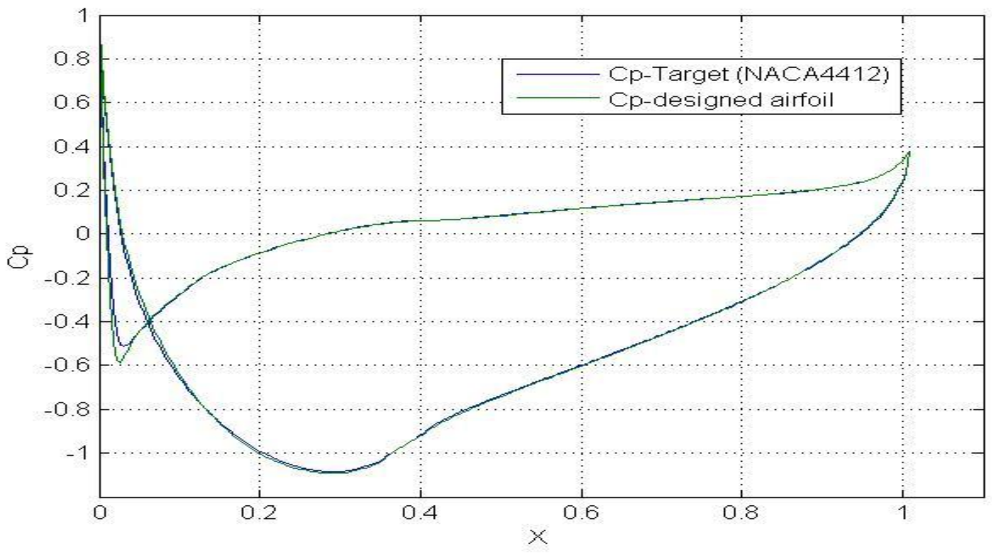


Figure 4.26 The Cp distributions of the designed airfoil compared with the target (NACA4412) airfoil for viscous M=0.6 case

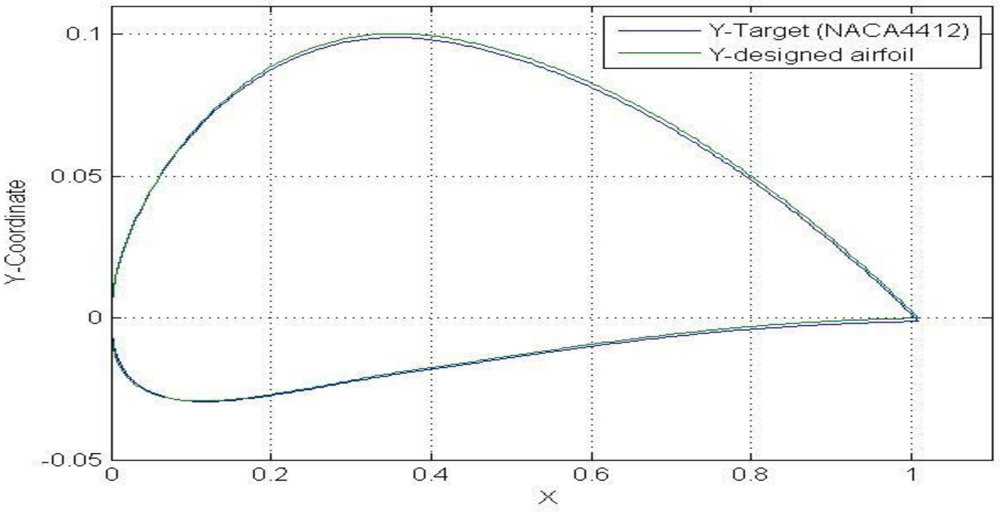


Figure 4.27 The shape of the designed airfoil compared with the target (NACA4412) airfoil for viscous M=0.6 case

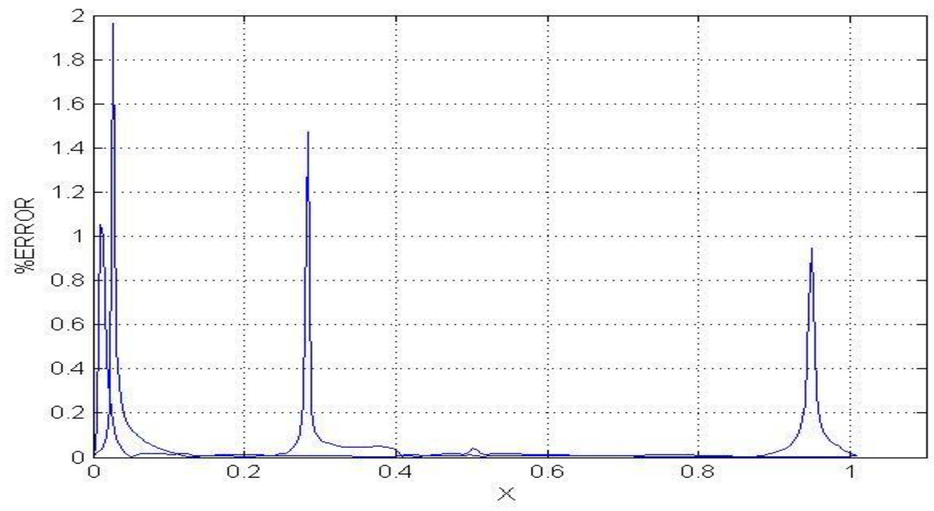


Figure 4.28 The errors of all grid points of the designed result for viscous M=0.6 case

Figure 4.29 shows the norm error in the viscous at M=0.6 case. The error decreases when the number of iteration increases. Figure 4.30 shows the velocity flow field of the initial airfoil NACA0012 and target airfoil NACA4412 for viscous at M=0.6 case. The pressure contours of the initial airfoil NACA0012 and NACA4412 are presented as well in Figure 4.31for viscous at M=0.6 case.

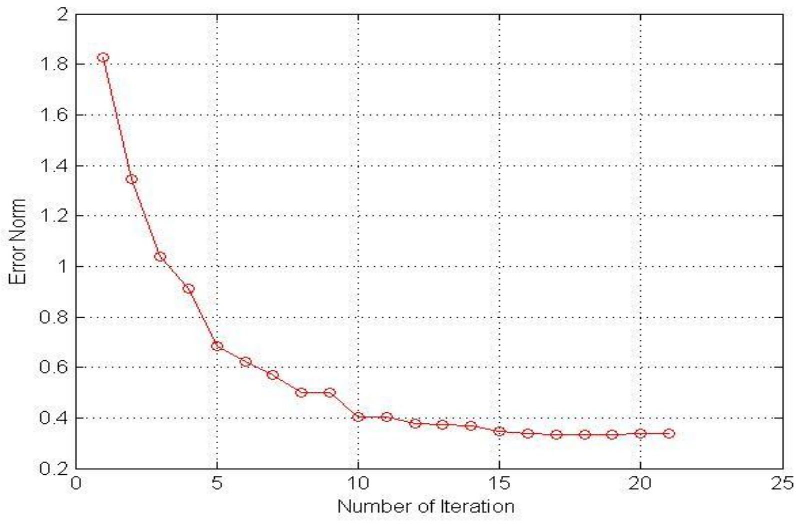
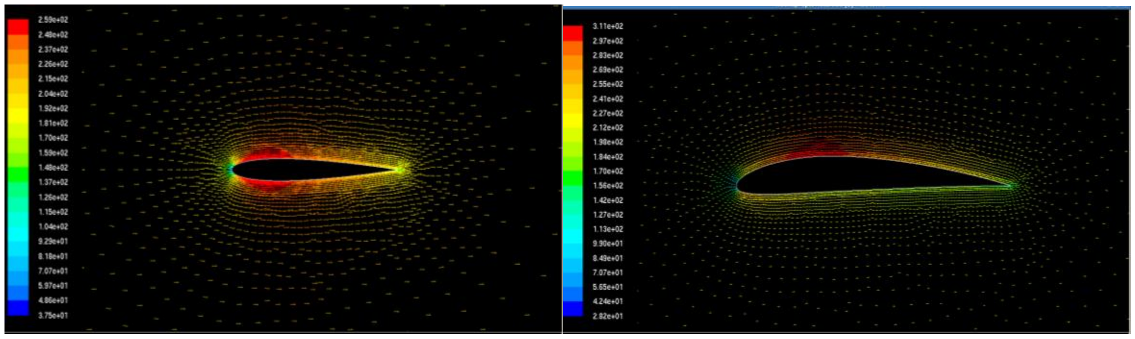
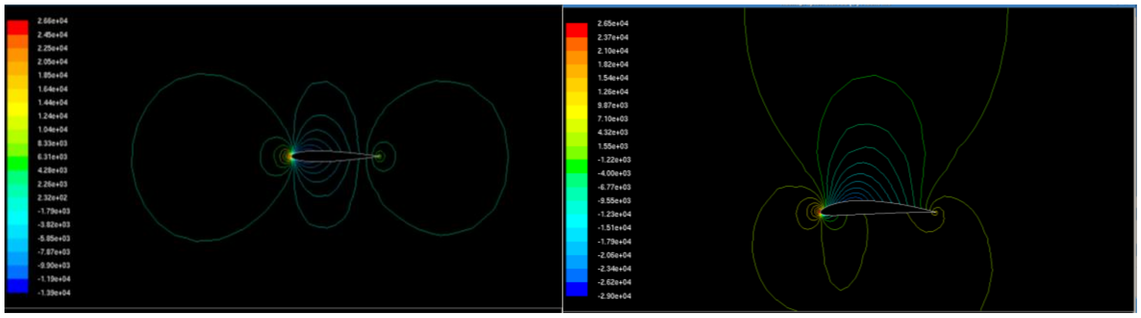


Figure 4.29 The norm error for viscous at M=0.6 case



(a) (b)

Figure 4.30 (a) Velocity flow field of the initial airfoil NACA0012 for viscous @M=0.6
(b) Velocity flow field of the target airfoil NACA4412 for viscous @M=0.6



(a) (b)

Figure 4.31 (a) Pressure contour of the initial airfoil NACA0012 for viscous @M=0.6
(b) Pressure contour of the target airfoil NACA4412 for viscous @M=0.6

4.2.4 The viscous Transonic Case at Mach 0.8

In this case, NACA0012 is used as an initial airfoil and NACA4412 is used as a target airfoil. The Cp distributions of both initial and target airfoils are demonstrated in Figure 4.32. For transonic case, local shock waves usually occur along both airfoil surfaces. Shock wave makes the Cp distribution has the discontinuity and it affects the convergence of the system. When the discontinuity occurs to the Cp distribution, it is more difficult to find MGM parameters that make the iterations converge. Sometimes, airfoil shapes when iterating change to those which make separations occur behind shocks; therefore, in transonic regime, the Cp distributions are very vulnerable usually result in divergence of the iteration.

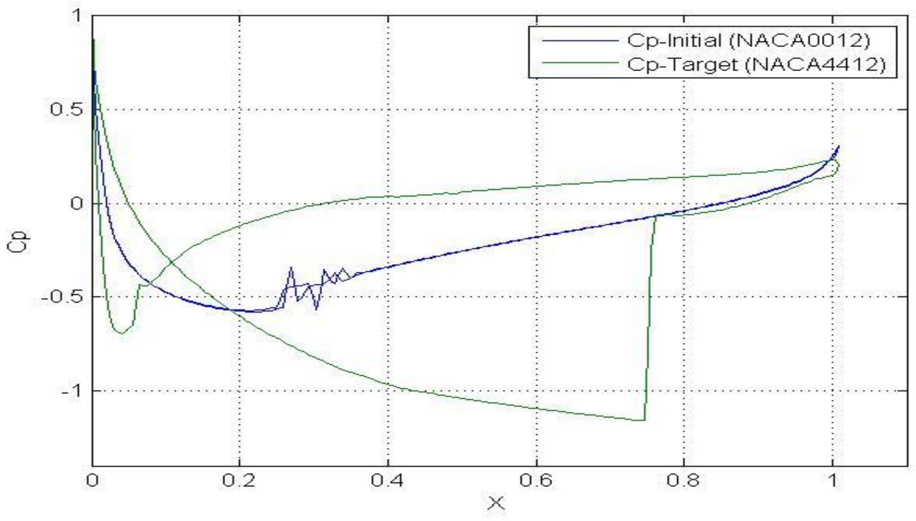


Figure 4.32 Cp Distributions of NACA0012 and NACA4412 of viscous M=0.8 case

By using FLUENT program as a flow solver setting viscous flow with Mach number equals 0.8 and using the coefficients $\beta_0 = 20, \beta_1 = 0.001, \beta_2 = 4$ in MGM equation, 33 iterations are used to get the good target airfoil with error at all grid points are less than 5 percent. The Cp distribution at the first, fifth, tenth, fifteenth, and thirty-third iteration are presented in Figure 4.33. Then the airfoil shapes at those iterations are also shown in Figure 4.34. The Cp distribution of the designed airfoil is shown in Figure 4.35 and the shape of the designed airfoil is shown in Figure 4.36. The errors at each grid points are indicated in Figure 4.37. In order to show the rate of convergence, the error norm is presented in Figure 4.38. Note that, in Figure 4.38, the error norm shows that the iterations seem to diverge at the first four iterations but it converged finally. At the first four iterations, the shape of new airfoils affects the separations to strongly occur so the Cp distributions are vulnerable when shock waves occur and it may make the iterations diverge very easily.

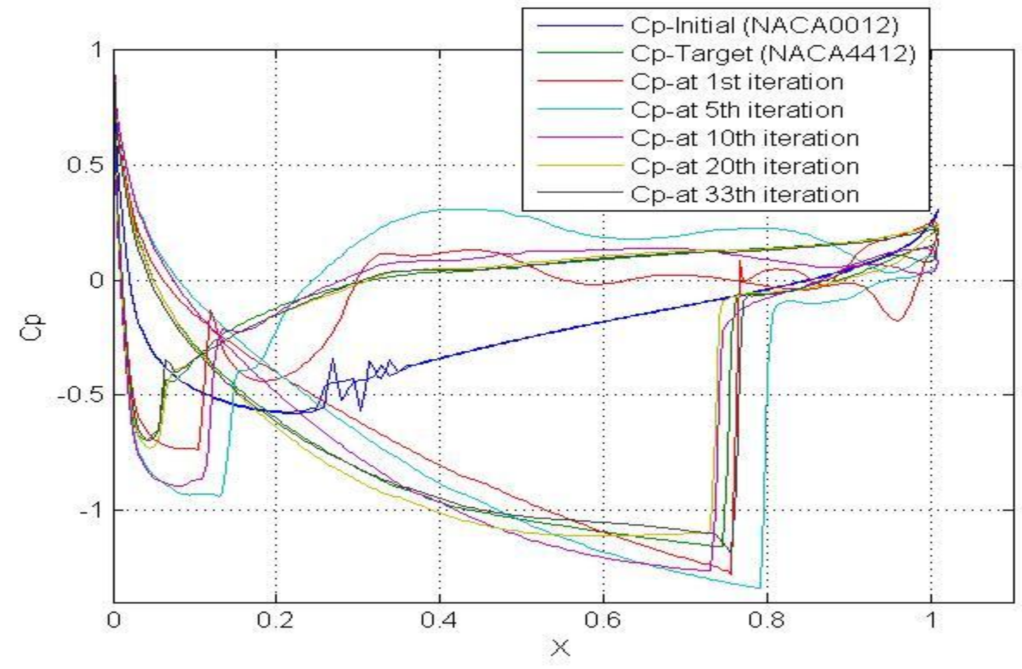


Figure 4.33 Cp Distributions of airfoils during iteration of viscous M=0.8 case

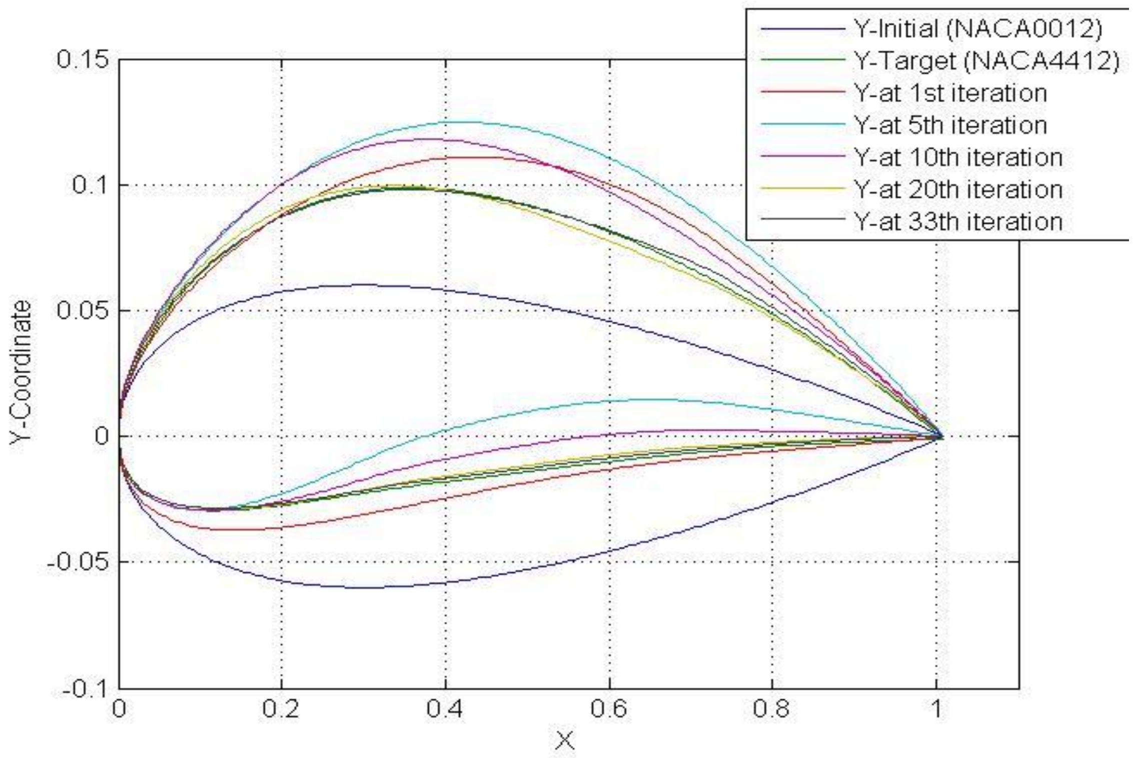


Figure 4.34 Shape of airfoils during iteration of viscous M=0.8 case

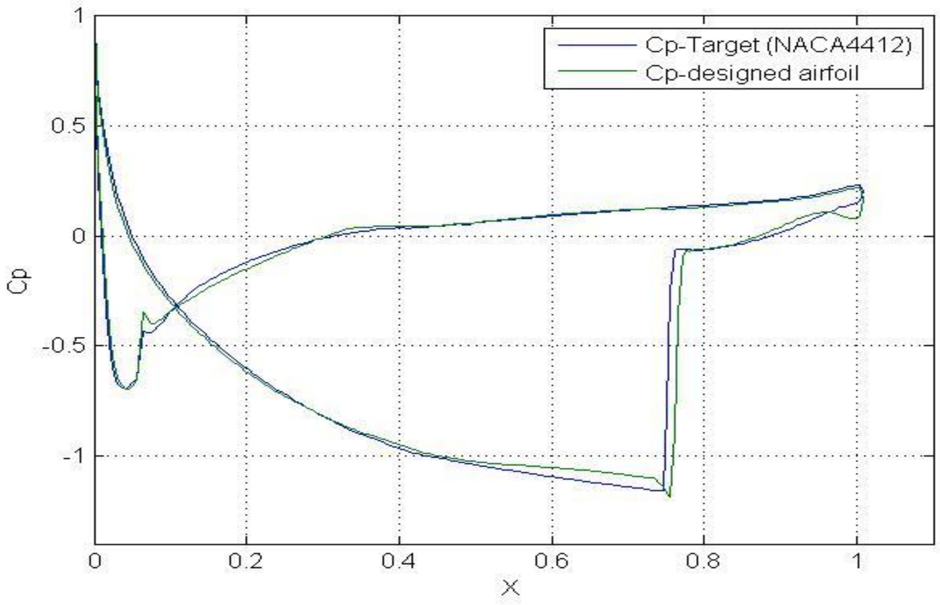


Figure 4.35 The Cp distributions of the designed airfoil compared with the target (NACA4412) airfoil for viscous M=0.8 case

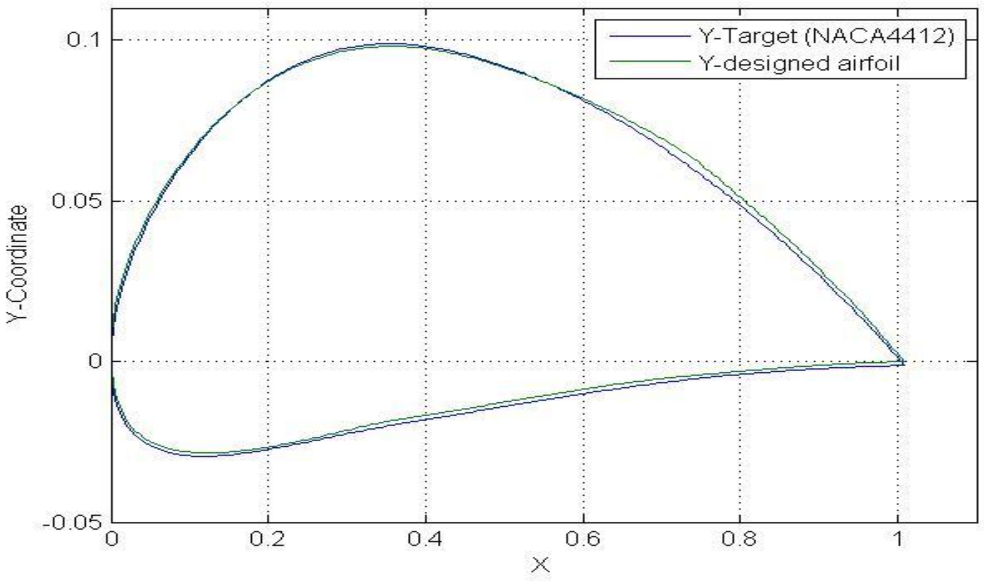


Figure 4.36 The shape of the designed airfoil compared with the target (NACA4412) airfoil for viscous M=0.8 case

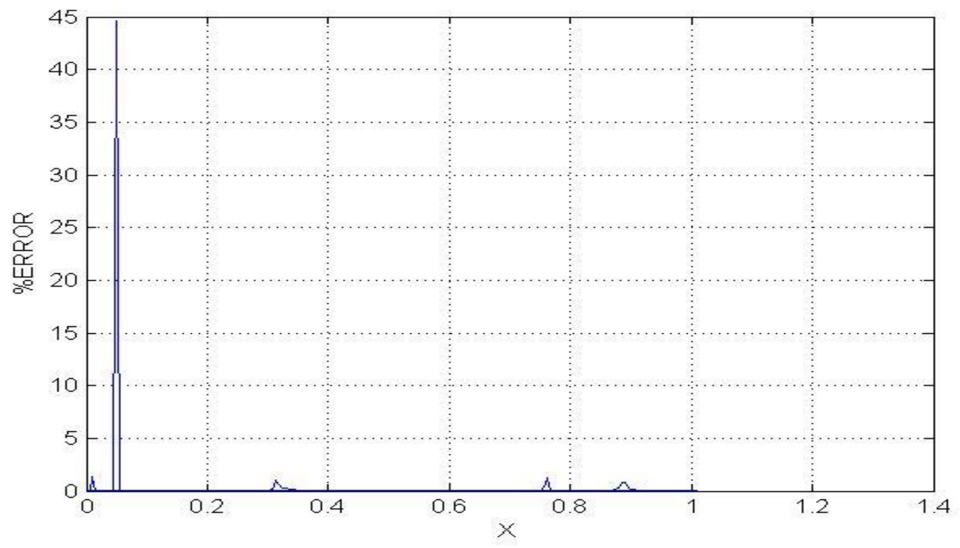


Figure 4.37 The errors of all grid points of the designed result for viscous M=0.8 case

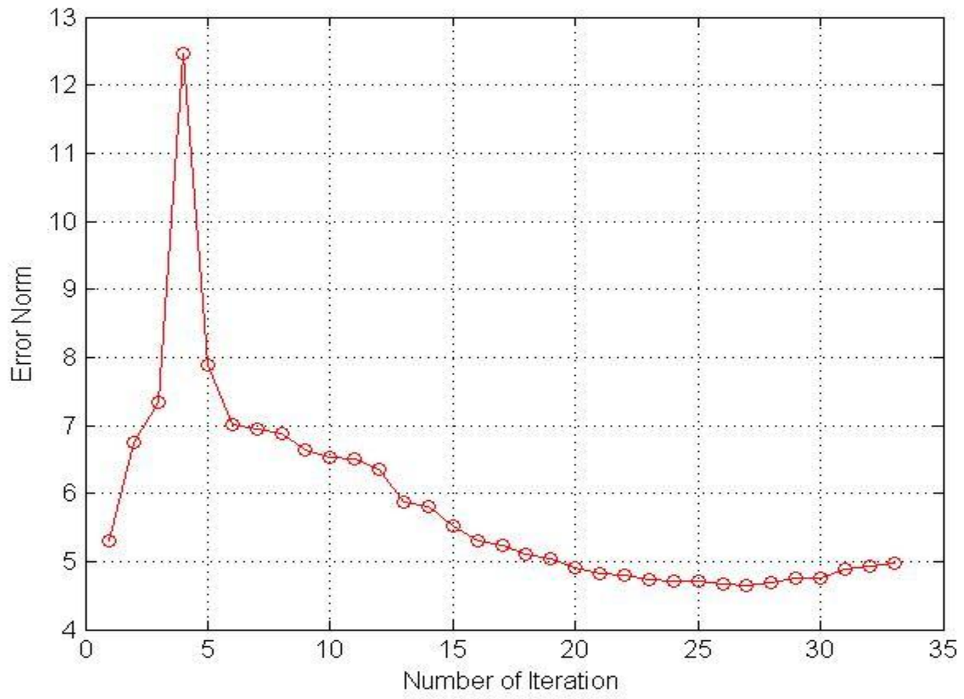
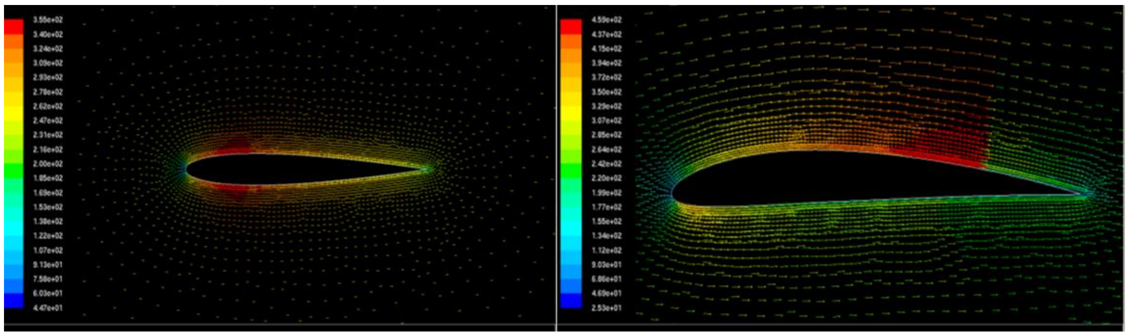


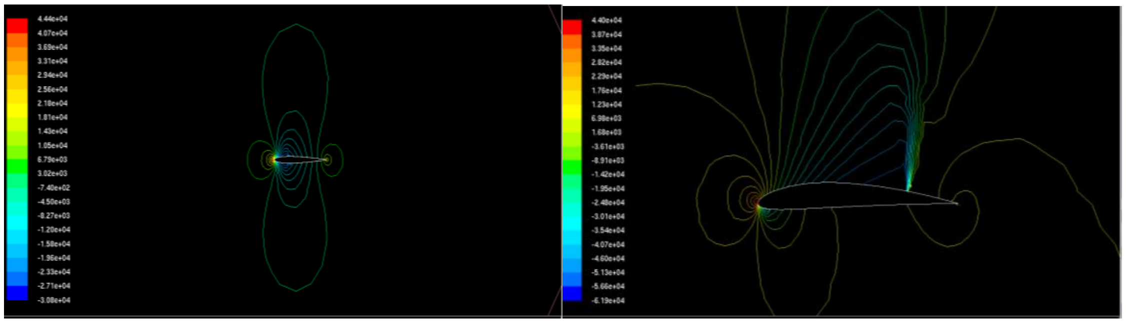
Figure 4.38 The norm error for viscous at M=0.8 case

Figure 4.39 shows the velocity flow field of the initial airfoil NACA0012 and target airfoil NACA4412 for viscous at M=0.8 case. Small shock waves occur on both sides of the initial airfoil but for the target airfoil, there is a big shock wave occurs on the upper airfoil only. The pressure contours of the initial airfoil NACA0012 and NACA4412 are presented in Figure 4.40 as well for viscous at M=0.8 case. A big pressure gradient can be obviously seen at the shock wave location.



(a) (b)

Figure 4.39 (a) Velocity flow field of the initial airfoil NACA0012 for viscous @M=0.8
(b) Velocity flow field of the target airfoil NACA4412 for viscous @M=0.8



(a) (b)

Figure 4.40 (a) Pressure contour of the initial airfoil NACA0012 for viscous @M=0.8
(b) Pressure contour of the target airfoil NACA4412 for viscous @M=0.8

4.3 Shock Free Airfoil Design

4.3.1 Removing Shock Wave at Mach 0.7 using NACA4412 as an initial airfoil

Consider the initial airfoil NACA4412 at Mach number 0.7, a shock wave locally occurs on the upper airfoil although the free stream Mach number is still in subsonic regime. Figure 4.41 shows the velocity flow field with the shock wave on the upper airfoil. The pressure field is shown in figure 4.42 to indicate the shock wave on the upper airfoil as well. The shock wave influents the discontinuity to the Cp distribution as shown in Figure 4.43.

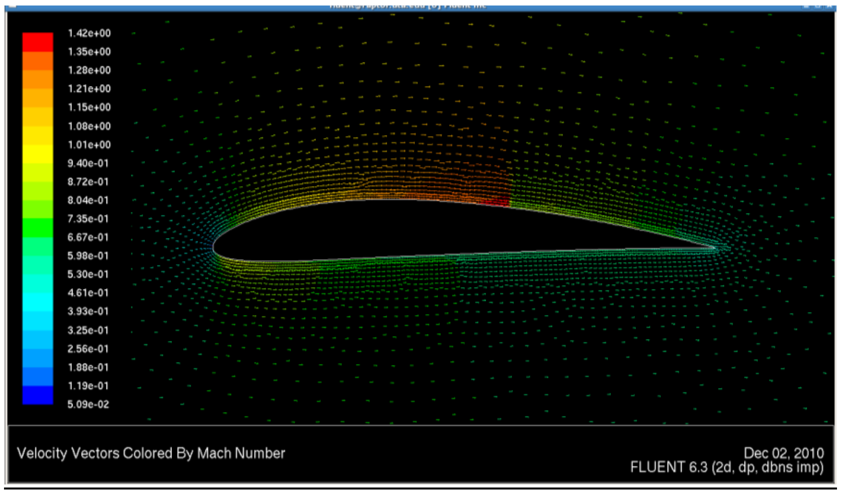


Figure 4.41 The velocity flow field of NACA4412 airfoil at far field M=0.7

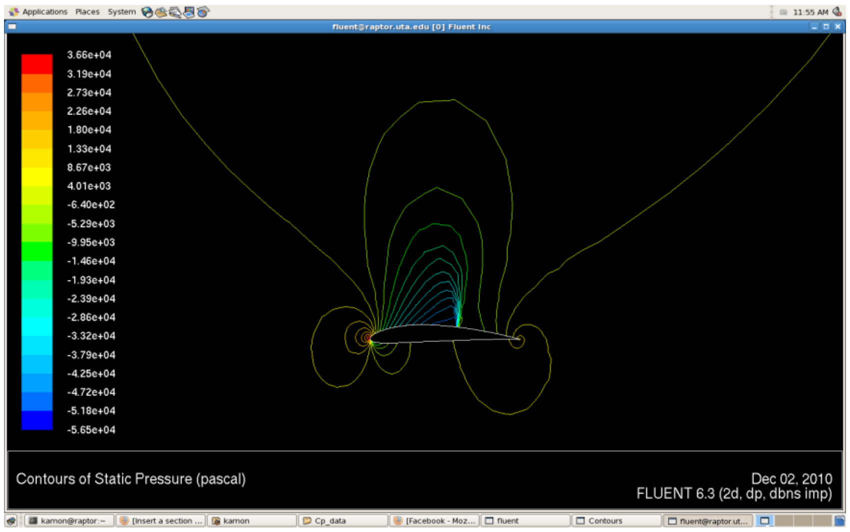


Figure 4.42 The pressure contour of NACA4412 airfoil at far field M=0.7

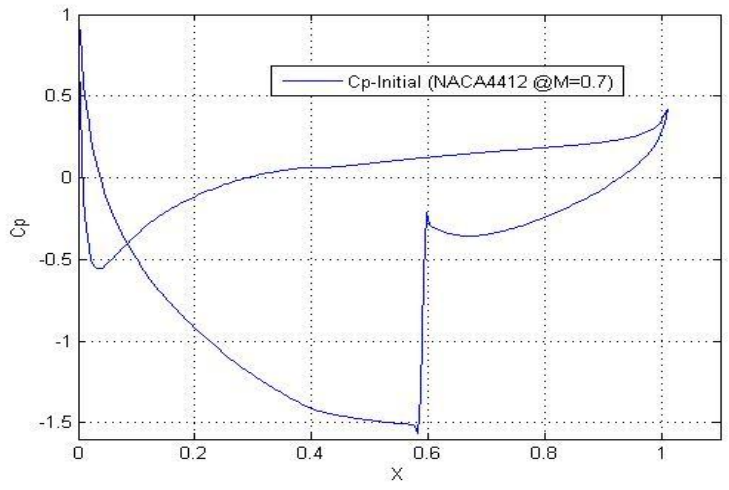


Figure 4.43 The Cp distribution of NACA4412 airfoil M=0.7

Local shock wave is the main cause of wave drag in transonic flow; therefore, we would like to design a new airfoil which does not produce shock wave to the flow field. According to the Cp distribution of NACA4412 airfoil at Mach number 0.7, we would like to remove the discontinuity of the Cp distribution by using continuous Cp distribution of NACA4412 airfoil at Mach number 0.4 as a target Cp distribution. Figure 4.44 shows the Cp distribution of NACA4412 at Mach 0.4 compared to at Mach 0.7.

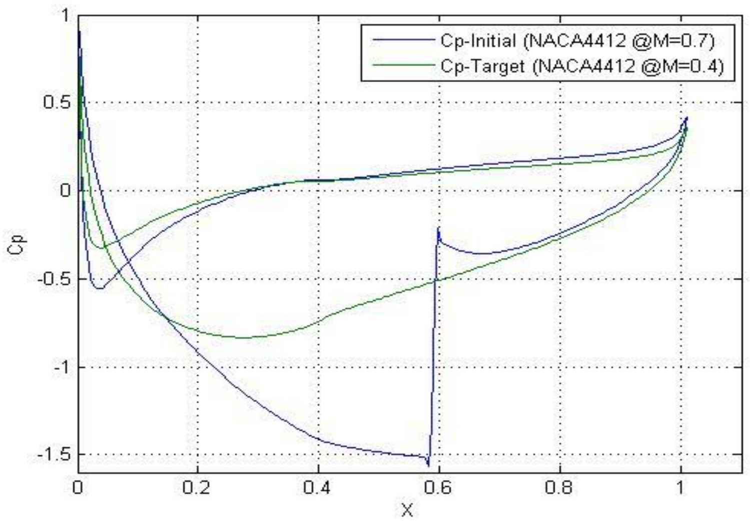


Figure 4.44 The Cp distribution of NACA4412 airfoil at M=0.4 and M=0.7

Set the flow solver, FLUENT, to be pressure base solver, ideal gas, inviscid flow, Mach=0.7 at the far field boundary, Second order upwind discretization, Courant Number = 5 at the starting, 10 after 500 iterations, and 2 after 1000 iterations, absolute criteria for residuals are 10e-5. The coefficients for the MGM equations are $\beta_0 = 20, \beta_1 = 0.001, \beta_2 = 4$. After 29 iterations, the the error at almost all points is less than 5 percent except at the leading edge zone of lower airfoil. It is because of the smooth and continuous boundary conditions at the leading edge that cause coordinates at the front part of the lower airfoil insignificantly change to get the Cp target. However, the shock wave on the upper airfoil is removed which is the main purpose in this case. Figure 4.45 shows the errors of all points at the 29th iteration. Figure 4.46 shows the shape of the designed airfoil compared with the initial airfoil and Figure 4.47 shows the Cp distributions of the designed airfoil compared with the target Cp distribution. Figure 4.48 shows the norm error which indicates that the error decreases when the number of iteration increase.

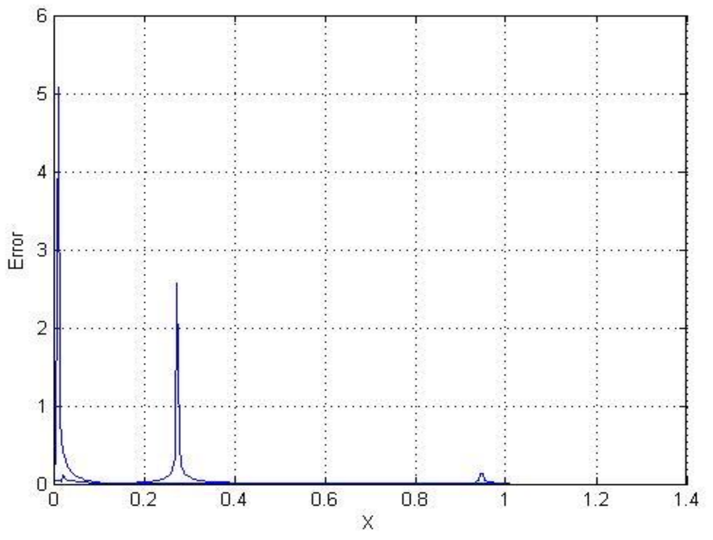


Figure 4.45 The errors of all points at the 29th iteration using Cp of NACA4412 at M=0.7 as an initial and at M=0.4 as an target for the shock removal case

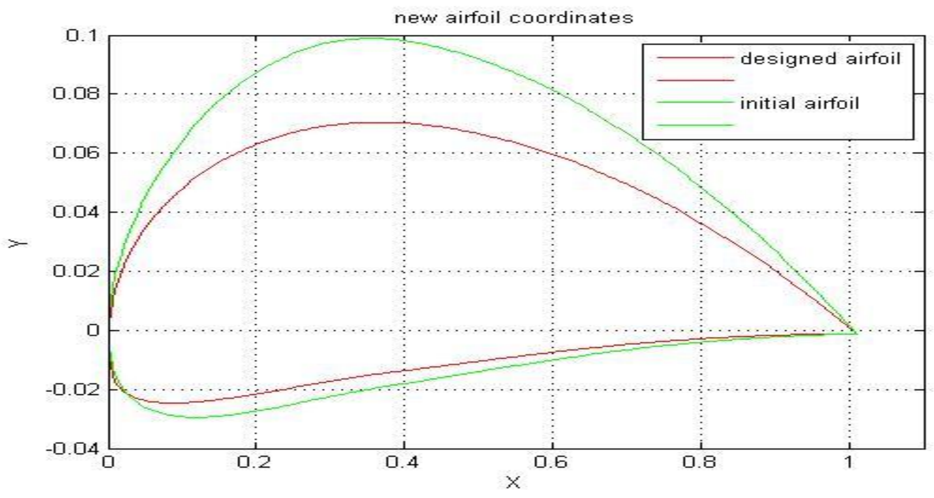


Figure 4.46 The shape of the designed airfoil compared with the initial airfoil using Cp of NACA4412 at M=0.7 and at M=0.4 as an target for the shock removal case

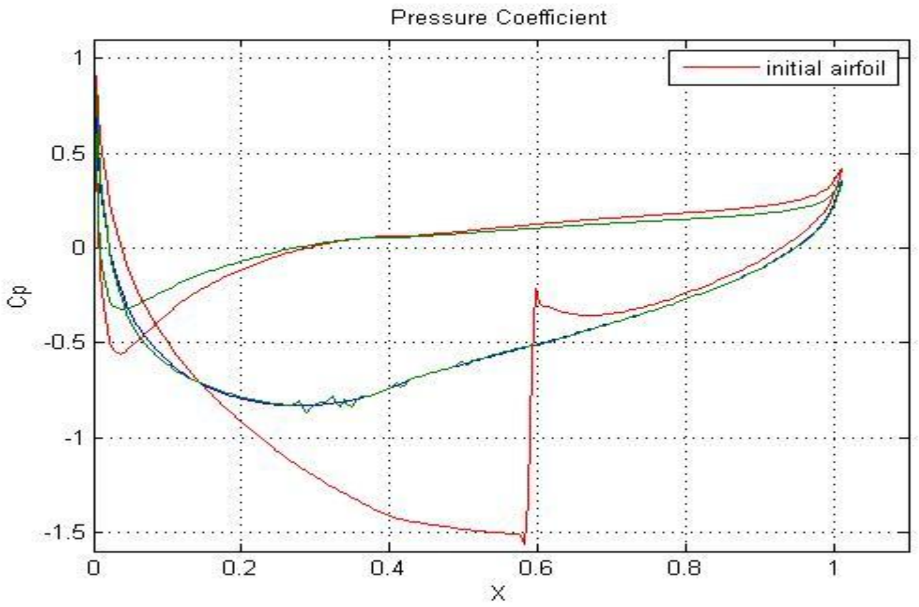


Figure 4.47 The Cp distributions of the designed airfoil(green line) compared with the target Cp distribution of NACA4412 at M=0.4 (blue line) for the shock removal case

For illustration, Figure 4.49 shows the comparison of the velocity flow field of the designed airfoil, the target airfoil, and the initial airfoil. Figure 4.50 shows the comparison of pressure contour of the designed airfoil, the target airfoil, and the initial airfoil.

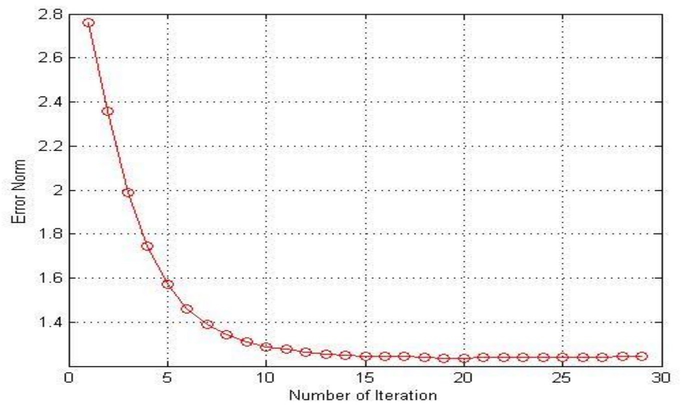
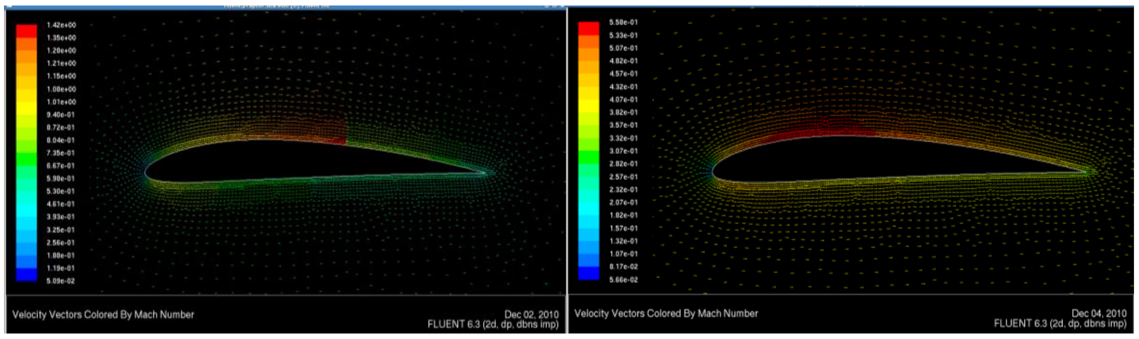
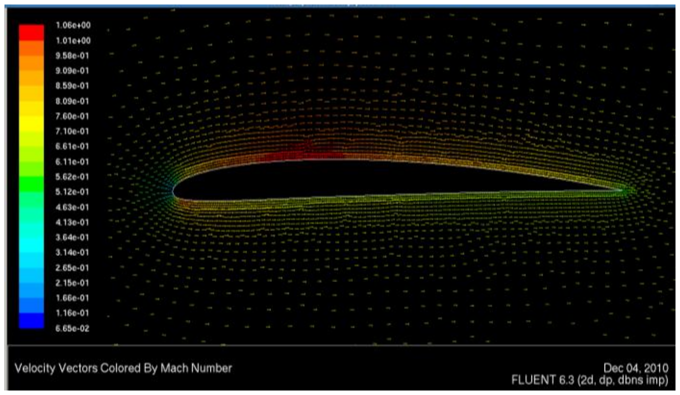


Figure 4.48 The norm errors using Cp of NACA4412 at M=0.7 as an initial and at M=0.4 as an target for the shock removal case



Velocity Vectors Colored By Mach Number
Dec 02, 2010
FLUENT 6.3 (2d, dp, dbns imp)
Velocity Vectors Colored By Mach Number
Dec 04, 2010
FLUENT 6.3 (2d, dp, dbns imp)

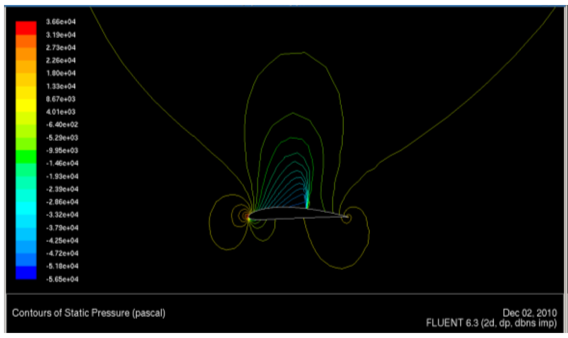
(a) (b)



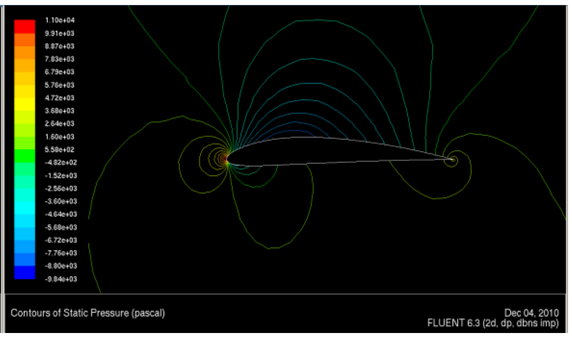
Velocity Vectors Colored By Mach Number
Dec 04, 2010
FLUENT 6.3 (2d, dp, dbns imp)

(c)

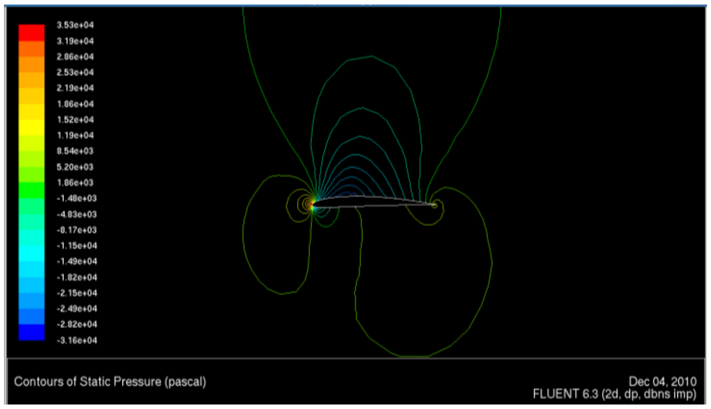
Figure 4.49 (a) Velocity flow field of the initial airfoil NACA4412 for inviscid @M=0.7
(b) Velocity flow field of the target airfoil NACA4412 for inviscid @M=0.4
(c) Velocity flow field of the designed airfoil for inviscid @M=0.7



(a)



(b)



(c)

Figure 4.50 (a) Pressure contour of the initial airfoil NACA4412 for inviscid @M=0.7
(b) Pressure contour of the target airfoil NACA4412 for inviscid @M=0.4
(c) Pressure contour of the designed airfoil for inviscid @M=0.7

However, in some case, we cannot completely remove shock waves from the initial airfoil because there is no airfoil geometry which does not produce shock wave for the desired Cp distribution in some conditions. For example, when we choose the Cp distribution of NACA4412 at M=0.6 which is the continuous Cp distribution as the target, we cannot get a new airfoil geometry which produce a Cp distribution as the target in M=0.7 condition. Figure 4.51 shows the designed Cp distribution compared with the target. The designed Cp at almost all points already approached the target accept at one zone on the upper airfoil called a high residual zone. Although the factor is multiplied to the force vector at that zone to produce a bigger change of y-coordinates, the Cp does not get closer to the target. The error norm does not decrease as well. It is because at the front part of the high zone, Cp distribution already reached the target Cp distribution; therefore, the shape of the front part does not change although it still produce local a Mach number which is greater than one. For that reason, the new airfoil always produces a shock wave when there is still a supersonic flow on the upper airfoil. Figure 4.52 shows the error norm which does not go down anymore after 20 iterations. It means that the designed airfoil at that point is the best shape corresponding to the target Cp distribution. Although shock wave is not removed completely, the new airfoil can significantly weaken the shock from the initial airfoil. Figure 4.53 shows the velocity contour of the designed airfoil which still have a small shock wave on the upper airfoil when using NACA4412 @M=0.6 as a target. Figure 4.54 shows the velocity contour of the designed airfoil which still have a small shock wave on the upper airfoil when using NACA4412 @M=0.5 as a target. It is obvious that, when using NACA4412 @M=0.5 as a target, the designed airfoil has a smaller shock wave than when using NACA4412 @M=0.6 as a target.

From the previous case, it is obvious that the target Cp distribution importantly affects the design result of the shock removal cases. Target Cp distribution should be chosen in the way that there is enough Cp-difference between the initial airfoil and target airfoil which can make enough change to y-coordinate to slow down the local mach number to be lower than one before it reaches the target Cp distribution. Figure 4.55 shows that when using Cp distribution of NACA4412 at M=0.5, the shock wave is smaller than when we use the Cp distribution of NACA4412 at M=0.6. Figure 4.56 shows the error norm of this case which does not go down anymore after 20 iterations although there is a high residual zone of the difference between the design Cp distribution and the target Cp distribution. Shock wave still cannot be completely removed until we use the target Cp distribution of NACA4412 at M=0.4.

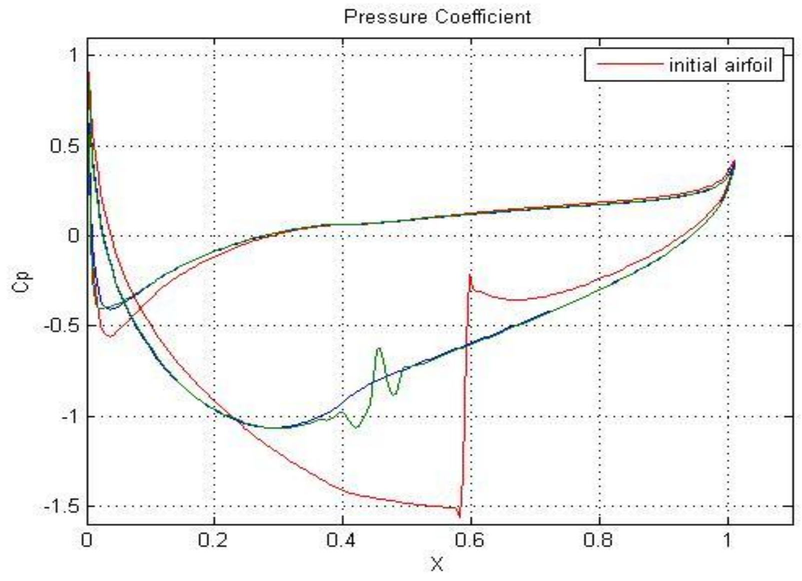


Figure 4.51 The Cp distributions of the designed airfoil (green line) compared with the target Cp distribution of NACA4412 at M=0.6 (blue line) for the shock removal case

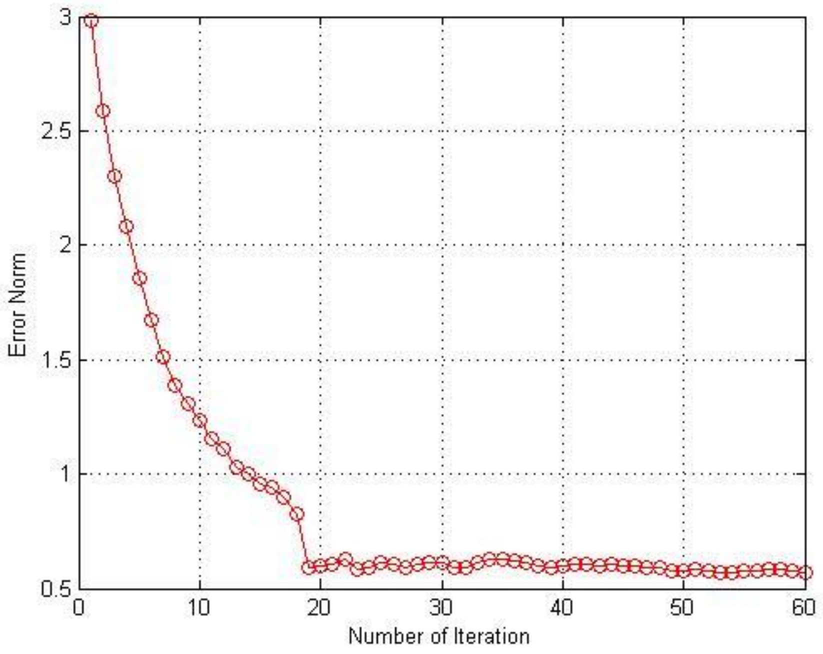


Figure 4.52 The errors norm using Cp of NACA4412 at M=0.7 as an initial and at M=0.6 as an target for the shock removal case

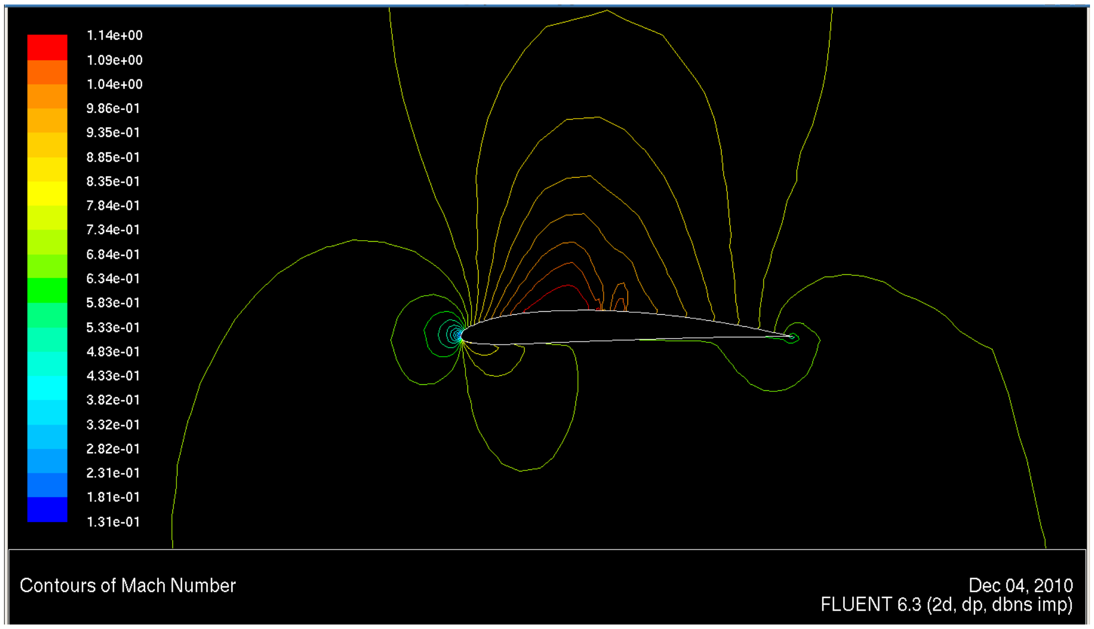


Figure 4.53 shows the velocity contour of the designed airfoil which still has a small shock wave on the upper airfoil when using NACA4412 @M=0.6 as a target

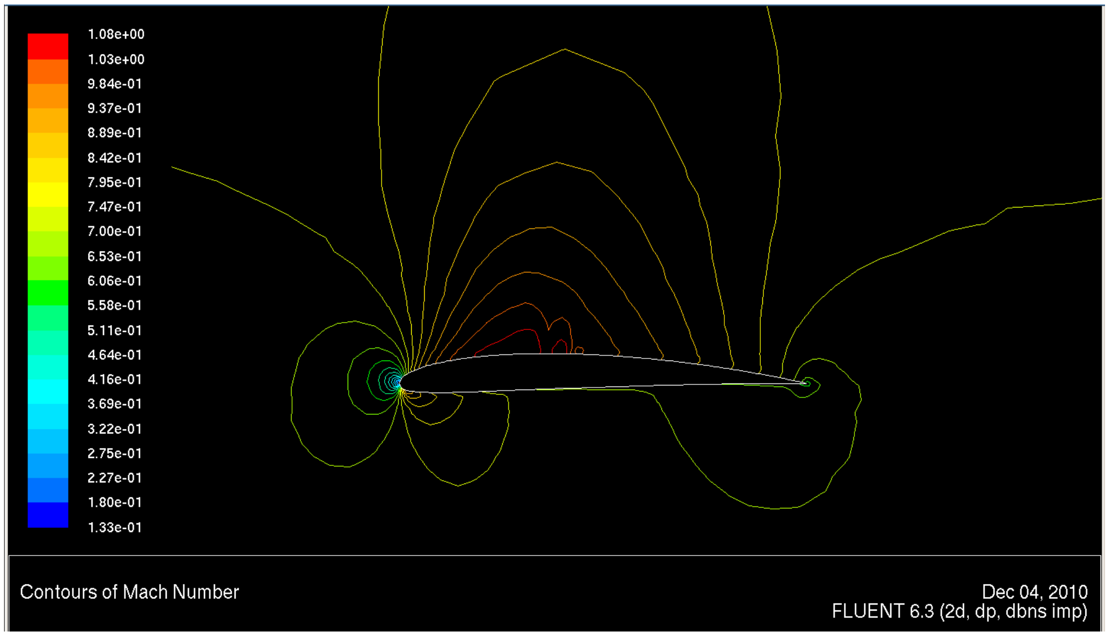


Figure 4.54 shows the velocity contour of the designed airfoil which still has a small shock wave on the upper airfoil when using NACA4412 @M=0.5 as a target

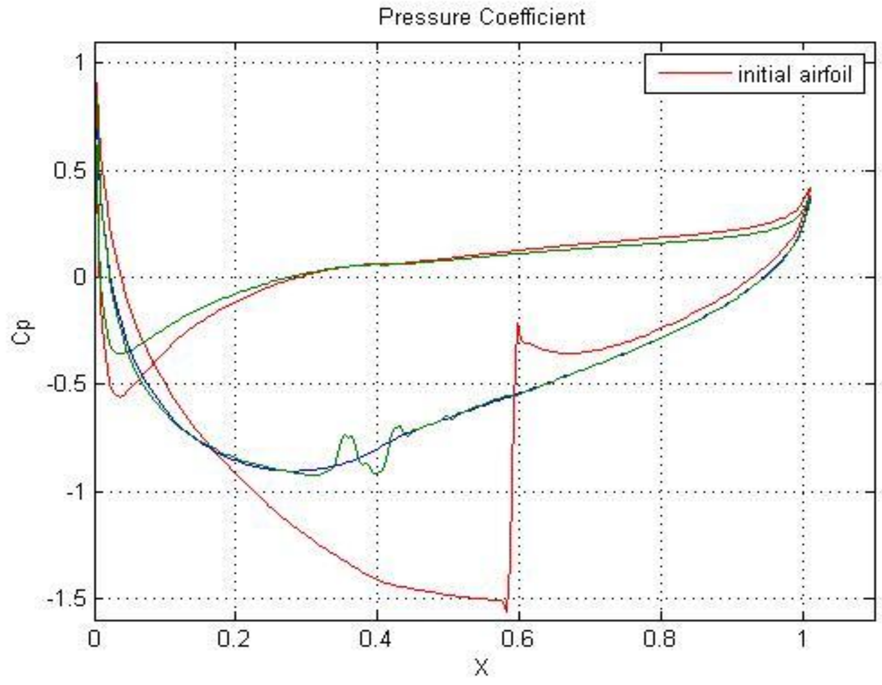


Figure 4.55 The Cp distributions of the designed airfoil(green line) compared with the target Cp distribution of NACA4412 at M=0.5 (blue line) for the shock removal case

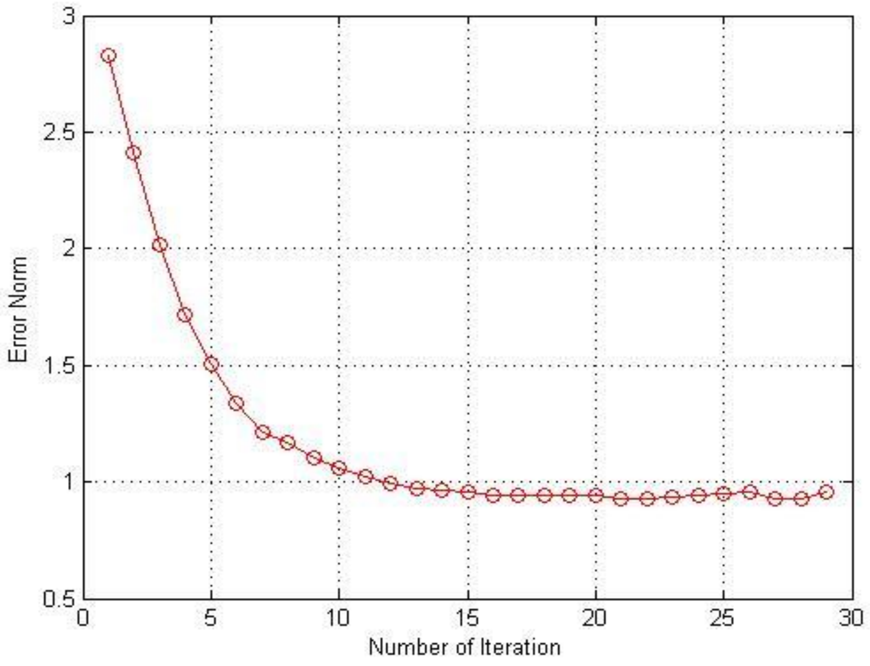


Figure 4.56 The errors norm using Cp of NACA4412 at M=0.7 as an initial and at M=0.5 as an target for the shock removal case

CHAPTER 5

CONCLUSION

The Finite Difference Method for Non-Uniform Grid was developed to discretize the Flexible Membrane Equation. The non-uniform-coordinate airfoils were meshed by using moving grid program so that the leading edge was finely mesh and the other zone were meshed with bigger grids; therefore, it reduces the computational time when analyze the flow with commercial software, FLUENT. FLUENT was set to a segregated solver when solving the incompressible flow, M<0.3, and set to coupled solver when solving the compressible flow, M>0.3. NACA0011 and NACA0012 are use as initials airfoils and NACA2315 and NACA4412 are use as a target airfoil to validate the program at both subsonic and transonic regimes.

In subsonic condition at M=0.2 which is considered to be the incompressible flow, the program worked well and can get the target airfoil when 45 iterations were used. The rates of convergence are pretty much the same for inviscid case and viscous case. The compressible viscous subsonic at M=0.6 case was tested and the program also worked well. Twenty one iterations were used to get the target airfoil. The program also gave a good result in transonic regime when the discontinuities occur. However, when there is discontinuity, the program has slower convergent rate. MGM parameters were chosen appropriately in order to get fast convergence when discontinuity occurs.

The program is successfully used to design a shock free airfoil when the initial airfoil has shock waves. In order to completely remove shock waves from the initial airfoil, appropriate target Cp distribution should be chosen otherwise there will be a small shock wave still on the designed airfoil. From section 4.3, using Cp distribution from lower Mach number is easier to completely remove shock waves when using the same number of iterations. When using the Cp distribution at higher Mach number as a target, shock wave cannot be removed completely. For example, we could not completely remove shock wave from the initial airfoil NACA4412 @M=0.7 when using the Cp distribution of NACA4412 @M=0.6 and @M=0.5 as targets. On the other hand, when using the Cp distribution of NACA4412 @M=0.4 as a target, shock wave could be removed completely. Although, in some cases, shock waves were not removed completely, the program could significantly weaken the shocks to be much smaller than the initial airfoil.

APPENDIX A

THE PROGRAM FOR GENERATING NACA 4-DIGIT AIRFOIL (MATLAB CODE)

```
%% This program is for generating NACA 4-digit airfoil
clc
clf
clear all
format 'long'

%% input the 4 digits
%m   is the digit which specifies the maximum camber in percentage of
the
      %chord (airfoil length)
%p   is the digit which indicates the position of the maximum camber
in
      %tenths of cord
%t   is the two digit which provide the maximum thickness of the
airfoil in
      %percentage of cord

m = 2;
p = 3;
t = 15;



%%
c=1; %chord length
p=p/10;
m=m/100;
t=t/100;

dx=0.02;
x=0:dx:1;
o=(1/dx)+1;
if p==0
    for i=1:o
    yc(i)=(m/((1-p)^2))*((1-2*p)+2*p*x(i)-x(i)^2);
    dyc(i)=(m/(1-p)^2)*2*(p-x(i));
    end
else
    for i=1:((p/dx)+1)
    yc(i)= (m/p^2)*(2*p*x(i)-x(i)^2);
    dyc(i)=(m/p^2)*2*(p-x(i));
    end
    for i=((p/dx)+1):o
    yc(i)=(m/((1-p)^2))*((1-2*p)+2*p*x(i)-x(i)^2);
    dyc(i)=(m/(1-p)^2)*2*(p-x(i));
    end
end

for i=1:o
    yt(i)=(t/0.2)*(0.2969*x(i)^0.5-0.1260*x(i)-
0.3516*x(i)^2+0.2843*x(i)^3-0.1015*x(i)^4);
end
```

```
for i=1:o
    teta(i)=atan(dyc(i));
    XU(i)=x(i)-yt(i)*sin(teta(i));
    YU(i)=yc(i)+yt(i)*cos(teta(i));
    XL(i)=x(i)+yt(i)*sin(teta(i));
    YL(i)=yc(i)-yt(i)*cos(teta(i));
end
YU(1)=0; YU(51)=0; YL(1)=0; YL(51)=0; XU(51)=1; XL(51)=1;


XU=XU';
YU=YU';
XL=XL';
YL=YL';

for i=1:51
    j=52-i;
    XLN(j)=XL(i);
    YLN(j)=YL(i);

end

XL=XLN';
YL=YLN';

%% we are doing an output matrix for GAMBIT form
display ('coordinate[X Y Z] for save as ".txt" and then input to
GAMBIT for generating the mesh')
coor=[XU YU;XL(2:51) YL(2:51)];
s=zeros(1,101)';
coor=[coor s];
coor=coor(1:100,:);
g=[0.5 0 0; 1.5 0 0; -0.5 0 0; 0.5 0.3 0; 0.5 -0.3 0; -10 10 0; -10 -
10 0; 20 -10 0; 20 10 0; -10 0 0; 20 0 0];
coor=[coor ; g]
figure(1)
plot(XU,YU,'*-r',XL,YL,'*-b')
grid on
```

APPENDIX B

THE PROGRAM FOR SOVING MGM EQUATION
(MATLAB CODE)

```
%% This Program is for solving the MGM equation by using Non-Uniform
%% grid Finite Difference Method

clear all
clf %clear figure
clc
format 'long'

%% If this is the first iteration please put I=1 else put I=2
I=2;
%% Catalye for high rate of convergence(one time for more than 10th
iteration)
cat=1;  %%% 4(1,10),7(1,20),8(5,1),9(5,1),13(10,1),18(1,20),19(1,20)
cat2=1;
%%
ne=206;

if I==1
   display('This is the first iteration')
else


%% New airfoil data input
format 'long'
fid = fopen('21(Cp_formatlabread).txt', 'r');
data_bot = textscan (fid, '%f %f %f ',206, 'headerlines',0);
fid = fopen('21(Cp_formatlabread).txt', 'r');
data_top = textscan (fid, '%f %f %f ',212, 'headerlines',205);

Cp_top = [data_top{3}];
Cp_bot = [Cp_top(207); data_bot{3}];
X_top=data_top{1};
Y_top=data_top{2};
X_bot=[X_top(207); data_bot{1}];
Y_bot=[Y_top(207); data_bot{2}];


end


%% NACA2315 data

fid = fopen('NACA4412(Cp_formatlabread).txt', 'r');
data1 = textscan (fid, '%f %f %f ',206, 'headerlines',0);
fid = fopen('NACA4412(Cp_formatlabread).txt', 'r');
data2 = textscan (fid, '%f %f %f ',212, 'headerlines',205);
format 'long'
Cp_top2315 = [data2{3}];
Cp_bot2315 = [Cp_top2315(207); data1{3}];
X_top2315=data2{1};
Y_top2315=data2{2};
```

```
X_bot2315=[X_top2315(207); data1{1}];
Y_bot2315=[Y_top2315(207); data1{2}];


% figure(1)
% clf %clear figure
% plot(X_top2315,Y_top2315,'o-r','MarkerSize',7)
% hold on
% plot(X_bot2315,Y_bot2315,'o-b','MarkerSize',7)
% xlabel ('X')
% ylabel ('Z')
% legend ('upper airfoil', 'lower airfoil')
% title('NACA2315')
% grid on
%
% figure(2)
% plot(X_top2315,Cp_top2315,'.-r','Markersize',12)
% hold on
% plot(X_bot2315,Cp_bot2315,'.-b','Markersize',12)
% xlabel ('X')
% ylabel ('Cp')
% legend ('upper airfoil', 'lower airfoil')
% title('Pressure Coefficient')
% grid on

%% NACA0011 data
fid = fopen('NACA0012(Cp_formatlabread).txt', 'r');
data3 = textscan (fid, '%f %f %f ',206, 'headerlines',0);
fid = fopen('NACA0012(Cp_formatlabread).txt', 'r');
data4 = textscan (fid, '%f %f %f ',212, 'headerlines',205);

Cp_top0011 = [data4{3}];
Cp_bot0011 = [Cp_top0011(207); data3{3}];
X_top0011=data4{1};
Y_top0011=data4{2};
X_bot0011=[X_top0011(207); data3{1}];
Y_bot0011=[Y_top0011(207); data3{2}];

% figure()
% plot(X_top0011,Y_top0011,'o-r','MarkerSize',7)
% hold on
% plot(X_bot0011,Y_bot0011,'o-b','MarkerSize',7)
% xlabel ('X')
% ylabel ('Z')
% legend ('upper airfoil', 'lower airfoil')
% title('NACA0011')
% grid on
% hold off
%
% figure()
% plot(X_top0011,Cp_top0011,'.-r','Markersize',12)
% hold on
% plot(X_bot0011,Cp_bot0011,'.-b','Markersize',12)
% xlabel ('X')
% ylabel ('Cp')
% legend ('upper airfoil', 'lower airfoil')
% title('Pressure Coefficient')
% grid on

%% START NUMERICAL CALCULATION FOR DELTA_Y

% Choose the value of constant beta0, beta1, and beta2
if I==1
beta0 = 25;
beta1 = 0.1;
beta2 = 4;
else
beta0 = 30;
beta1 = 0.1;
beta2 = 4;
end
%



%% new constants
%% for top airfoil 207 points total
x1=X_top0011;

% Calculating dx. (from element 1 to element 206) x(1) to x(207) will
be used.
for i=1:206

    dx1(i)=(x1(i+1)-x1(i));

end

% Calculating the coefficients A, B, and C (from A(2) to A(206) which
are
% 205 data)

for i=2:206

    C11=(dx1(i-1)*dx1(i)^2+dx1(i)*dx1(i-1)^2)/2;
    C21=dx1(i-1)^2*dx1(i)+dx1(i)^2*dx1(i-1);

    A1(i)=beta2*(dx1(i)/C11)-beta1*(dx1(i)^2/C21);
% A(2) to A(206) exclude at the leading edge and trailing edge
    B1(i)=beta0-beta2*((dx1(i-1)+dx1(i))/C11)-beta1*((dx1(i-1)^2-
dx1(i)^2)/C21); % B(2) to B(206) exclude at the leading edge and
trailing edge
    C1(i)=beta1*(dx1(i-1)^2/C21)+beta2*(dx1(i-1)/C11);
%C(2) to C(206) exclude at the leading edge and trailing edge
```

```
end


% Construct Matrix [M] from  [M]{dy}= {C}, where {C}={Cp(target)-
Cp(actual)}

M1(1,1)=B1(2);
M1(1,2)=C1(2);
for i=2:204
    j=i;
    M1(i,j-1)=A1(i);
    M1(i,j)=B1(i);
    M1(i,j+1)=C1(i);

end
M1(205,204)=A1(206)+C1(206)*(1-(x1(207)-x1(205))/(x1(206)-x1(205)));
M1(205,205)=B1(206)+C1(206)*((x1(207)-x1(205))/(x1(206)-x1(205)));


% Construct Matrix {C}
if I==1

    for i=1:205
    d1(i)=Cp_top2315(i+1)-Cp_top0011(i+1);
    end

else

    for i=1:205
    d1(i)=Cp_top2315(i+1)-Cp_top(i+1);
    end

end

d1=cat*d1;  %%%%%%%%%% catalyzing


% Use Thomas Algorithm to solve the system of equations

% Compute the new coefficients by forward sweep the matrix in order to
% obtain a matrix with two diagonals in removing the coefficients "a"
% A_new(1,1)=A(1,1);

M1_new(1,2)=M1(1,2)/M1(1,1);
d1_new(1)=d1(1)/M1(1,1);

for i=2:204
    M1_new(i,i+1)=M1(i,i+1)/(M1(i,i)-M1_new(i-1,i)*M1(i,i-1));
% new coef "c"
end
```

```
for i=2:205
    d1_new(i)    =(d1(i)-d1_new(i-1)*M1(i,i-1))/(M1(i,i)-M1_new(i-
1,i)*M1(i,i-1));  % new {C}
end

% Backward sweep the matrix in order to calculate the solution
% we obtain initially
dY1(205)=d1_new(205);

% Then, we calculate the solution for all other position 'x'
for i=204:-1:1
    dY1(i)=d1_new(i)-M1_new(i,i+1)*dY1(i+1);
end

dY1=[0 dY1];

% extrapolate
dY1(207)=(1-((x1(207)-x1(205))/(x1(206)-x1(205))))*dY1(205)+((x1(207)-
x1(205))/(x1(206)-x1(205)))*dY1(206);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% for bottom airfoil 207 points
total%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x2=X_bot0011;

% Calculating dx. (from element 1 to element 206) x(1) to x(207) will
be used.

for i=1:206

    dx2(i)=x2(i+1)-x2(i);

end

% Calculating the coefficients A, B, and C (from A(2) to A(206) which
are
% 205 data)

for i=2:206

    C12=(dx2(i-1)*dx2(i)^2+dx2(i)*dx2(i-1)^2)/2;
    C22=dx2(i-1)^2*dx2(i)+dx2(i)^2*dx2(i-1);

    A2(i)=beta2*(dx2(i)/C12)-beta1*(dx2(i)^2/C22);
% A(2) to A(206) exclude at the leading edge and trailing edge
```

```
    B2(i)=beta0-beta2*((dx2(i-1)+dx2(i))/C12)-beta1*((dx2(i-1)^2-
dx2(i)^2)/C22); % B(2) to B(206) exclude at the leading edge and
trailing edge
    C2(i)=beta1*(dx2(i-1)^2/C22)+beta2*(dx2(i-1)/C12);
%C(2) to C(206) exclude at the leading edge and trailing edge

end


% Construct Matrix [M] from  [M]{dy}= {C}, where {C}={Cp(target)-
Cp(actual)}

M2(1,1)=B2(2);
M2(1,2)=C2(2);
for i=2:204
    j=i;
    M2(i,j-1)=A2(i);
    M2(i,j)=B2(i);
    M2(i,j+1)=C2(i);

end
M2(205,204)=A2(206);
M2(205,205)=B2(206);


% Construct Matrix {C}
if I==1

    for i=1:205
    d2(i)=Cp_bot2315(i+1)-Cp_bot0011(i+1);
    end
    d2(1)=d2(1)+dY1(207)*A2(2);

else
    for i=1:205
    d2(i)=Cp_bot2315(i+1)-Cp_bot(i+1);
    end
    d2(1)=d2(1)+dY1(207)*A2(2);

end

d2=-
1*cat2*d2;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Use Thomas Algorithm to solve the system of equations

% Compute the new coefficients by forward sweep the matrix in order to
% obtain a matrix with two diagonals in removing the coefficients "a"
% A_new(1,1)=A(1,1);

M2_new(1,2)=M2(1,2)/M2(1,1);
```

```
d2_new(1)=d2(1)/M2(1,1);

for i=2:204
    M2_new(i,i+1)=M2(i,i+1)/(M2(i,i)-M2_new(i-1,i)*M2(i,i-1));
% new coef "c"
end

for i=2:205
    d2_new(i)    =(d2(i)-d2_new(i-1)*M2(i,i-1))/(M2(i,i)-M2_new(i-
1,i)*M2(i,i-1));  % new {C}
end

% Backward sweep the matrix in order to calculate the solution
% we obtain initially
dY2(205)=d2_new(205);

% Then, we calculate the solution for all other position 'x'
for i=204:-1:1
    dY2(i)=d2_new(i)-M2_new(i,i+1)*dY2(i+1);
end

dY2=[dY1(207) dY2 0];

 %% Then we get a new shape of the airfoil from the first iteration
if I==1

    for i=1:207
    Y_top_new(i)=Y_top0011(i)+dY1(i);

    Y_bot_new(i)=Y_bot0011(i)+dY2(i);
    end

else

    for i=1:207
    Y_top_new(i)=Y_top(i)+dY1(i);

    Y_bot_new(i)=Y_bot(i)+dY2(i);
    end

end
new_coor=[x1 Y_top_new'; x2 Y_bot_new'];

figure(9)
plot(x1,Y_top_new,'-r',x2,Y_bot_new,'-r','Markersize',7)
grid on
hold on
plot(x1,Y_top0011,'-g',x2,Y_bot0011,'-g','Markersize',7)
hold on
plot(x1,Y_top2315,'-b',x2,Y_bot2315,'-b','Markersize',7)
xlabel ('X')
```

```
ylabel ('Z')
legend ('new airfoil', '-','initial airfoil', '-','target airfoil')
title('new airfoil coordinates')
grid on

hold off

%% Creat an output text file of new airfoil coordinate
format 'long'
fid = fopen('naca0011.dat', 'r');
data11 = textscan (fid, '%f %f %f ',412, 'headerlines',1);
k3=[data11{1}];
k1=new_coor(1:207,2:2);
for i=1:207
    k11(i)=k1(208-i);
end
k2=new_coor(209:413,2:2);
for i=1:205
    k21(i)=k2(206-i);
end
kk=[k11'; k21'];

newcoor=[k3 kk]

%% Calculate Errer
xx1=[X_top2315; X_bot2315];
ww1=[Cp_top2315; Cp_bot2315];
ww2=[Cp_top; Cp_bot];

ggg=[ww1]-[ww2];

for i=1:414
    rr(i)=abs(ggg(i))/abs(ww1(i));
end

rrr=rr';

fff=[xx1 rrr];

figure(10)
plot(xx1,rrr)
xlabel ('X')
ylabel ('%ERROR')
axis([0 1.1 0 2])
grid on

figure(11)
plot(xx1,ww1,xx1,ww2)
axis([0 1.1 -1.5 1])
xlabel ('X')
ylabel ('Cp')
```

```
legend ('Cp-target (NACA4412)', 'Cp-the designed airfoil')%%,'1st
iteration', '2nd iteration','3rd iteration','4th iteration','5th
iteration')
%title('new airfoil coordinates')
grid on
```

REFERENCES

[1] Jahangirian, A. and Shahrokhi, A.(2009) 'Inverse design of transonic airfoils using genetic algorithm and a new parametric shape method', Inverse Problems in Science and Engineering, 17: 5, pp. 681-699

[2] Dulikravich, George S. and Baker, Daniel P. (1999) 'Aerodynamic Shape Inverse Design Using a Fourier Series Method', American Institute of Aeronautics & Astronautics, AIAA99-0185, pp. 1-15

[3] Fernandez, Anibal F. and Kulas, Lukasz (N.A.) 'A Simple Finite Difference Approach Using Unstructured Meshes from FEM Mesh Generators', IEEE Xplore, pp. 585-588

[4] Volpe, Ernani V., Oliveira, Guilherme L., Santos, Luis C. C., Hayashi, Marcelo T. and Ceze, Marco A. B.(2009) 'Inverse aerodynamic design applications using the MGM hybrid formulation', Inverse problems in Science and engineering, 17:2, pp. 245-261

[5] Nili-Ahmadabadi, Mahdi, Durali, Mohammad, Hajilouy-Benisi, Ali and Ghadak, Farhad (2009) 'Inverse design of 2-D subsonic ducts using flexible string algorithm', Inverse Problems in Science and Engineering, 17:8, pp. 1037-1057

[6] Asharafizadeh, A., Raithby, G.D., and Stubley, G.D. (2003) 'Direct Design of Ducts', J. Fluids Eng. Trans. ASME, 125, pp. 158-165

[7] Garabedian, P., and McFadden, G., 'Design of Supercritical Swept Wings' AIAA Journal, Vol. 20, No. 3, March 1982, pp. 289-291

[8] Garabedian, P., and McFadden, G., (1982) 'Computational Fluid Dynamics of Airfoils and Wings in Transonic, Shock, and Multi-dimensional Flows', Advances in Scientific Computing, Academic Press, pp. 1-16

[9] Malone, J. B., Vadyak, J., and Sankar, L. N., ' A Technique for the Inverse Aerodynamic Design of Nacelles and Wing Configurations', AIAA Journal of Aircraft, Vol. 24, No. 1, January 1987, pp. 8-9

[10] Banker, D. P., 'A Fourier Series Approach to the Elastic Membrane Inverse Shape Design Problem in Aerodynamics', M.Sc. Thesis, Dept. of Aerospace Eng., The Pennsylvania State University, May 1999.

[11] Barrett, Thomas, R., Bressloff, Neil, W., and Keane, Andy J., 'Airfoil Shape Design and Optimization Using Multifidelity Analysis and Embedded Inverse Design', AIAA Journal, Vol. 44, No. 9, September 2006.

[12] Henriques, J.C.C., Marques da Silva, F., Estanqueiro, A.I., and Gato, L.M.C., 'Design of a new urban wind turbine airfoil using a pressure-load inverse method', Renewable Energy, Vol. 34,June 2009, pp. 2728-2734

[13] Wu, Zhe-Min, Chen, Chi, and Liu, Gao-Lian., ‘Multipoint inverse shape design of airfoils based on variable-domain variational principle’, Aircraft Engineering and Aerospace Technology, Vol. 76, No. 5, 2004, pp. 516-522

[14] Santos, Luis Carlos de Castro.(1999, June). ‘A fixed point algorithm for the inverse solution of fluid flow equations’ 3rd International Conference on Inverse Problems in Engineering, Port Ludlow, Washington

[15] Dennis, Brian, H.(2004, February) ‘Deforming Unstructured Meshes for Computations with a Moving Boundary’

[16] Laskowski, G. M., Vicharelli, A., Medic, G., Elkins, C. J., Eaton, J. K., and Durbin, P. A. ‘Inverse design of an experimental measurements in a double-Passage Transonic Turbine Cascade Model’ Journal of Turbomachinery, Vol.127, July 2005, pp. 619-626

[17] Bonaiuti, Duccio, Zangeneh, Mehrdad, Aartojarvi, Reima, and Eriksson, Jonas. “Parametric design of a waterjet pump by means of inverse design, CFD calculations and experimental analyses” Journal of Fluids Engineering, Vol. 132, march 2010, pp. 031104-(1-15)

[18] Dulikravich, G. S., “Design and Optimization Tools Development”, New Design Concepts for High Speed Air Transport, Springer, New York, 1997, pp. 159-236

[19] Fujii, K., and Dulikravich, G. S. “Recent Development of Aerodynamic Design Methodologies – Inverse Design and Optimization” Vieweg Series on Notes on Numerical Fluid Mechanics, Springer, 1999.

[20] © Fluent Inc. “Getting Started” FLUENT MANUAL, November 28, 2001

BIOGRAPHICAL INFORMATION

Kamon Thinsurat was born in Phattalung, Thailand, in September 1984. He moved to Krabi, Thailand, after that until he finished grade 12 at Ammartpanitchanukul School. He received his Bachelor Degree in Aerospace Engineering from King Mongkut's University of Technology North Bangkok, Bangkok, Thailand, in March 2007. In 2008, he continued his Master Degree in Aerospace Engineering at The University of Texas at Arlington, Texas, USA.