1-1-2012

Shape Optimization of Hydrofoils

William Cocke

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Mechanical Engineering and Material Science

Thesis Examination Committee:
Ramesh Agarwal (Chair)
David Peters
Harold Brandon

SHAPE OPTIMIZATION OF HYDROFOILS NEAR A FREE SURFACE USING A GENETIC ALGORITHM

by

William Travis Cocke

A thesis presented to the School of Engineering and Applied Science
of Washington University in St. Louis in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

May 2012
Saint Louis, Missouri

ABSTRACT

Shape Optimization of Hydrofoils Near a Free Surface Using a Genetic Algorithm

by

William Travis Cocke

Master of Science in Mechanical Engineering

Washington University in St. Louis, 2012

Research Advisor: Professor Ramesh K. Agarwal

In this thesis, a genetic algorithm (GA) is employed for shape optimization of hydrofoils for application in a sailing craft. The hydrofoil for a sailing craft should have high lift at lower speeds and low drag at higher speeds. Computations are performed for a hydrofoil in deep water as well as one close to the free surface. The commercially available software FLUENT is used for calculation of the flow field, and the software GAMBIT is used for the geometry and mesh generation. Volume of Fluid (VOF) method in FLUENT is employed to define the air/water interface. Genetic algorithm is implemented with GAMBIT and FLUENT for shape optimization of hydrofoils. Maximization of lift to drag ratio is used as the optimization criteria. Optimized shapes are obtained for a hydrofoil in deep water at $Re = 10 \times 10^6$ and $Re = 20 \times 10^6$ at angles of attack of 0, 2, 4, and 6 degrees. Optimized shapes are also obtained for a hydrofoil near a free surface at $d/c = 0.5$ where d is the depth under the free surface and c is the chord length of the hydrofoil. It is shown that GA optimization technique is capable of accurately and efficiently finding the globally optimum hydrofoils.

Acknowledgments

First and foremost, I would like to thank my parents, Al and Juliet, for everything that they have done for me over the course of my entire life. Without them and their persistent philosophy that I should always invest in myself and my education, I would not be where I am today, and I certainly would not be writing these words now.

I would also like to thank everyone that has helped me with this project, both technically and mentally. Tudor Foote and Tim Wray, you have gone above and beyond to help me solve all of the intricacies of working with a genetic algorithm and writing a thesis in general. Zoë Roberts, I know that you will say that you didn't do anything to help me with this thesis, but you have done more than you could know just keeping me sane at the end of the day.

Thanks to my committee members, Dr. David Peters and Dr. Harold Brandon, for taking the time to help me achieve this monumental step in my life.

Finally, an enormous amount of gratitude goes to my advisor, Dr. Ramesh Agarwal. I know that working with me may not have always been a smooth experience for him, but I have learned a great deal about engineering and myself under his guidance. Thank you for teaching me that, no matter how well you plan, your project will always take longer than you think.

William T. Cocke

Washington University in St. Louis
May 2012

Contents

List of Tables

List of Figures

Chapter 1

1. Introduction

1.1 Motivation

Design optimization has been a subject of great interest in the aerospace industry for several decades since optimization can ultimately lead to lighter, faster, and more fuel-efficient aircraft and spacecraft. In recent years, it is increasingly being used in the design of large ships as well as for speedboats, hydrofoil-craft and sailing-craft. Although hydrofoil-craft have been around since Alexander Graham Bell designed and built the first of its kind in 1908, there still remains much room for improving the performance of a hydrofoil-craft. In recent decades hydrofoil-craft equipped with fully submerged hydrofoil wings have gone somewhat out of style; as a result there has been limited work on optimization of hydrofoil shapes. An excellent review of hydrofoils and hydrofoil-craft is given by Acosta [1]. Recently, there is renewed interest in the shape optimization of hydrofoils [5]. The optimized hydrofoil wing/craft has the potential of increasing the efficiency of marine vehicles allowing for greater loads to be transported faster with lower fuel consumption. Our goal is to optimize the shape of a hydrofoil for application in a sailing craft, with the objective of higher lift at lower speeds and low drag at higher speeds. This is achieved by implementing a genetic algorithm [4] in the commercial CFD solver ANSYS-FLUENT [7].

1.2 Brief Review of Literature

The theoretical work done on the study of the performance of hydrofoils and hydrofoil craft before 1973 is well documented in the review paper of Acosta [1] . All this work has been based on inviscid potential flow theory since it was done before the advent of Computational Fluid Dynamics. From 1980's onwards, hydrofoils have been studied using the more advanced CFD tools based on the solution of Euler and Navier-Stokes equations, e.g. the recent work of Arabshahi et al [2] and Mulvany et al [3] is based on the solution of Renolds-Averaged Navier Stokes (RANS) equations in conjunction with a turbulence model for computation of hydrofoil flow fields. There have been only few studies reported in the literature for shape optimization of hydrofoils. The thesis of Tozzi [5] described one such attempt for shape optimization of a hydrofoil in deep water. The focus of the present work is on shape optimization of hydrofoils for both in deep water and near a free surface. For this purpose, we employ a genetic algorithm [4]. The genetic algorithm code employed in our work is described in [6]. The flow field of the hydrofoil is computed by using the ANSYS flow solver FLUENT [7] and the mesh is generated by the ANSYS geometry modeling and grid generator software GAMBIT [8].

1.3 Scope of the Thesis

The objective of research in the thesis is to optimize the shape of hydrofoils in deep water and close to the free surface (air/water interface) in order to maximize their lift to drag ratio and thereby improve their aerodynamic performance. To achieve this objective, we implement a genetic algorithm based optimizer in conjunction with the commercial CFD solver ANSYS-FLUENT. Steady Reynolds-Averaged Navier-Stokes (RANS) equations in conjunction with a one-equations Spalart-Allmaras turbulence model [9] are solved by FLUENT. Geometry modeling and mesh generation in the computational domain is accomplished by ANSYS-GAMBIT. Volume of Fluid (VOF) method built in FLUENT is employed for the air/water interface. Computations are performed for hydrofoils at Reynolds number of ten-and-twenty million at angle of attack of 0, 2, 4, and 6 degree.

Chapter 2

2. Computational Approach for Shape Optimization of Hydrofoil

This chapter briefly describes the salient features of the Computational Fluid Dynamics (CFD) software and the Genetic Algorithm and their integration for shape optimization.

2.1 The Computational Fluid Dynamics Software

Computational Fluid Dynamics (CFD) is a branch of fluid mechanics that that employs the numerical methods to discretize the governing equations of fluid dynamics on a set of points (mesh) around or inside a body and solves these equations on a computer to determine the flow field.

In order to obtain the flow field solution (velocity, pressure, temperature, etc.) for a given geometry, CFD software generally has several modules-the geometry modeling software module, the grid or mesh generation software module, the flow field solver and the module for post processing the flow field data.

In this thesis, we employ the well known commercial CFD software ANSYS. ANSYS has a module for geometry modeling and mesh generation called "GAMBIT", the flow solver called "FLUENT" and the post-processing the software "CFPOST." For genetic algorithm, we have modified and employed code developed in the CFD lab at Washington University by Maschmeyer et al. [6]. Some of the details of the CFD software and genetic algorithm are described below.

2.2 Geometric Modeling and Mesh Generation

The commercially available software "Gambit" [8] from ANSYS Inc. is used to generate a hybrid unstructured/structured mesh for each hydrofoil considered by the genetic algorithm. A journal file is used to automatically produce a mesh that is transported to FLUENT to evaluate the lift and drag of a hydrofoil. The journal must be robust enough to create a good mesh around any arbitrary hydrofoil. The far-field boundary of this computational domain is kept the same for each calculation. An unstructured mesh is generated in the computational domain in the vicinity of the hydrofoil. When meshing is completed, the boundaries are defined and the 2D mesh is written to a file.

2.3 Flow Field Solutions

ANSYS CFD solver FLUENT [7] is employed for computation of the flow field of the mesh generated by GAMBIT. Reynolds-Averaged Navier-Stokes equations are solved in conjunction with the one-equation Spalart-Allmaras turbulence model by a second-order accurate finite-volume algorithm computed. The flow field data is then processed to determine the lift and drag coefficients - C_l and C_d - of a particular hydrofoil. A journal file is often written to automatically initialize and evaluate each hydrofoil in a given generation of genetic algorithm. The journal file initializes the calculations for a given free-stream velocity and angle of attack. Temperature and static pressure are defined at sea level conditions of 298.5 K and 101325 Pa respectively.

2.4 The Genetic Algorithm

2.4.1 Overview

Genetic algorithm (GA) based optimization software is employed to obtain an optimized hydrofoil shape at cruising speeds of a sailing craft. The GA software program optimizes a two dimensional hydrofoil shape to maximize its lift to drag ratio. For a given cruise speed and angle of attack, the program generates a series of families of hydrofoil shapes. The program generates a number of random hydrofoils in a given generation (e.g. 20) and calculates the flow field and lift to drag ratio for each hydrofoil in the generation using GAMBIT/FLUENT and then employs a fitness selection criteria (maximum lift to drag

ratio) to discard 50% of the hydrofoils from the generation, then employs cross-over and random mutation to adjust Bezier Curve control coordinate points that define the new hydrofoil shapes for the next generation. The process is continued until the convergence towards a globally optimal hydrofoil shape (with highest lift to drag ratio) is achieved. This process may require 50 to 100 generations or even more with each generation having 20 or more individuals (hydrofoils) before convergence is achieved.

2.4.2 Genetic Algorithms

Genetic algorithms are a class of stochastic optimization algorithms inspired by the biological evolution. In GA, a set or *generation* of input vectors, called *individuals*, is iterated over, successively combining traits (aspects) of the best individuals until a convergence is achieved. In general, GA employs the following steps [4].

1. **Initialization**: Randomly create N individuals.
2. **Evaluation:** Evaluate the fitness of each individual.
3. **Natural selection:** Remove a subset of the individuals. Often the individuals that have the lowest fitness are removed; although culling, the removing of those individuals with similar fitness, is sometimes performed.
4. **Reproduction:** Pick pairs of individuals to produce an offspring. This is often done by roulette wheel sampling; that is, the probability of selecting some individual h_i for reproduction is given by:

$$P[h_i] = \frac{fitness(h_i)}{\sum_j fitness(h_j)} \qquad (1)$$

A crossover function is then performed to produce the offspring. Generally, crossover is implemented by choosing a crossover point on each individual and swapping alleles – or vector elements – at this point as illustrated in Figure1.

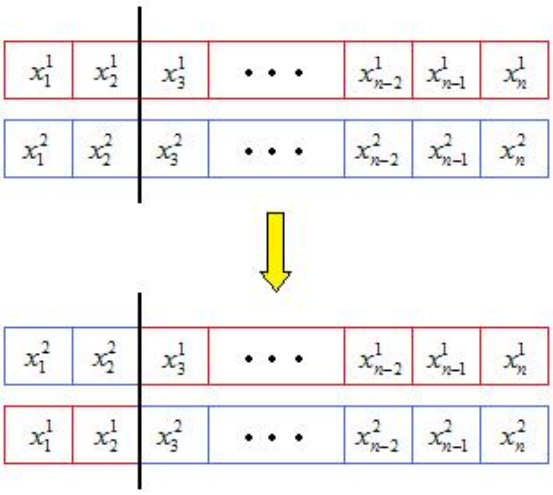

Figure 1: Illustration of the general crossover function in GA

5. **Mutation:** Randomly alter some small percentage of the population.

6. **Check for Convergence:** If the solution has converged, return the best individual observed. If the solution has not yet converged, label the new generation as the current generation and go to step 2. Convergence is achieved after a certain number of generations.

2.4.3 Hydrofoil Representation by a Bezier Curve

In all the optimization studies reported in this thesis, Bezier curves are used to obtain different hydrofoil shapes in a given generation. A "Bezier" Curve is defined by the equation

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i \qquad t \in [0,1]. \qquad (2)$$

Two smooth curves are able to represent the top and bottom surface of the hydrofoil. In our work, four control points were used for both the top and bottom surface of the hydrofoil to create the Bezier Curve. The coordinates for these points were:

$X_0 = 0, X_1 = 0, 0 \leq X_2 \leq 0.98, X_3 = 1$

$Y_0 = 0, 0.01 \leq Y_1 \leq 0.3, -0.1 \leq Y_2 \leq 0.4, Y_3 = 0$

for the curve representing the top surface of the hydrofoil and

$X_0 = 0, X_1 = 0, 0.1 \leq X_2 \leq 0.98, 0.1 \leq X_3 \leq 0.98, X_4 = 1$

$Y_0 = 0, -0.3 \leq Y_1 \leq 0, -0.2 \leq Y_2 \leq 0.4, -0.2 \leq Y_2 \leq 0.4, Y_4 = 0$

for the curve representing the bottom surface of the hydrofoil.

For both surfaces,

$X_0 \leq X_1 \leq X_2 \leq X_3$ and $Y_{i,bottom} < Y_{i,top}$

Fifty points were spaced equally along each curve. These points were later imported into GAMBIT. This representation was sufficient to represent any hydrofoil shape for the genetic algorithm. In addition, a constraint on the thickness of the hydrofoil was imposed; it allowed the variation in thickness by a very small percentage.

2.4.4 Evaluation of an Individual Hydrofoil Shape

Figure 2 shows the schematics of the optimization process employing GA with GAMBIT/FLUENT.

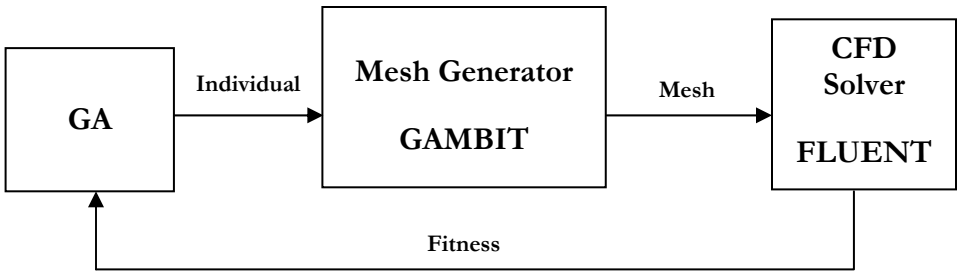

Figure 2: Schematic of Information Flow in the Genetic Algorithm Optimization Process

For each individual hydrofoil, the genetic algorithm (GA) first creates a data file with all the points representing the top and bottom curves of the hydrofoil. A batch file is then run which opens Gambit, imports the data file into the program, and runs a journal file as described in Section 2.2. Once the GA reads a mesh file in the workspace, the batch file running Fluent is started. This file opens Fluent, imports the mesh file, and runs a journal file as described in Section 2.3. The GA reads the lift and drag coefficients from two files,

written by Fluent, named "cl-history" and "cd-history" respectively. The fitness value is determined by taking the lift-to-drag ratio.

2.4.5 Advancing to the Next Generation with Crossover and Mutation

In our work, a generation size of 20 individuals (hydrofoils) was employed in GA. A natural selection rate of 50% and a mutation rate of 4% was used with no culling tolerance. The 50% of the individuals (hydrofoils) with low values of fitness (C_l/C_d) were removed after each generation. The discarded 50% of the individuals were replaced by employing the extrapolation-based crossover scheme.

Since the fitness value can be both positive and negative, roulette wheel sampling cannot be easily used for selecting the reproducing individuals; there, reproduction is done by randomly selecting two individuals. Note, however, that the fittest individuals are still most likely to reproduce because the top 50% of each generation perpetuates to the next generation (and thus will have another chance to reproduce). The offspring individual is then obtained by stepping a random amount in the direction of the fitter parent according to equation (3) below.

$$crossover(x_1, x_2) = rand(0,1) \cdot (x_2 - x_1) + x_2 \qquad (3)$$

In our numerical experiments, GA converges in approximately 100 generations.

Chapter 3

3. Results for Optimized Hydrofoil Shapes for Deep Water

This chapter shows the results for optimized hydrofoil shape for deep water (without free surface) using a genetic algorithm. It should be mentioned that similar results were previously obtained by Zach Moscicki in 2010 when he was working as an undergraduate research student under the supervision of Ramesh Agarwal [10]. The results are for flow speeds of ten meters per second and twenty meters per second at angle of attack of zero, two, four, and six degrees. Figure 3 shows the typical evolution towards an optimized hydrofoil in GA at flow speed of twenty meters per second and zero degree angle of attack. The free stream Reynolds number of this flow is 20 million.

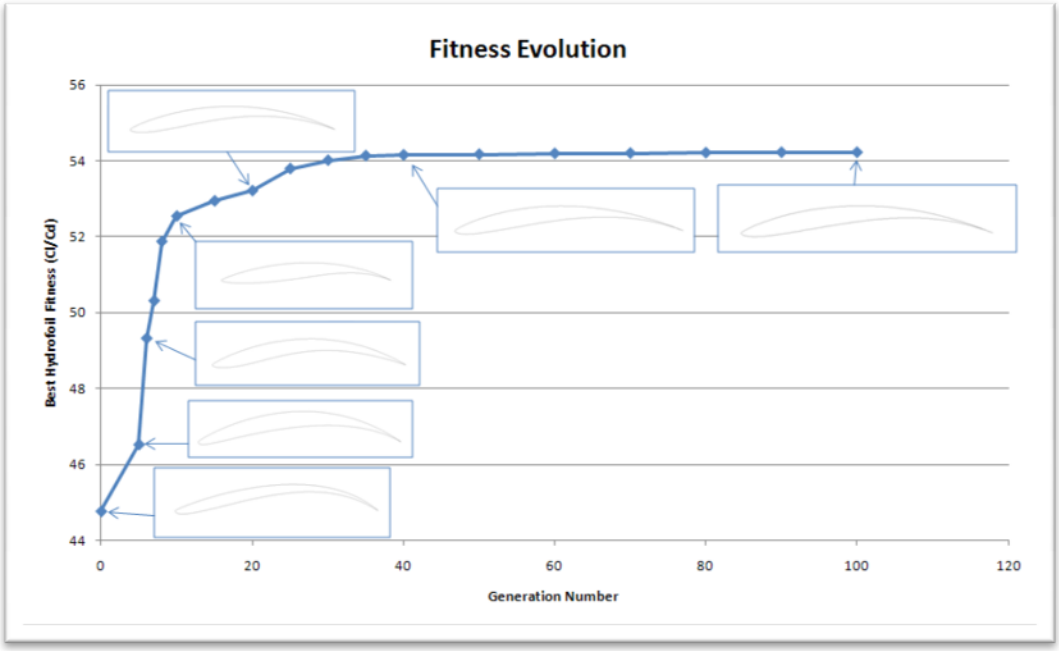

Figure 3: Optimal evolution of hydrofoil using GA for maximum lift to drag ratio, V_∞ = 20 m/s, Angle of Attack $\alpha = 0^o$, Re = 10 x 10^6, Deep Water

3.1 Optimized Hydrofoil Shape at V_∞ = 10 m/s, $\alpha = 0^o$ and $Re_\infty = 10 \times 10^6$

Figures 4(a) – 4(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 51.04.

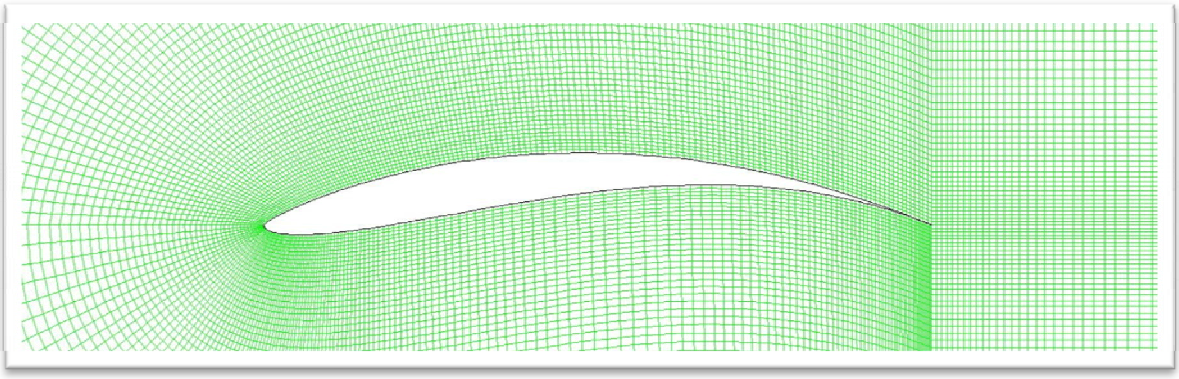

Figure 4(a): Mesh for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 0^o$, $Re_\infty = 10 \times 10^6$, Deep Water

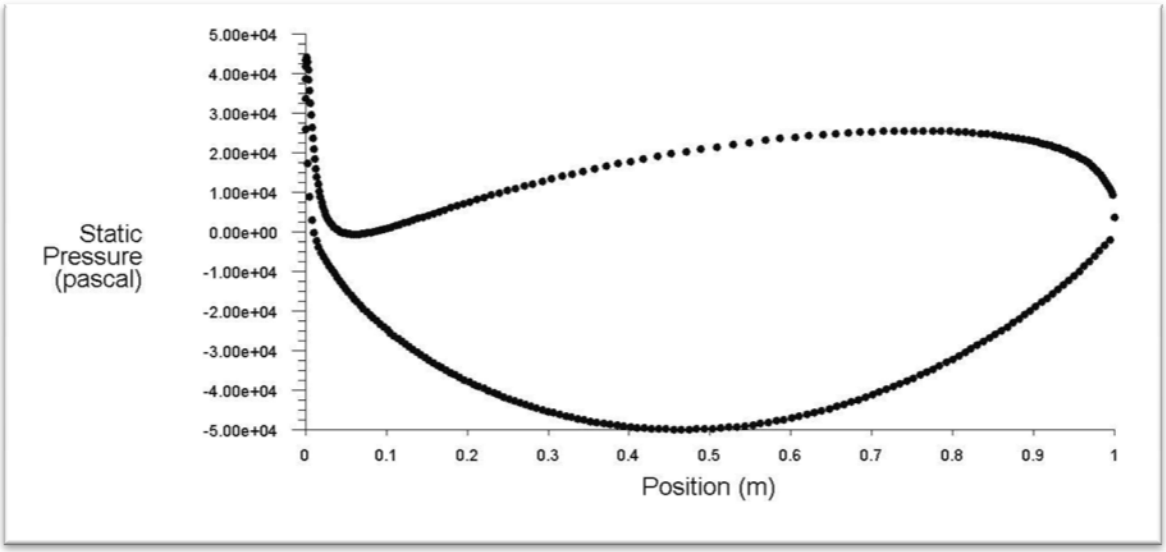

Figure 4(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 0^o$, $Re_\infty = 10 \times 10^6$, Deep Water

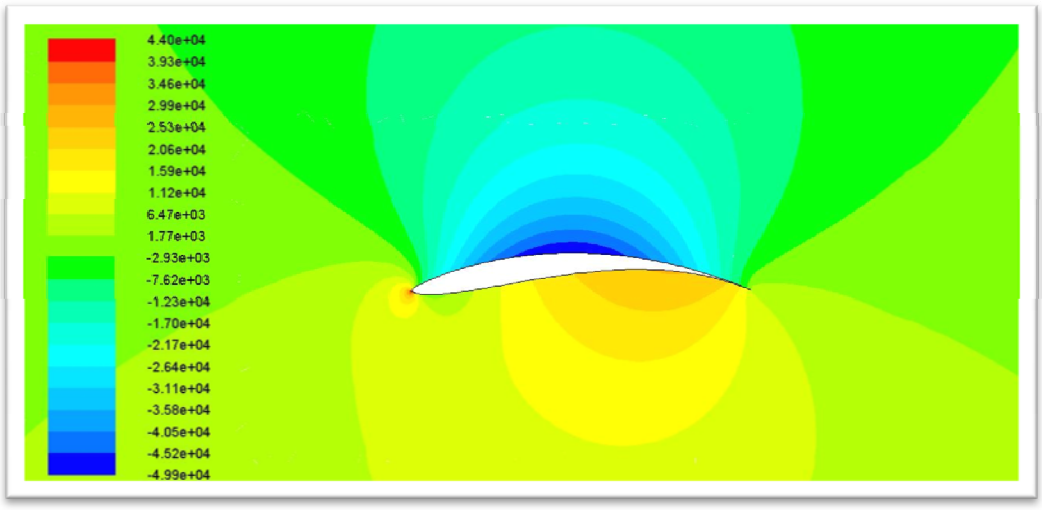

Figure 4(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 0^o$, $Re_\infty = 10 \times 10^6$, Deep Water

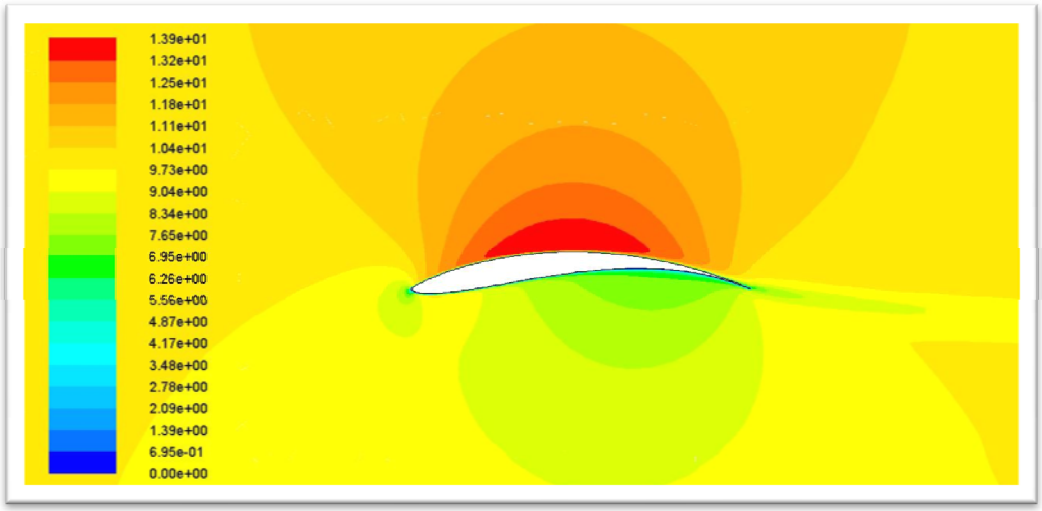

Figure 4(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 0^0$, $Re_\infty = 10 \times 10^6$, Deep Water

3.2 Optimized Hydrofoil Shape at V_∞ = 10 m/s, $\alpha = 2^o$ and $Re_\infty = 10 \times 10^6$

Figures 5(a) – 5(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 54.96.

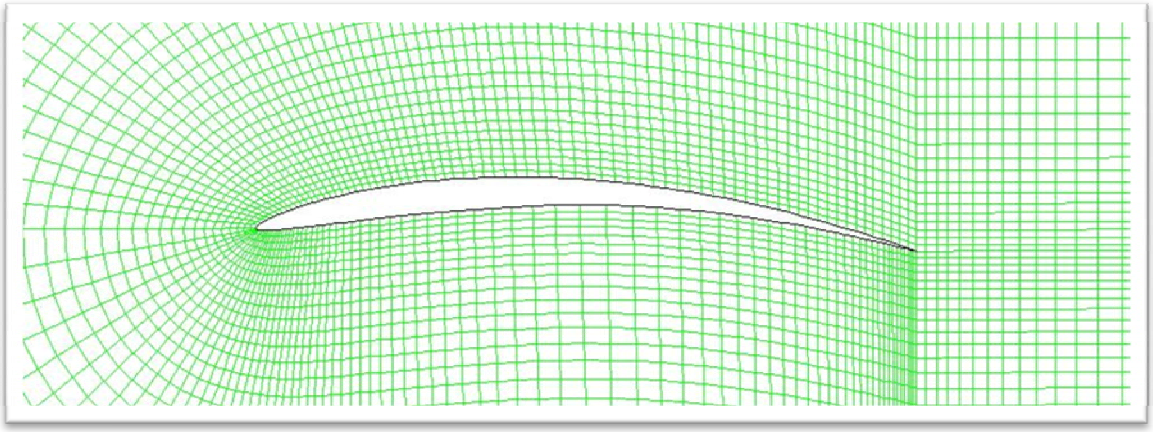

Figure 5(a): Mesh for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 2^o$, $Re_\infty = 10 \times 10^6$, Deep Water

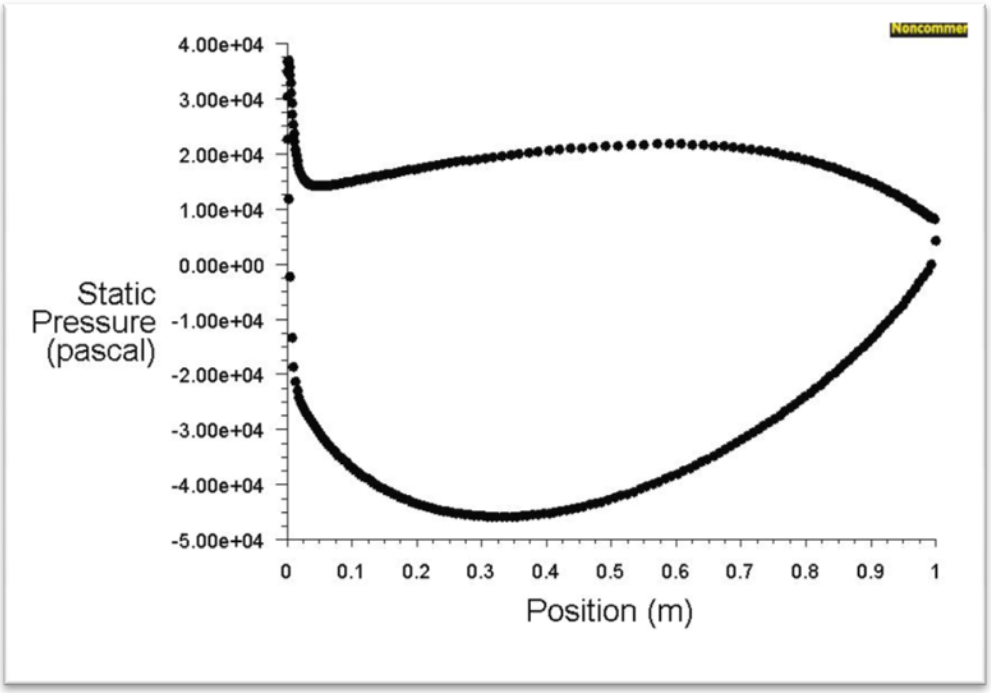

Figure 5(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 2^o$, $Re_\infty = 10 \times 10^6$, Deep Water

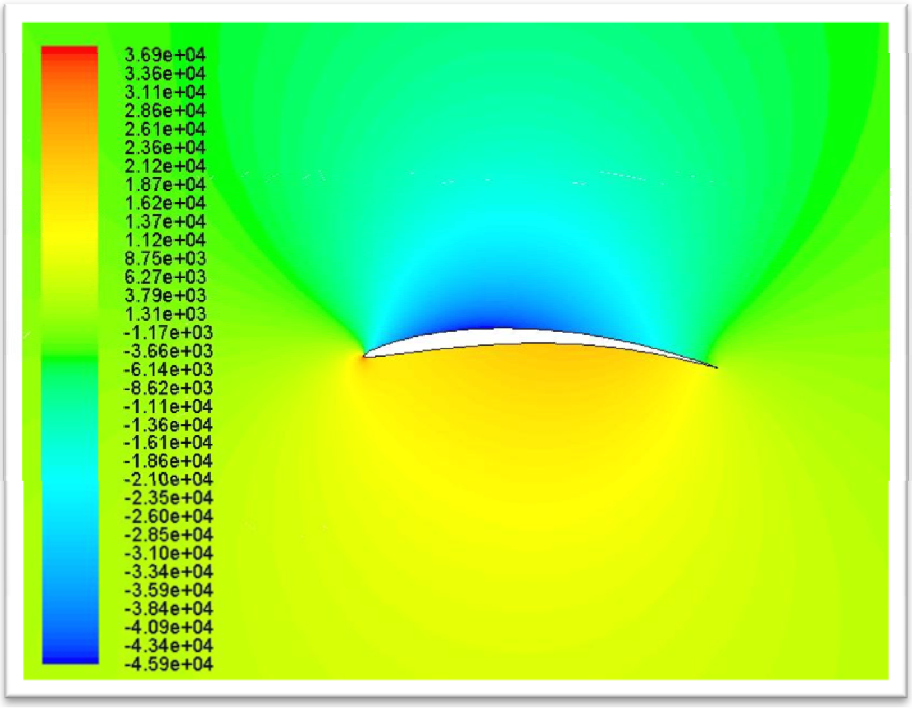

Figure 5(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 2^o$, $Re_\infty = 10 \times 10^6$, Deep Water

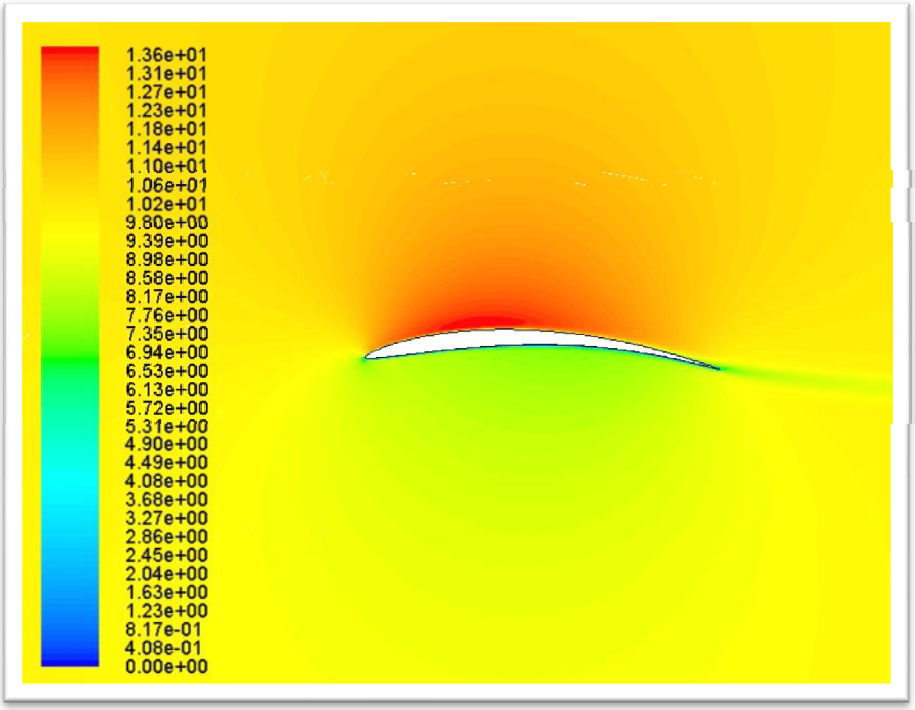

Figure 5(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 2^0$, $Re_\infty = 10 \times 10^6$, Deep Water

3.3 Optimized Hydrofoil Shape at V_∞ = 10 m/s, $\alpha = 4^o$ and $Re_\infty = 10 \times 10^6$

Figures 6(a) – 6(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 45.71.

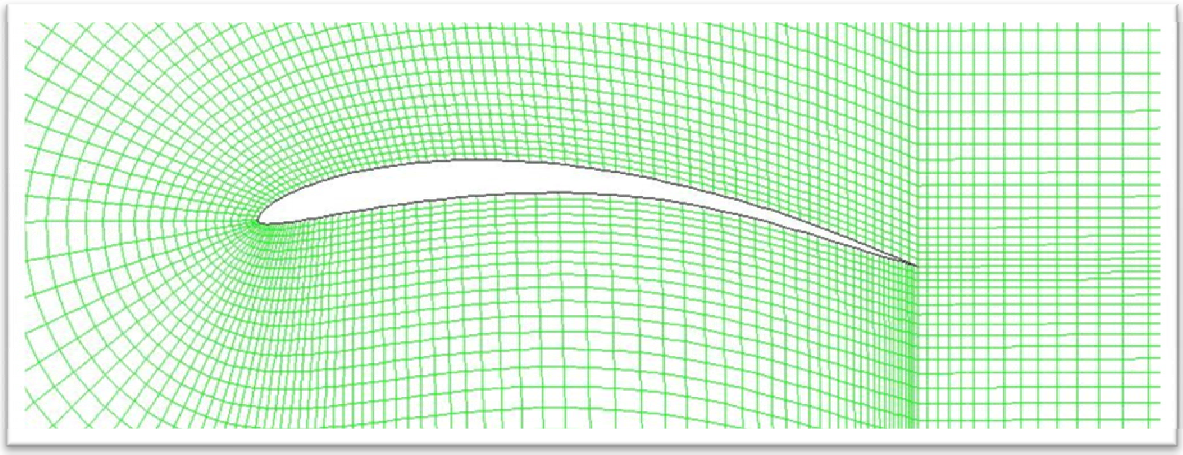

Figure 6(a): Mesh for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 4^o$, $Re_\infty = 10 \times 10^6$, Deep Water

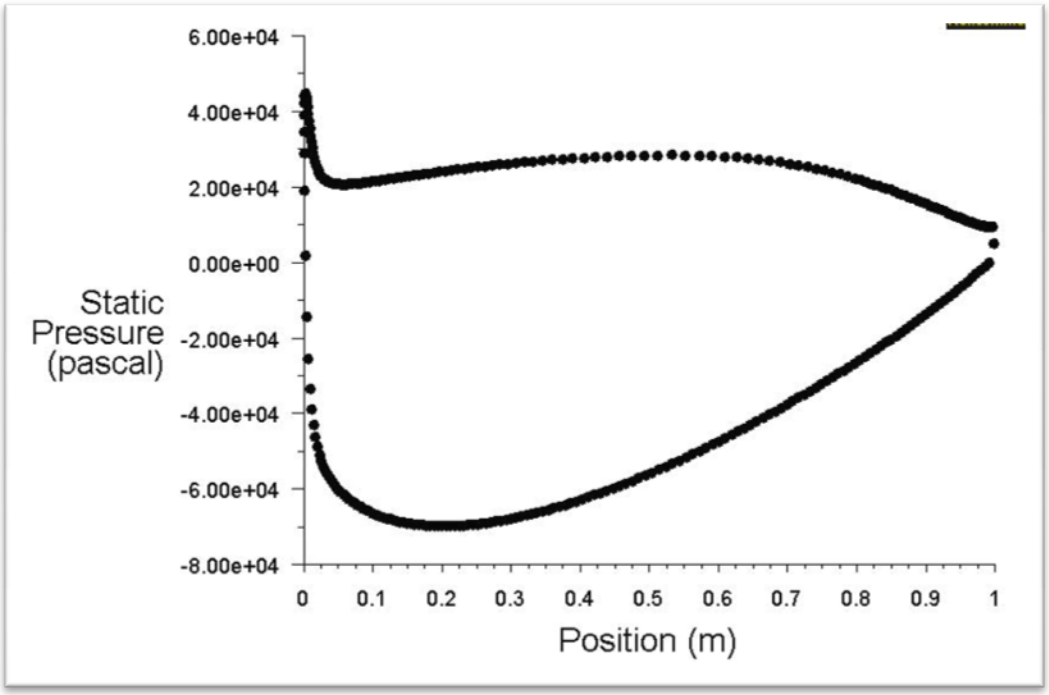

Figure 6(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 4^o$, $Re_\infty = 10 \times 10^6$, Deep Water

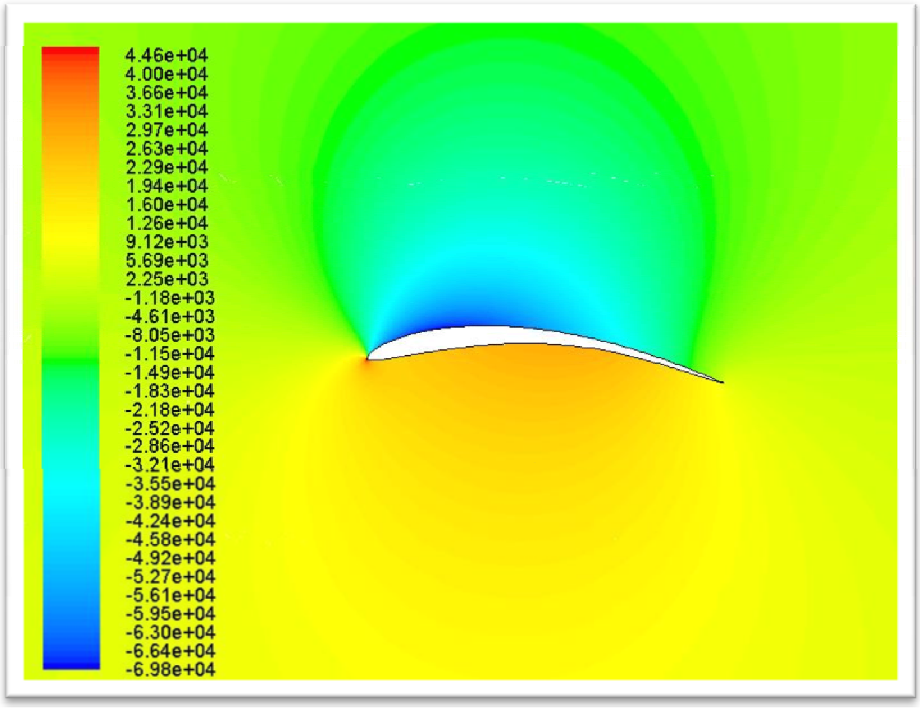

Figure 6(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 4^o$, $Re_\infty = 10 \times 10^6$, Deep Water

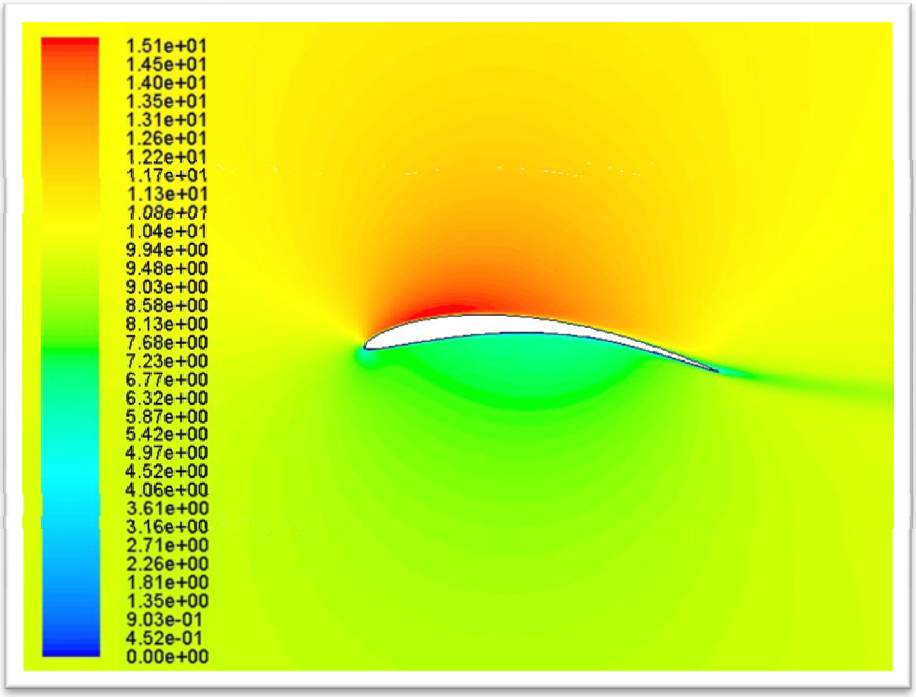

Figure 6(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 4^0$, $Re_\infty = 10 \times 10^6$, Deep Water

3.4 Optimized Hydrofoil Shape at V_∞ = 10 m/s, $\alpha = 6^o$ and $Re_\infty = 10 \times 10^6$

Figures 7(a) – 7(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 41.55.

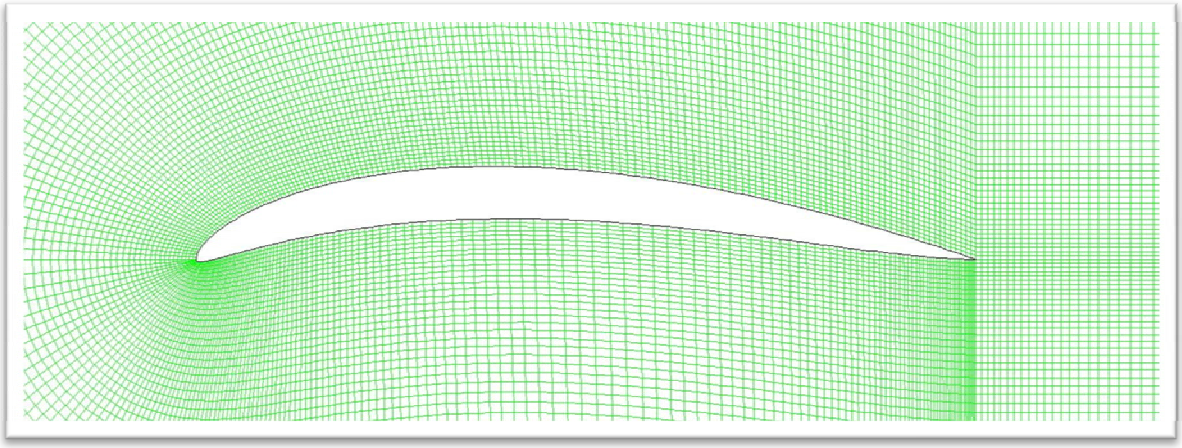

Figure 7(a): Mesh for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 6^o$, $Re_\infty = 10 \times 10^6$, Deep Water

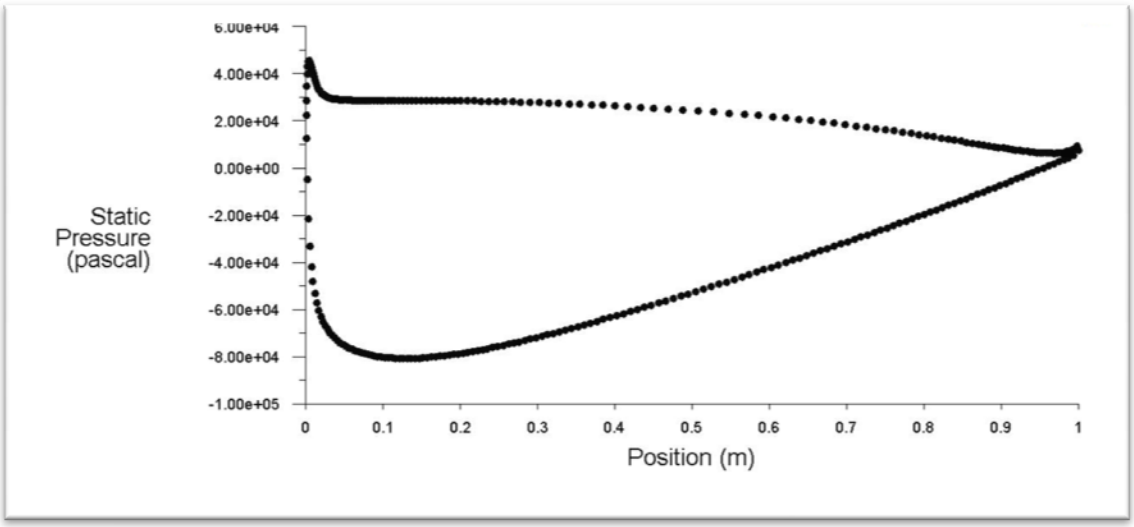

Figure 7(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 6^o$, $Re_\infty = 10 \times 10^6$, Deep Water

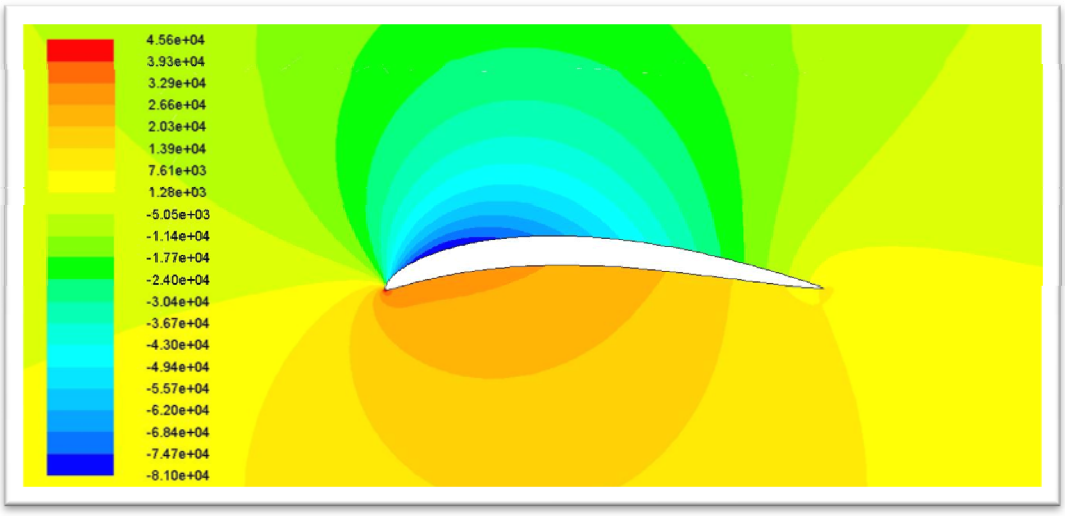

Figure 7(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 6^o$, $Re_\infty = 10 \times 10^6$, Deep Water

Figure 7(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 6^0$, $Re_\infty = 10 \times 10^6$, Deep Water

3.5 Optimized Hydrofoil Shape at V_∞ = 20 m/s, $\alpha = 0^o$ and $Re_\infty = 20 \times 10^6$

Figures 8(a) – 8(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 54.22.

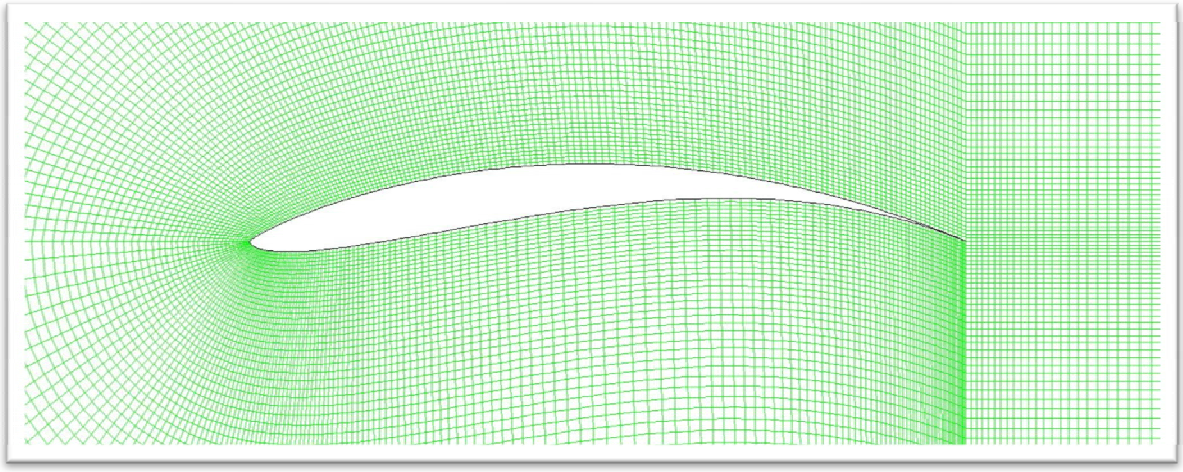

Figure 8(a): Mesh for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 0^o$, $Re_\infty = 20 \times 10^6$, Deep Water

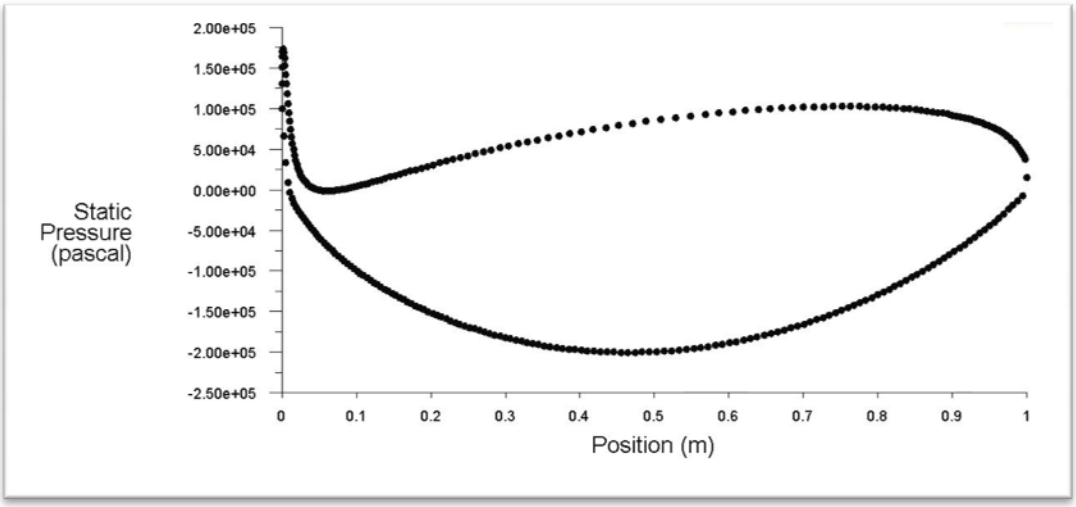

Figure 8(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 0^o$, $Re_\infty = 20 \times 10^6$, Deep Water

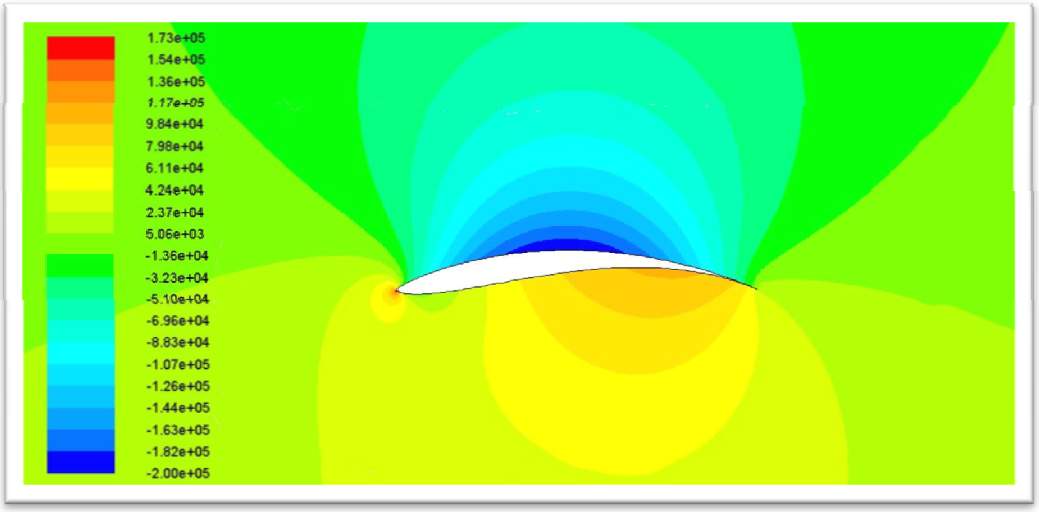

Figure 8(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 0^o$, $Re_\infty = 20 \times 10^6$, Deep Water

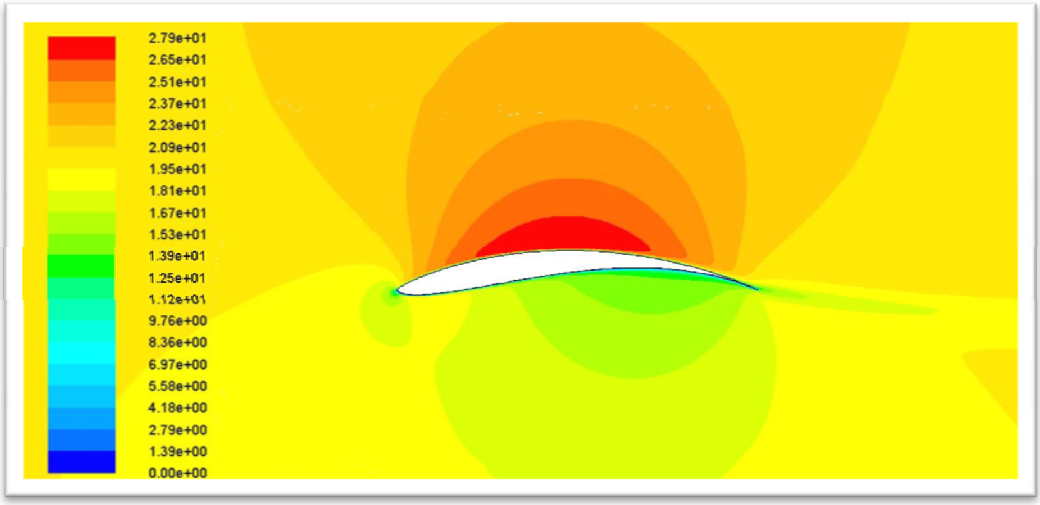

Figure 8(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 0^0$, $Re_\infty = 20 \times 10^6$, Deep Water

3.6 Optimized Hydrofoil Shape at V_∞ = 20 m/s, $\alpha = 2^o$ and $Re_\infty = 20 \times 10^6$

Figures 9(a) – 9(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 54.38.

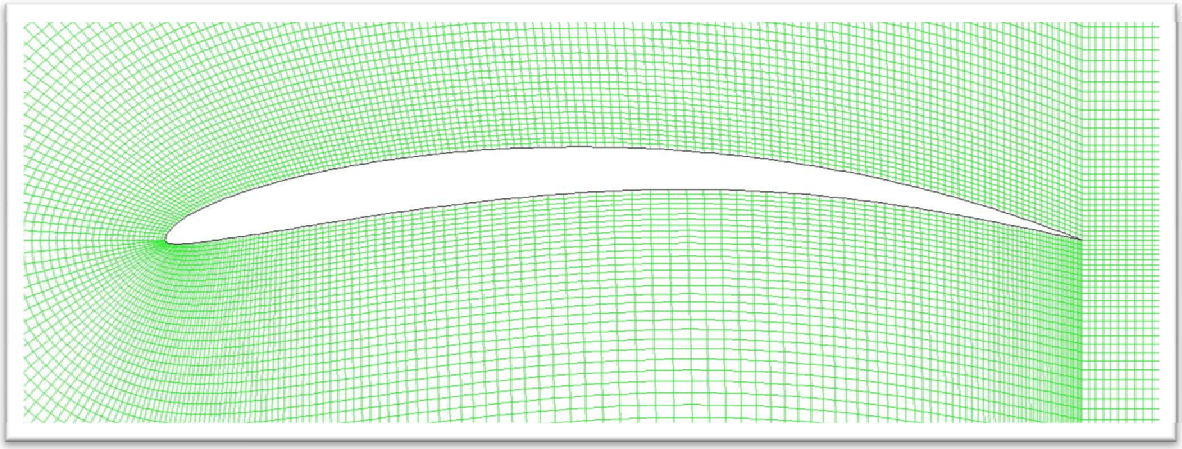

Figure 9(a): Mesh for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 2^o$, $Re_\infty = 20 \times 10^6$, Deep Water

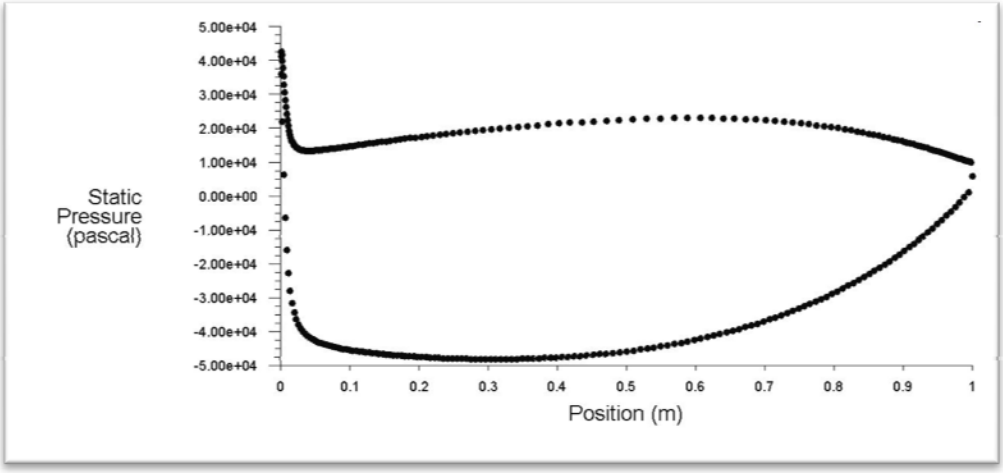

Figure 9(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 2^o$, $Re_\infty = 20 \times 10^6$, Deep Water

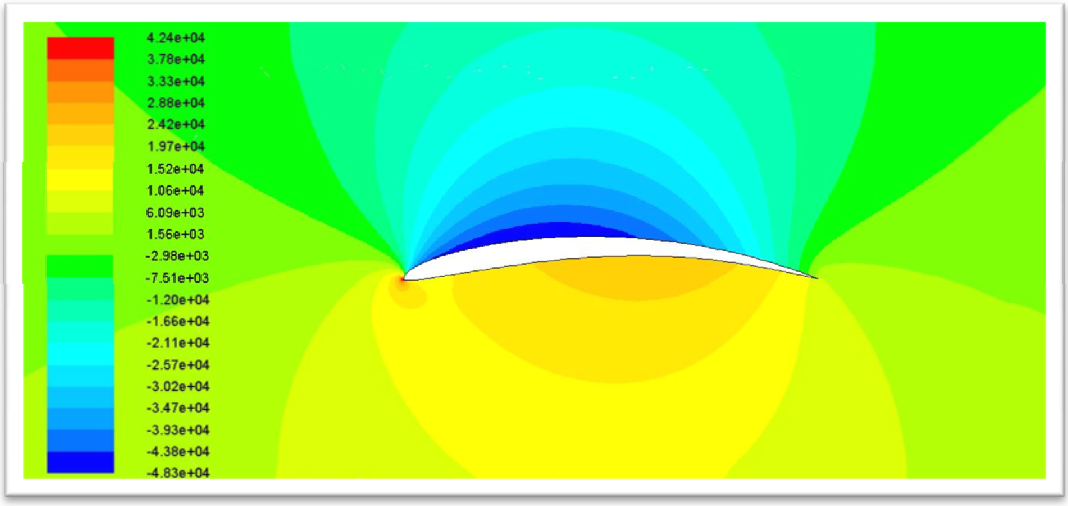

Figure 9(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 2^o$, $Re_\infty = 20 \times 10^6$, Deep Water

Figure 9(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 2^0$, $Re_\infty = 20 \times 10^6$, Deep Water

3.7 Optimized Hydrofoil Shape at V_∞ = 20 m/s, $\alpha = 4^o$ and $Re_\infty = 20 \times 10^6$

Figures 10(a) – 10(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 49.92.

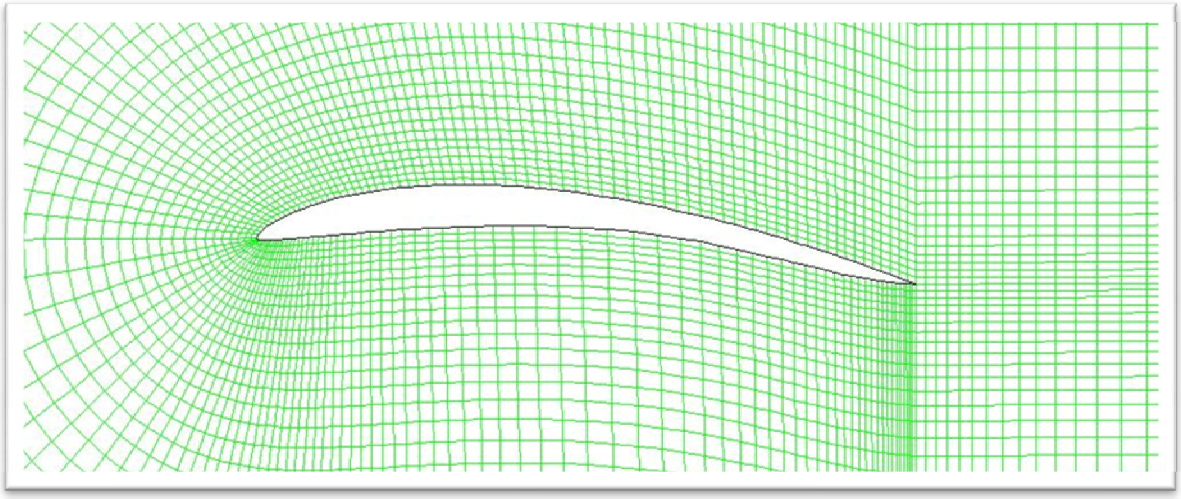

Figure 10(a): Mesh for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 4^o$, $Re_\infty = 20 \times 10^6$, Deep Water

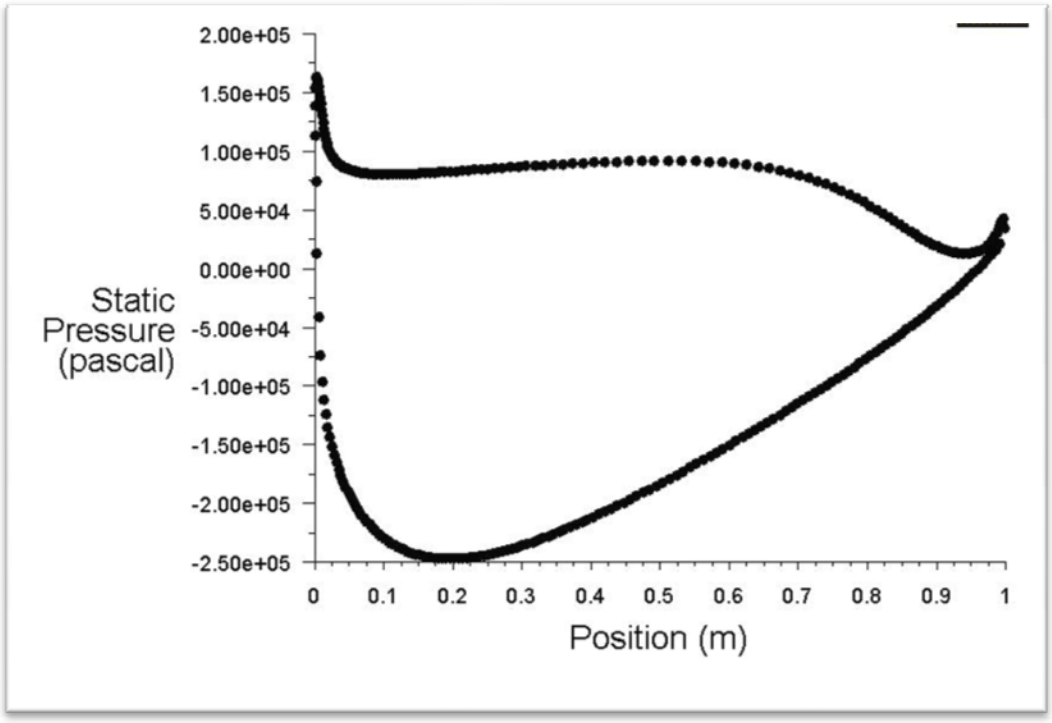

Figure 10(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 4^o$, $Re_\infty = 20 \times 10^6$, Deep Water

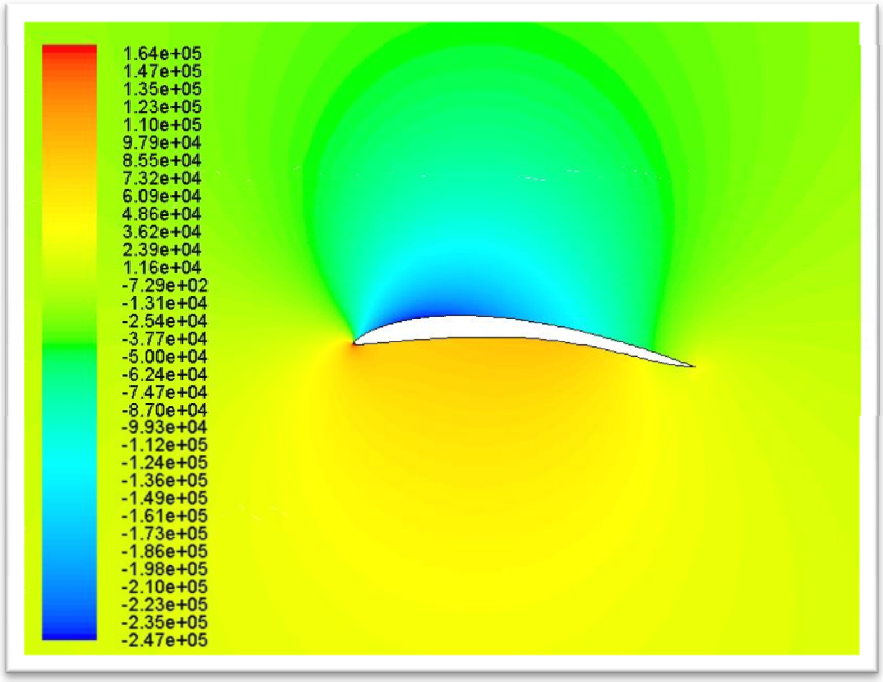

Figure 10(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 4^o$, $Re_\infty = 20 \times 10^6$, Deep Water

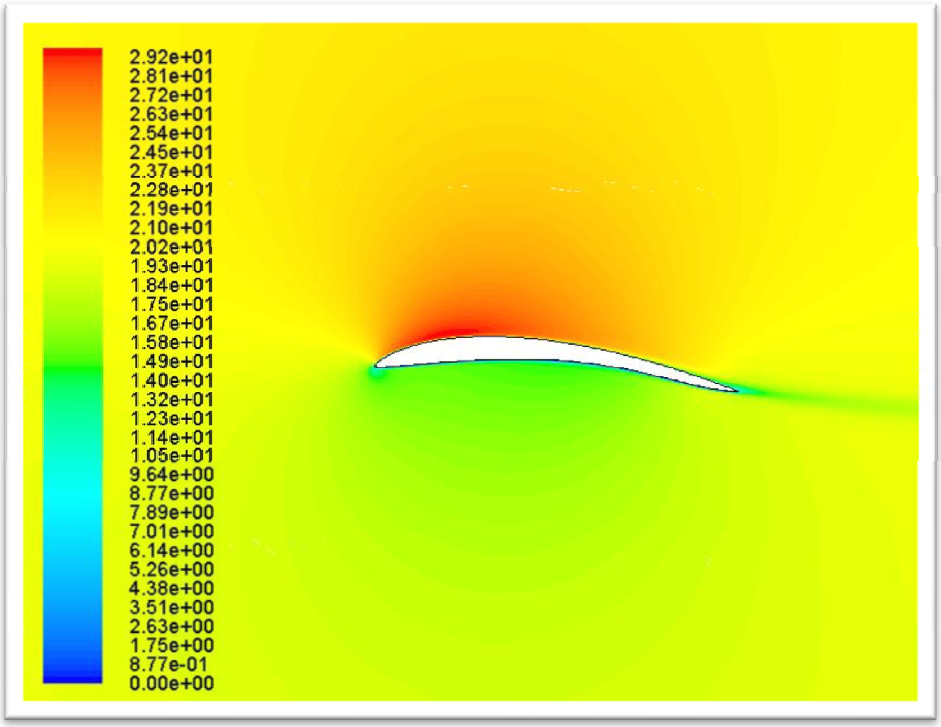

Figure 10(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 4^0$, $Re_\infty = 20 \times 10^6$, Deep Water

3.8 Optimized Hydrofoil Shape at V_∞ = 20 m/s, $\alpha = 6^o$ and $Re_\infty = 20 \times 10^6$

Figures 11(a) – 11(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 42.01.

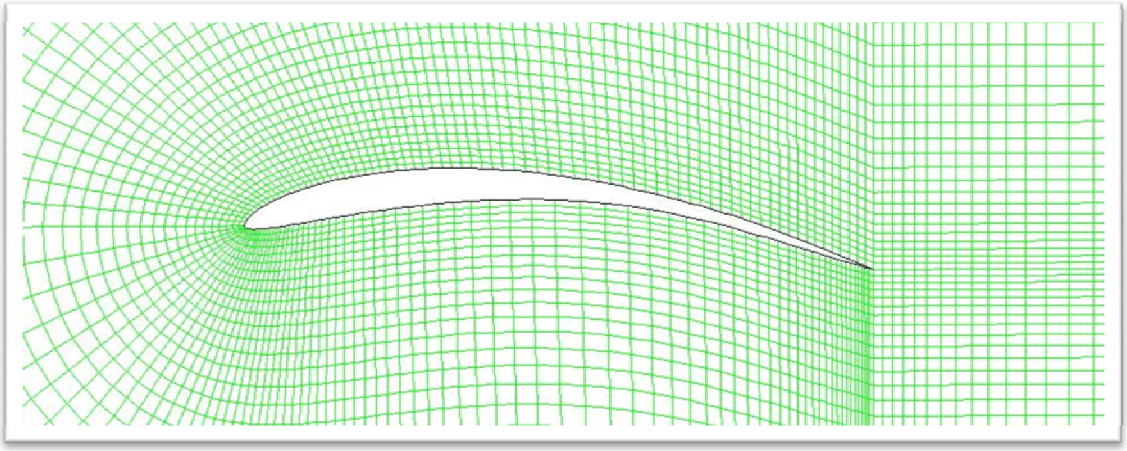

Figure 11(a): Mesh for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 6^o$, $Re_\infty = 20 \times 10^6$, Deep Water

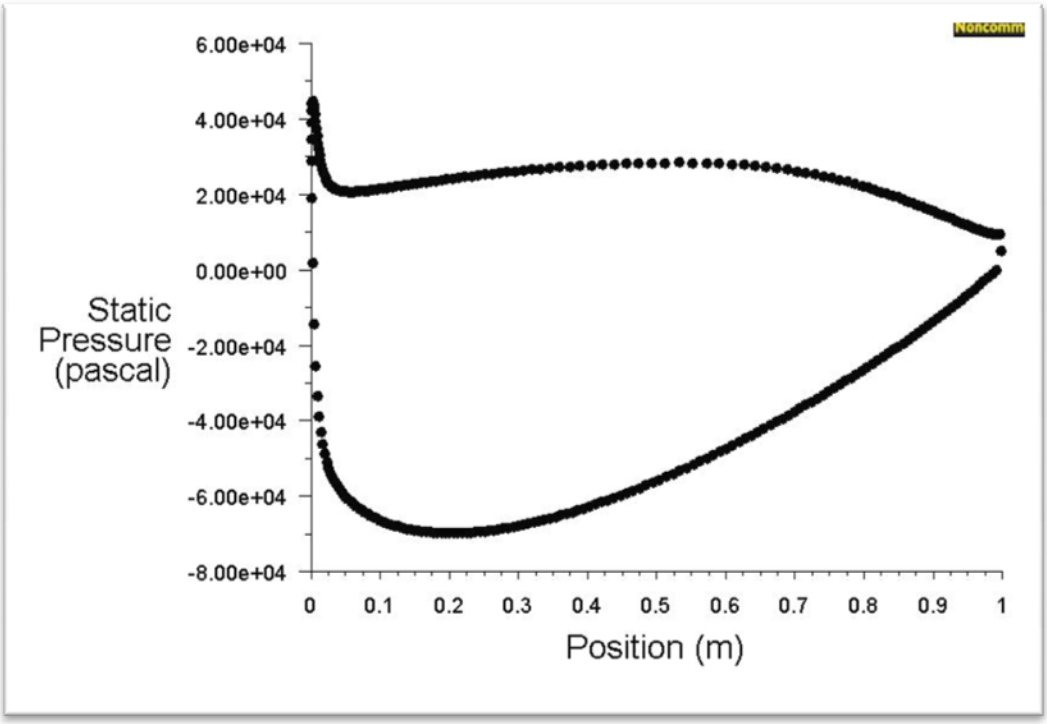

Figure 11(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 6^o$, $Re_\infty = 20 \times 10^6$, Deep Water

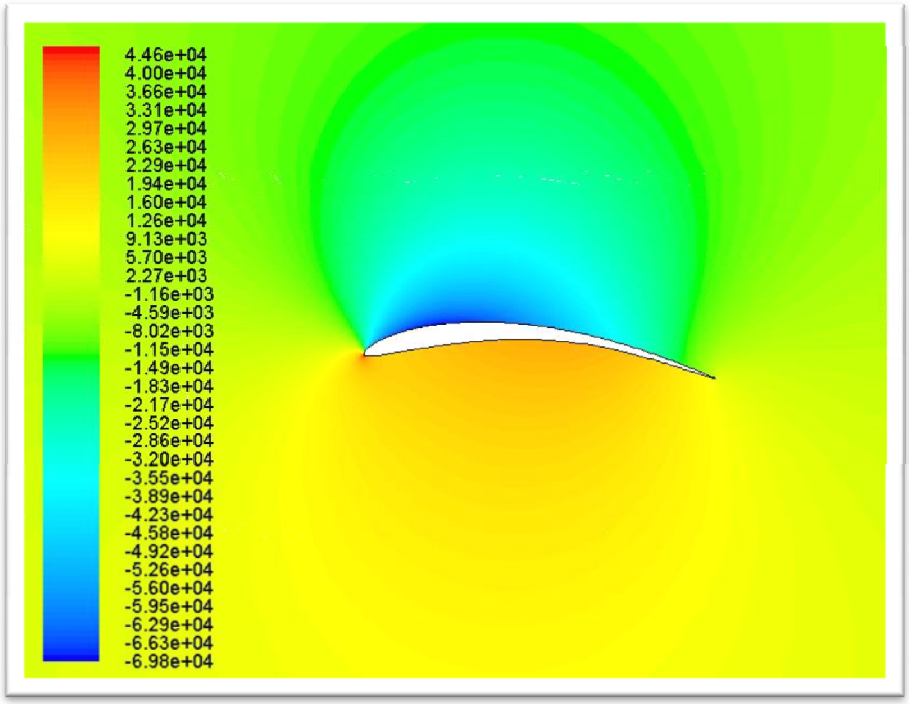

Figure 11(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 6^o$, $Re_\infty = 20 \times 10^6$, Deep Water

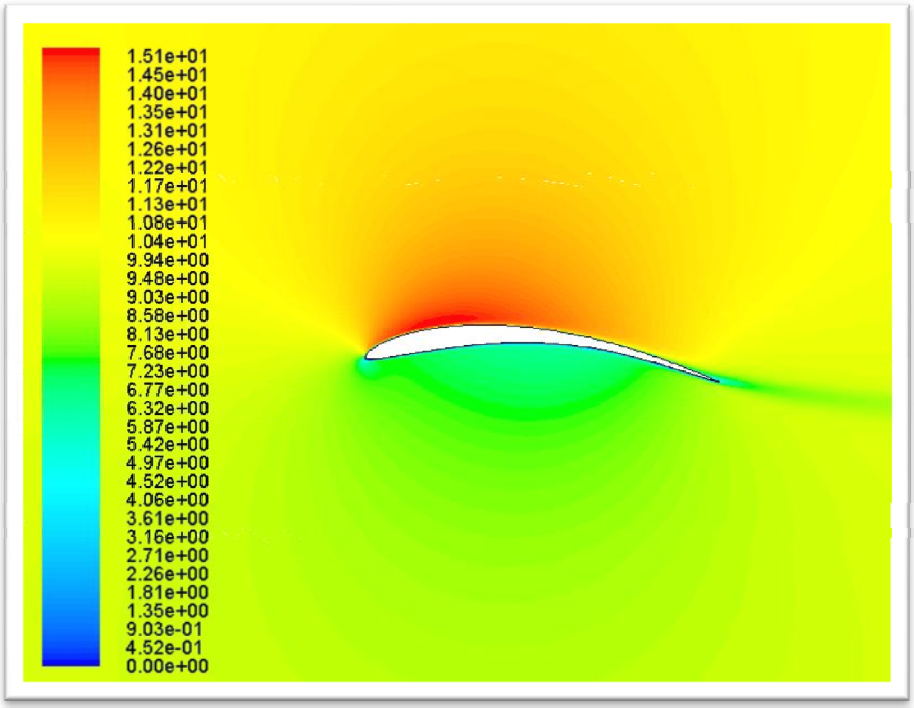

Figure 11(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 0^0$, $Re_\infty = 10 \times 10^6$, Deep Water

Table 1 gives the C_l/C_d values of the baseline hydrofoils and optimized hydrofoils in deep water for $Re_\infty = 10 \times 10^6$ and 20×10^6 at angles of attack of $\alpha = 0$, 2, 4 and 6 degree.

Table 1: C_l/C_d for Baseline and Optimized Hydrofoil Shapes for Deep Water

(Cl/Cd)	Baseline Hydrofoil in Deep Water		Hydrofoil Optimized for Deep Water	
	$Re_\infty = 10$ Million	$Re_\infty = 20$ Million	$Re_\infty = 10$ Million	$Re_\infty = 20$ Million
Zero Degrees	**36.12**	**37.83**	**51.04**	**54.22**
Two Degrees	**38.03**	**39.53**	**54.96**	**54.38**
Four Degrees	**34.02**	**35.01**	**45.71**	**49.92**
Six Degrees	**24.82**	**28.17**	**41.55**	**42.01**

Chapter 4

4. Results for Optimized Hydrofoil Shapes Including the Effect of Free Surface

This chapter shows the computations of optimized hydrofoil shapes when the influence of free surface is included. The results are shown for flow speed of ten meters per second and twenty meters per second for angle of attack of zero, two, four, and six degrees. The hydrofoils are 0.5 meters below the free surface (air/water interface). Figure 12 shows the typical evolutions of an optimized hydrofoil at V_∞ = 10 m/s, $\alpha = 0^o$, $Re_\infty = 10 \times 10^6$, and d/c = 0.5 where d is the depth of the hydrofoil from free surface and c is the chord length.

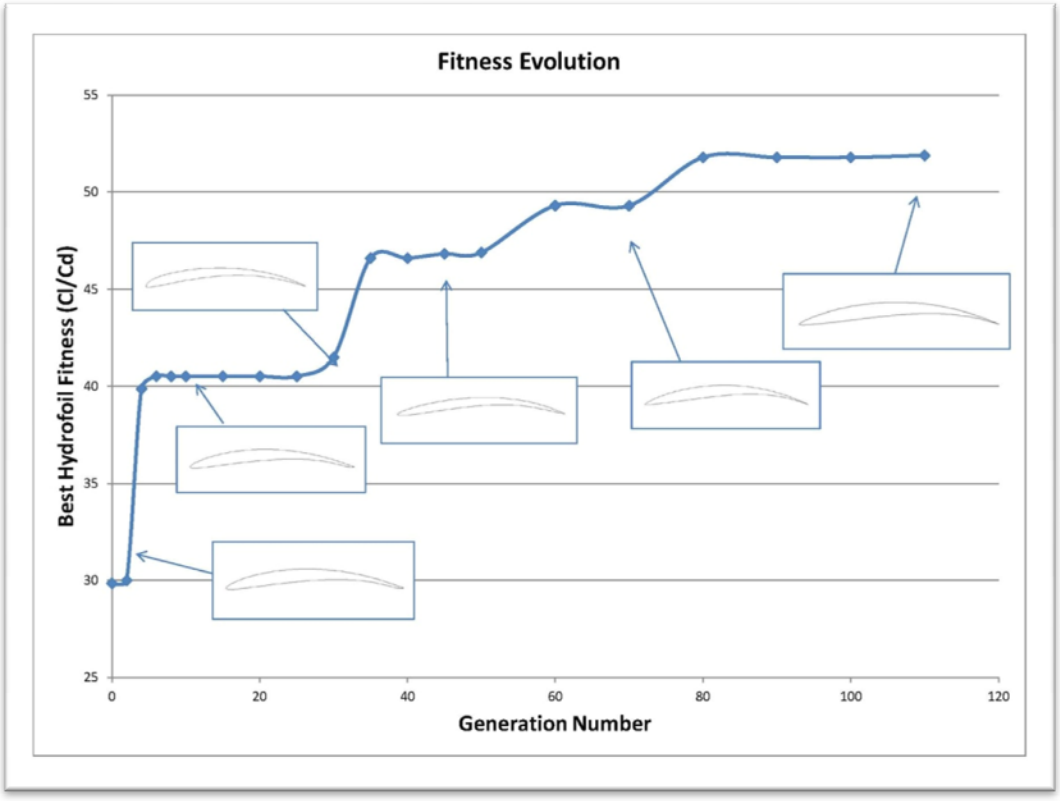

Figure 12: Optimal evolution of a hydrofoil using GA for maximum lift to drag ratio, V_∞ = 10 m/s, $\alpha = 0^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

4.1 Optimized Hydrofoil Shape at V_∞ = 10 m/s, $\alpha = 0^o$ and $Re_\infty = 10 \times 10^6$, d/c = 0.5

Figures 13(a) – 13(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 51.88.

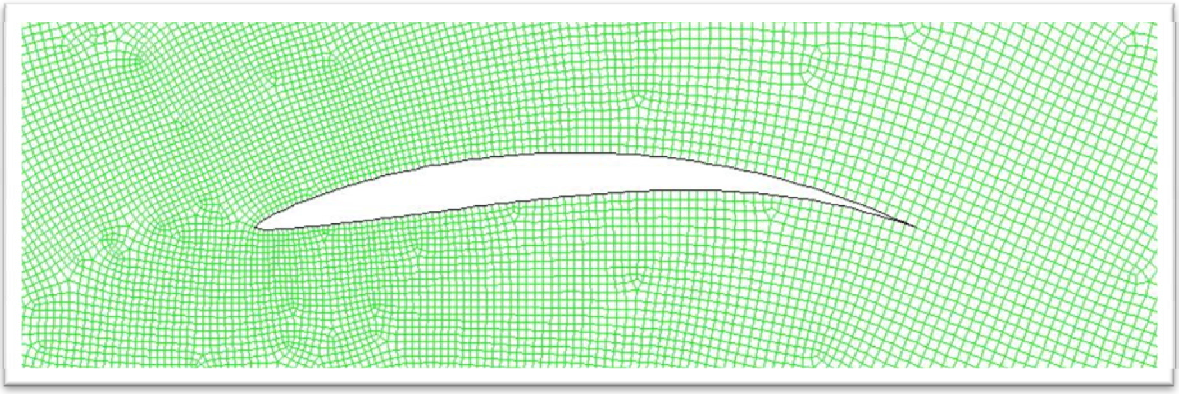

Figure 13(a): Mesh for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 0^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

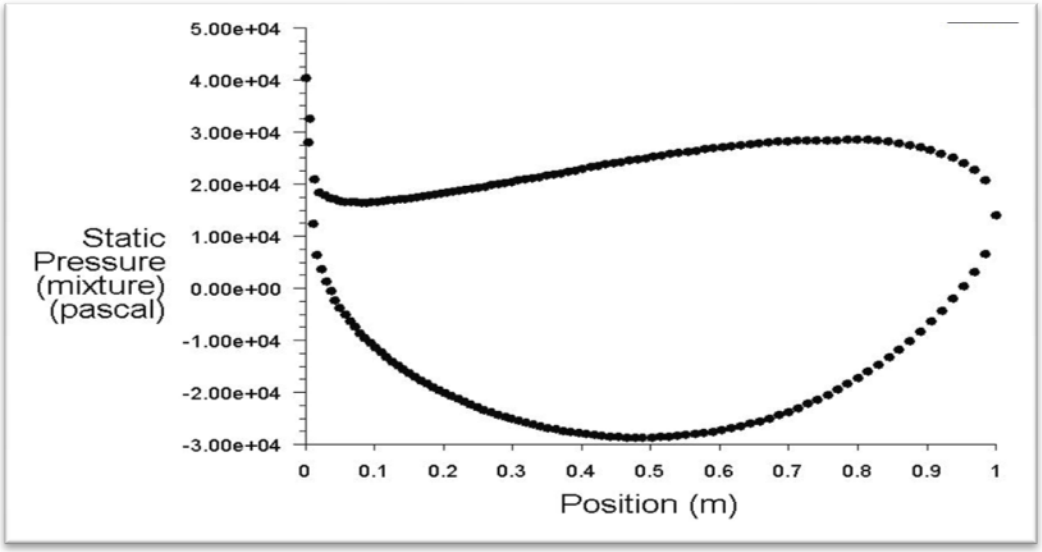

Figure 13(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 0^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

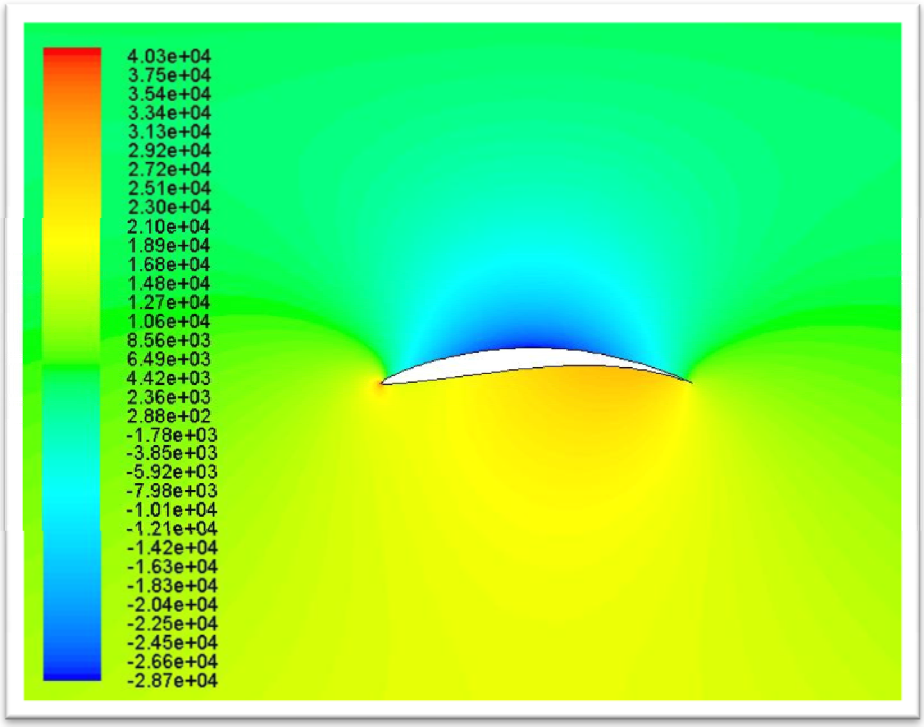

Figure 13(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 0^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

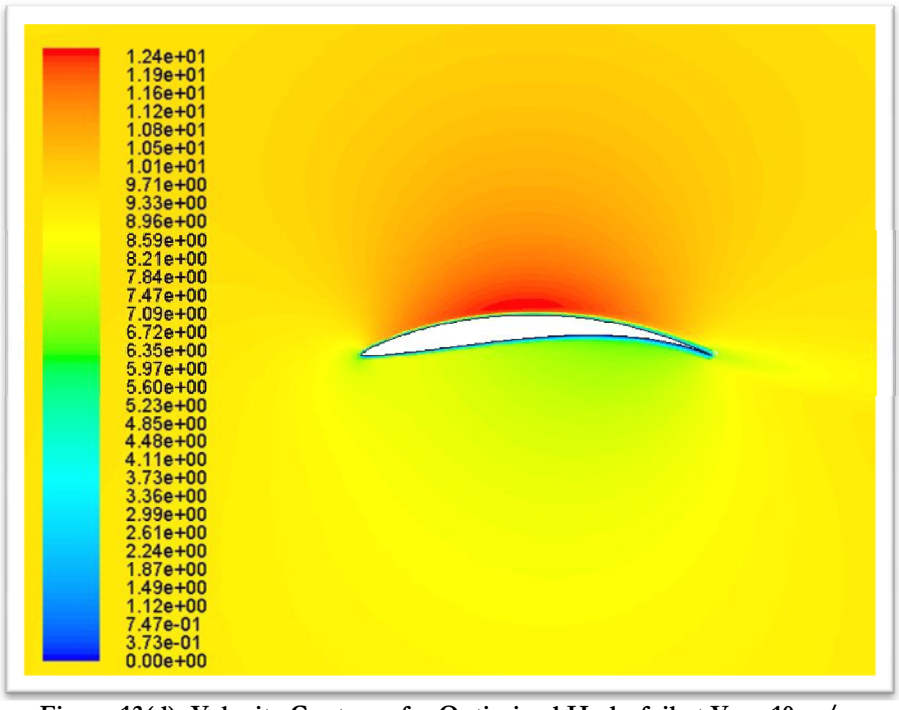

Figure 13(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 0^0$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

4.2 Optimized Hydrofoil Shape at V_∞ = 10 m/s, $\alpha = 2^o$ and $Re_\infty = 10 \times 10^6$, d/c = 0.5

Figures 14(a) – 14(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 52.03.

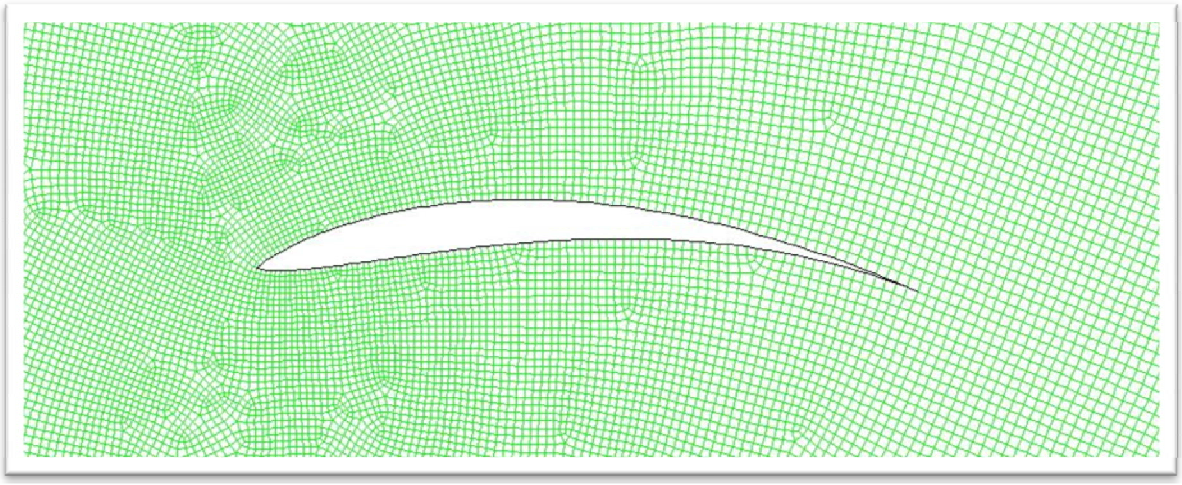

Figure 14(a): Mesh for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 2^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

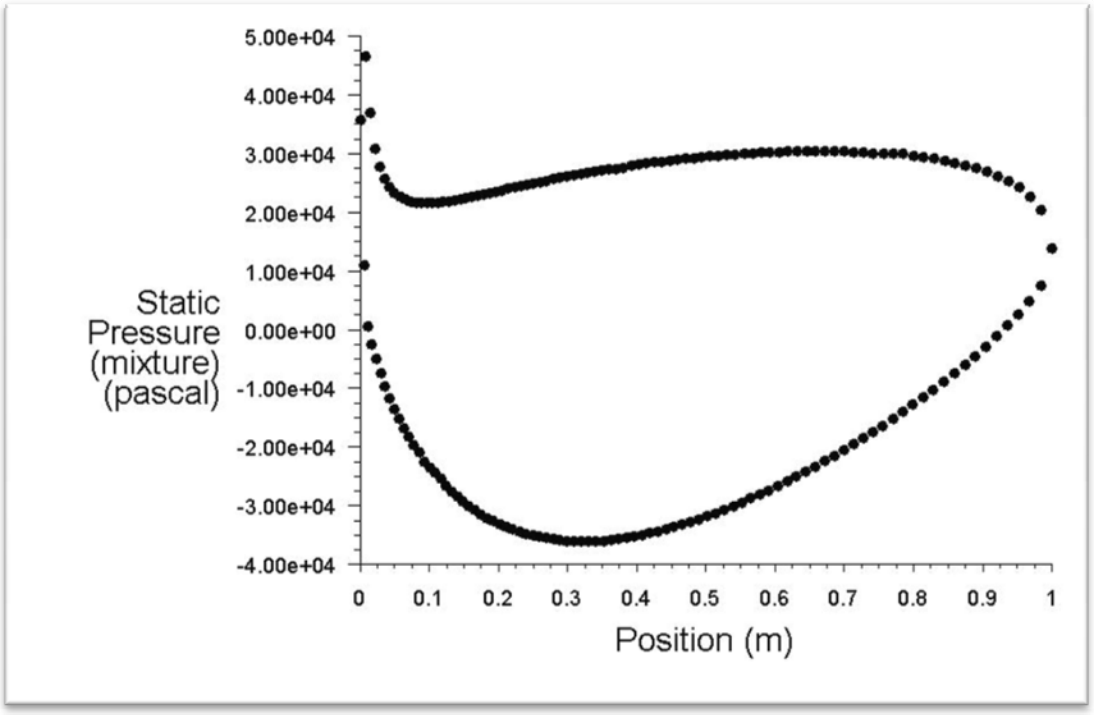

Figure 14(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 2^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

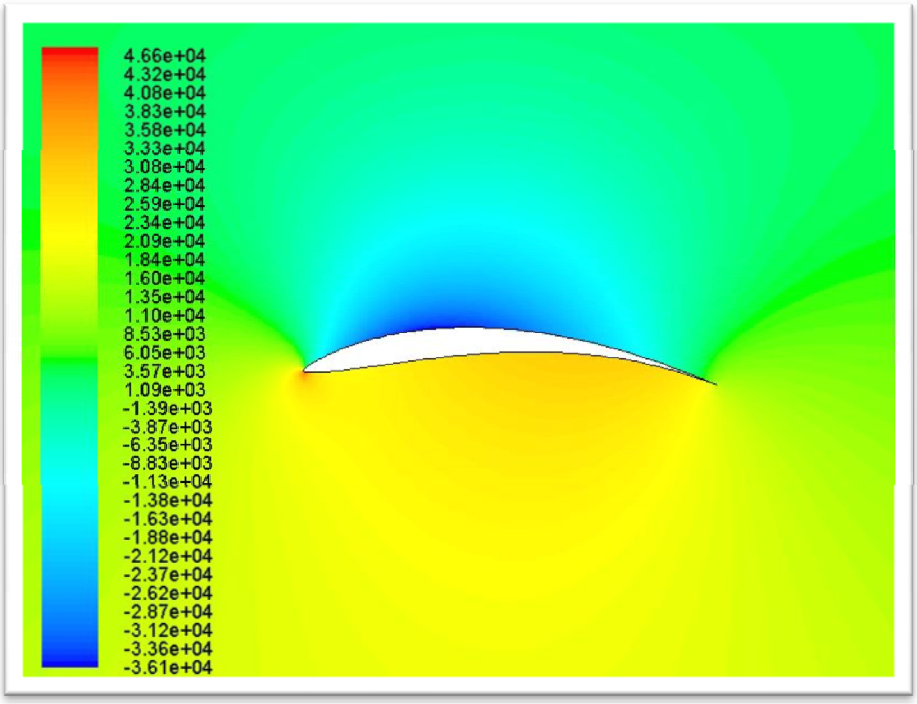

Figure 14(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 2^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

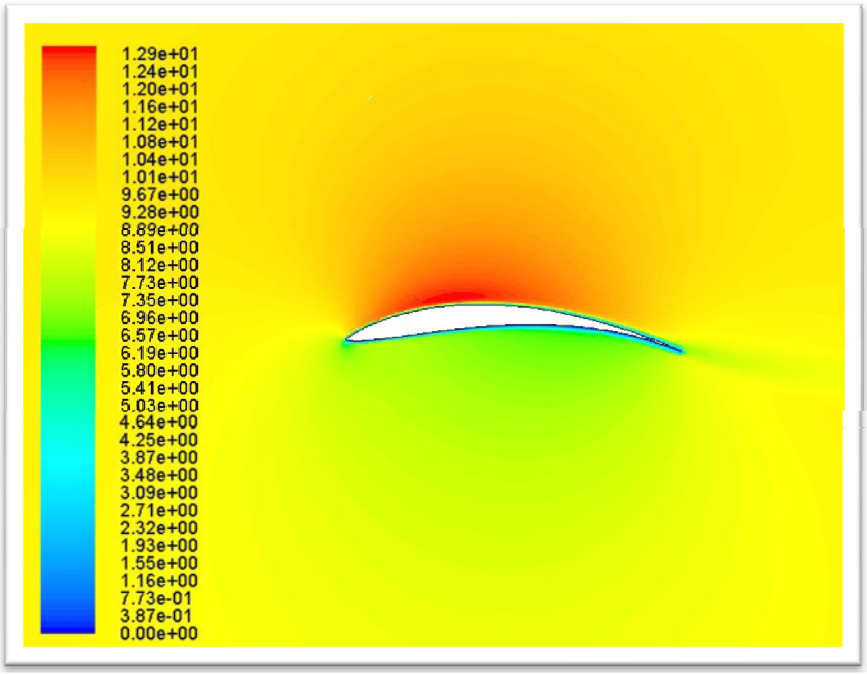

Figure 14(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 2^0$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

4.3 Optimized Hydrofoil Shape at V_∞ = 10 m/s, $\alpha = 4^o$ and $Re_\infty = 10 \times 10^6$, d/c = 0.5

Figures 15(a) – 15(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 45.90.

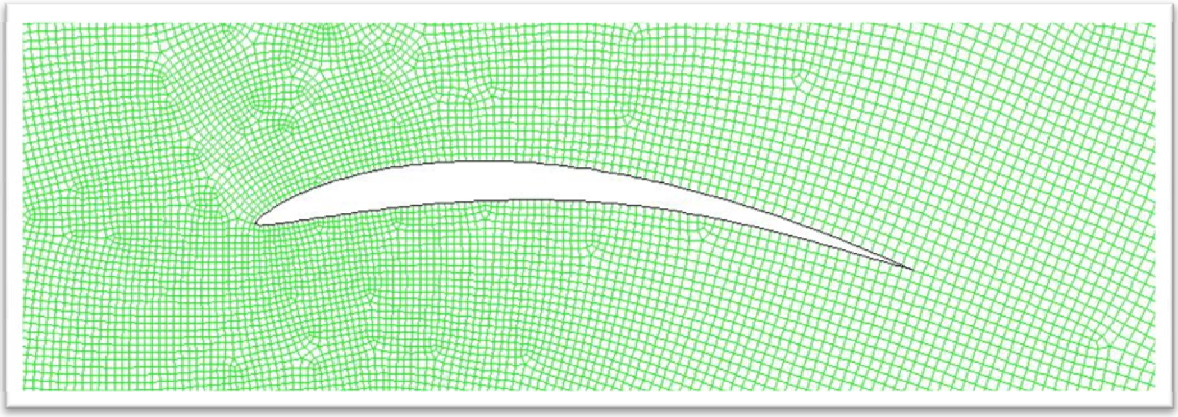

Figure 15(a): Mesh for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 4^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

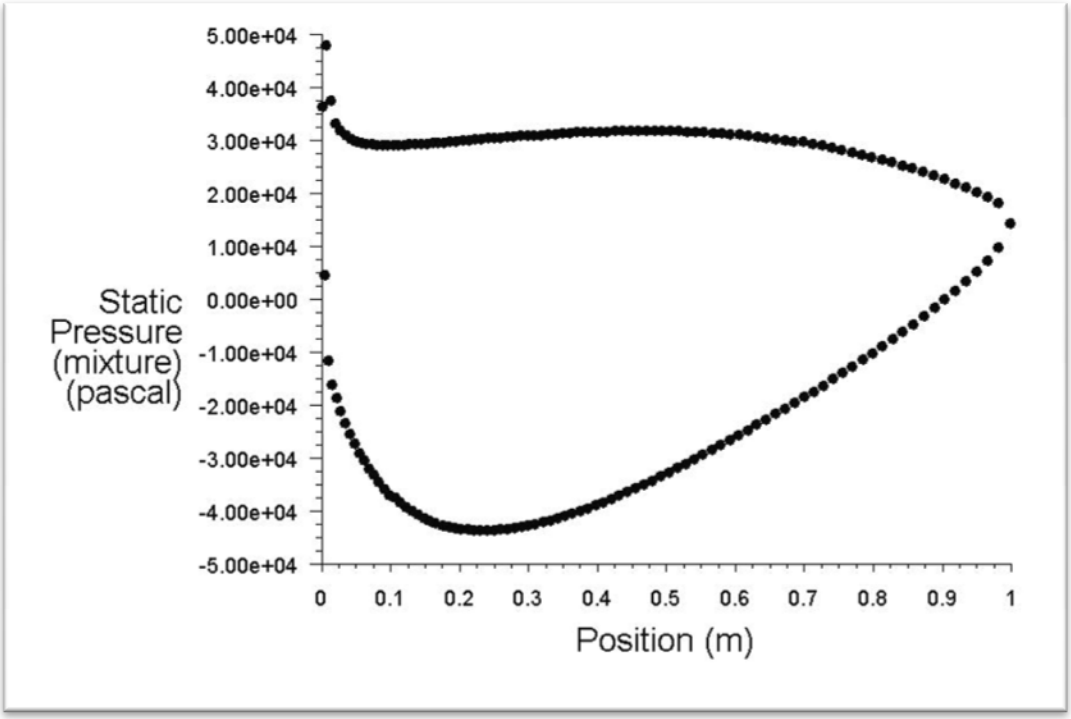

Figure 15(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 4^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

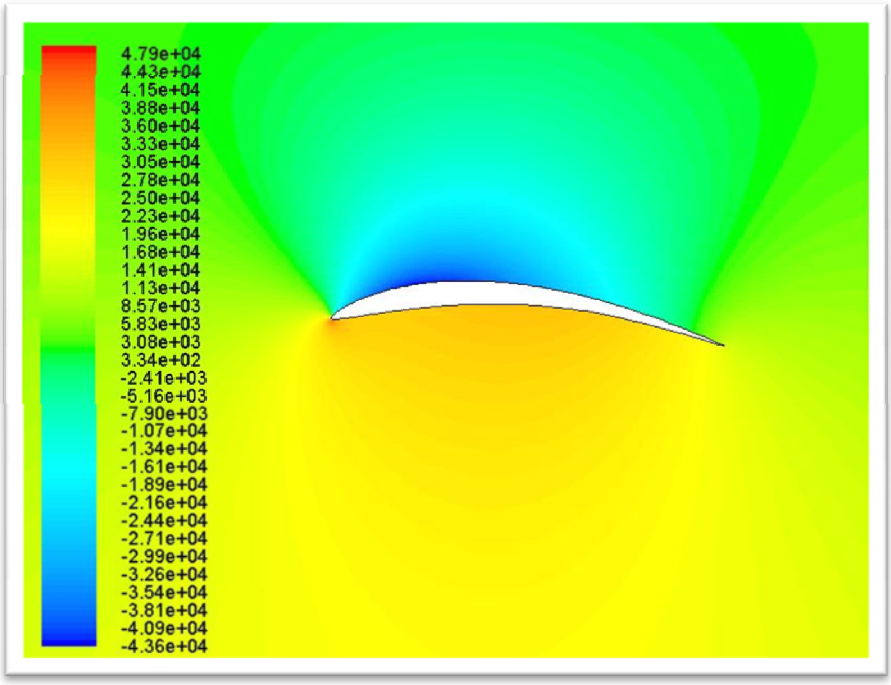

Figure 15(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 4^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

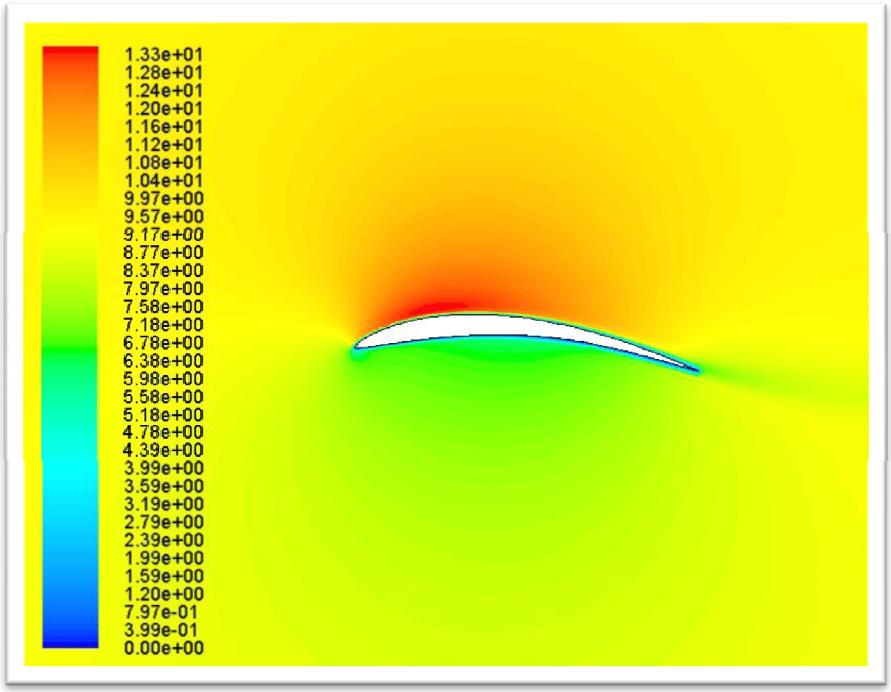

Figure 15(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 4^0$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

4.4 Optimized Hydrofoil Shape at V_∞ = 10 m/s, $\alpha = 6^o$ and $Re_\infty = 10 \times 10^6$, d/c = 0.5

Figures 16(a) – 16(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 40.06.

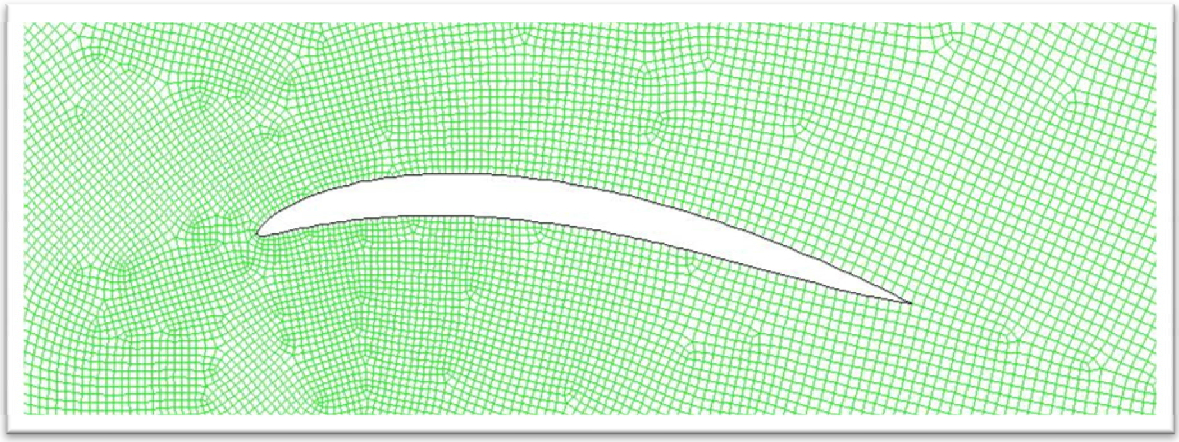

Figure 16(a): Mesh for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 6^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

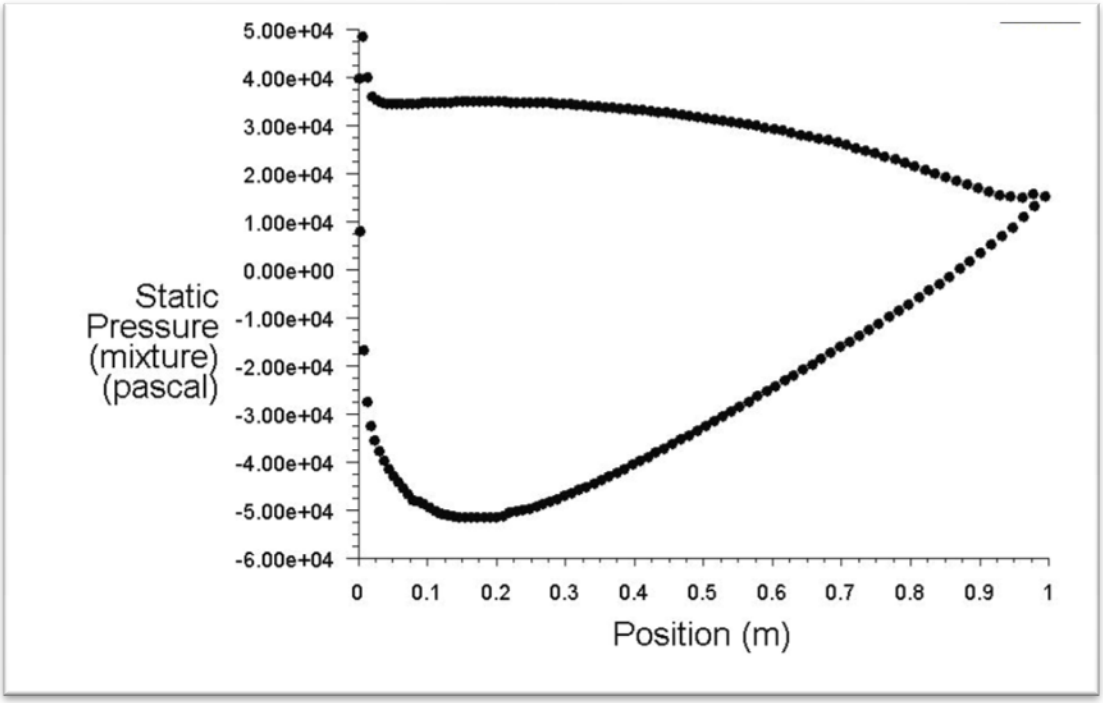

Figure 16(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 6^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

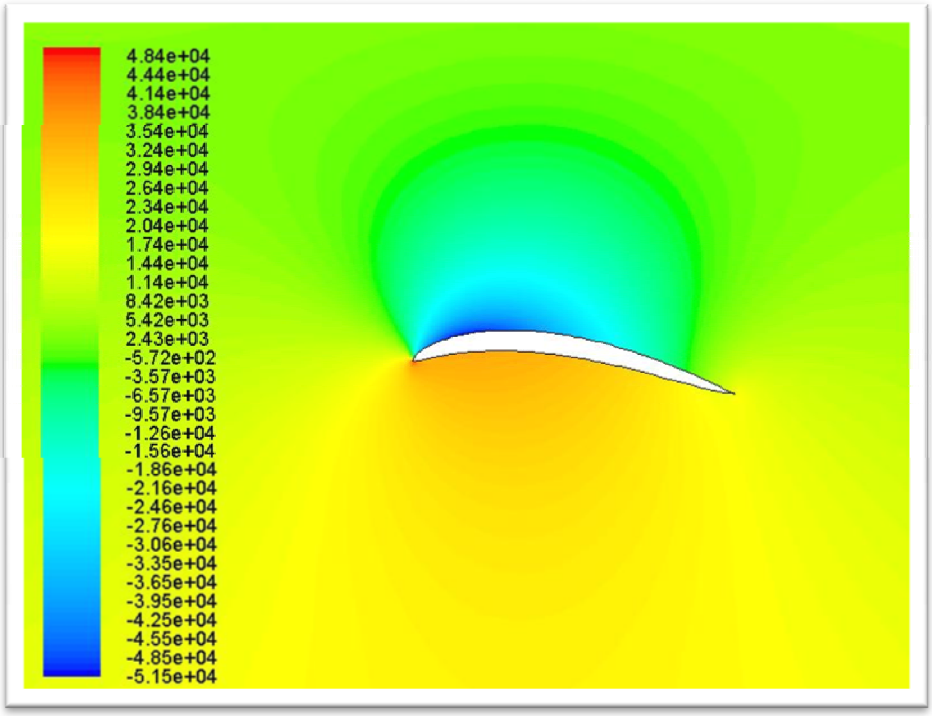

Figure 16(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 6^o$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

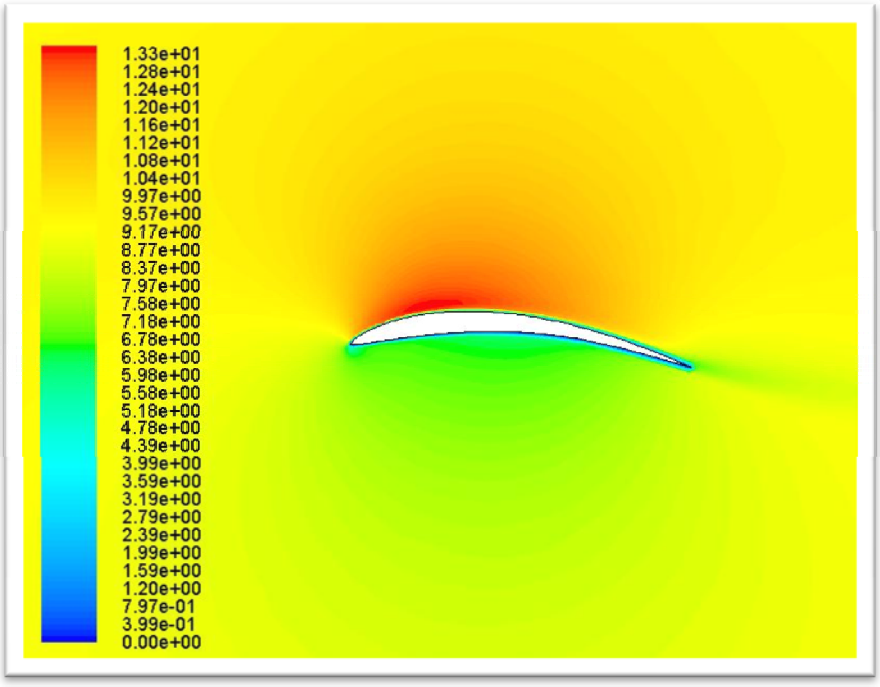

Figure 16(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 10 m/s, $\alpha = 6^0$, $Re_\infty = 10 \times 10^6$, d/c = 0.5

4.5 Optimized Hydrofoil Shape at V_∞ = 20 m/s, $\alpha = 0^o$ and $Re_\infty = 20 \times 10^6$, d/c = 0.5

Figures 17(a) – 17(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 56.31.

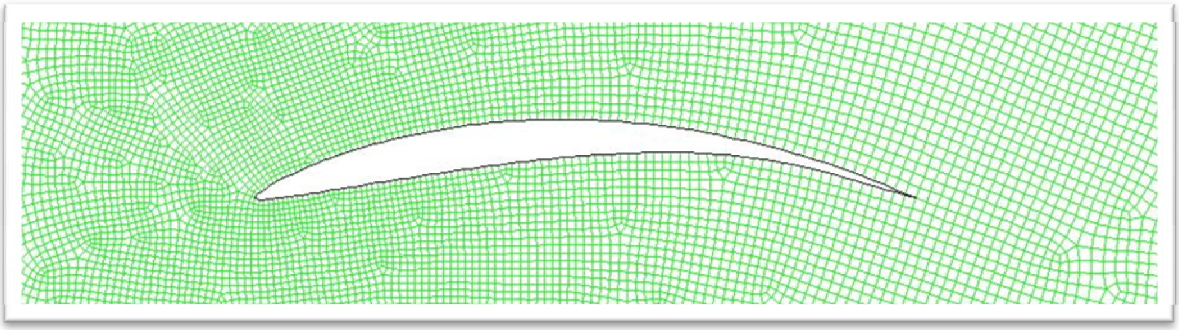

Figure 17(a): Mesh for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 0^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

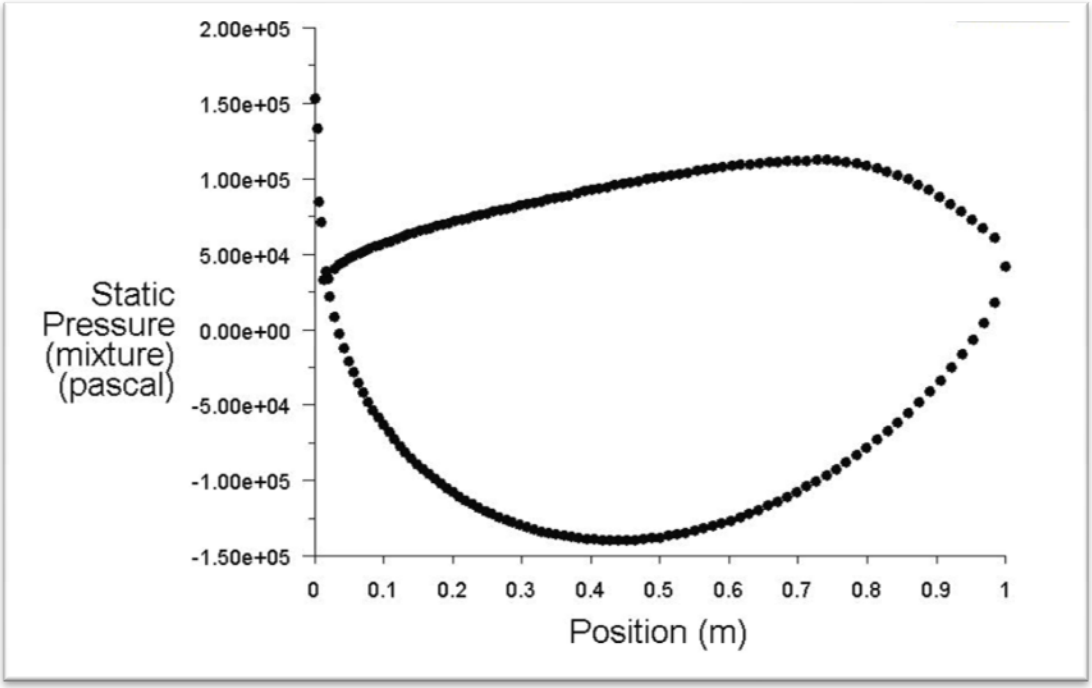

Figure 17(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 0^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

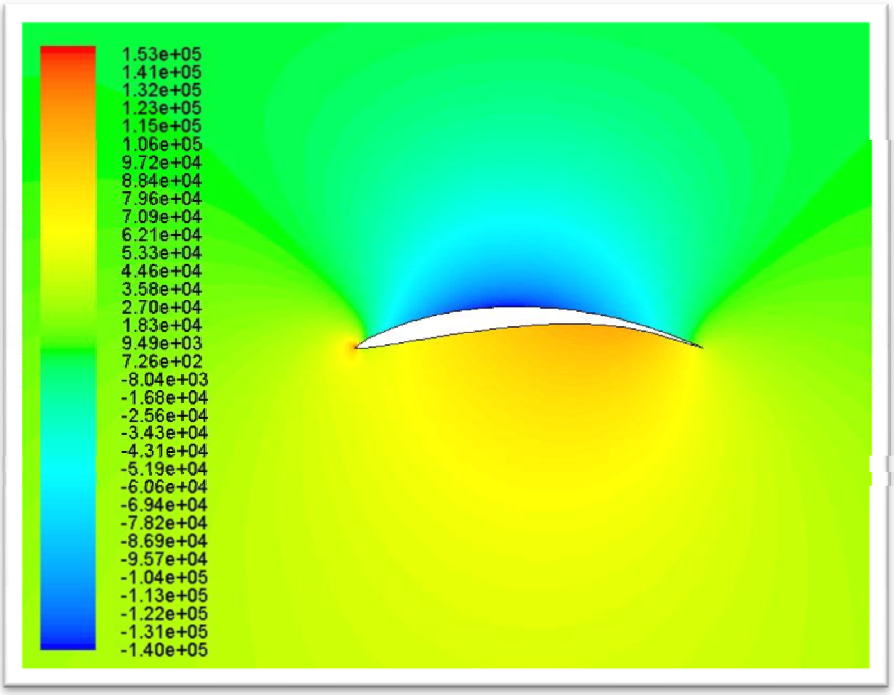

Figure 17(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 0^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

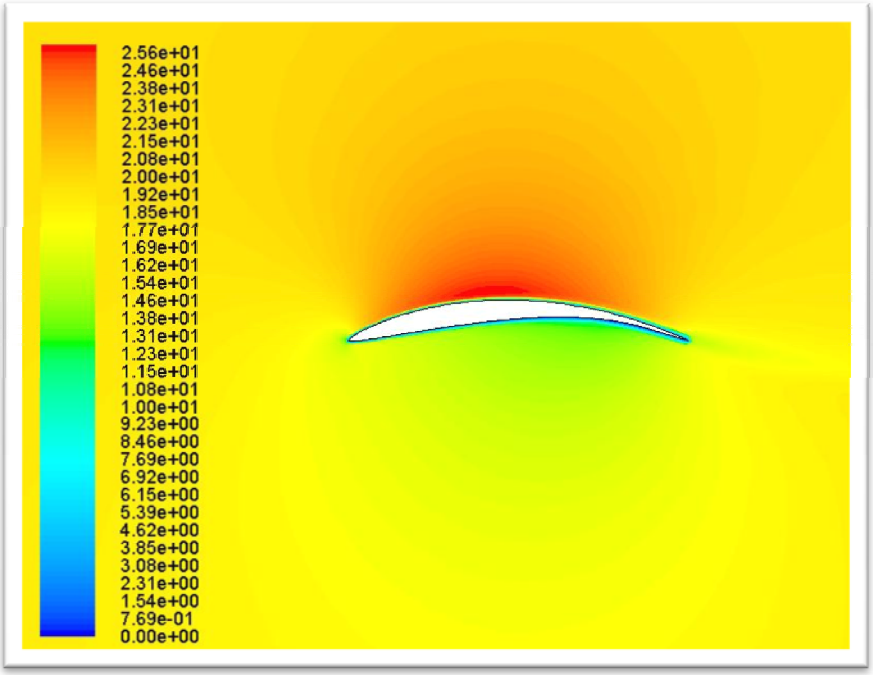

Figure 17(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 0^0$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

4.6 Optimized Hydrofoil Shape at V_∞ = 20 m/s, $\alpha = 2^o$ and $Re_\infty = 20 \times 10^6$, d/c = 0.5

Figures 18(a) – 18(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 54.47.

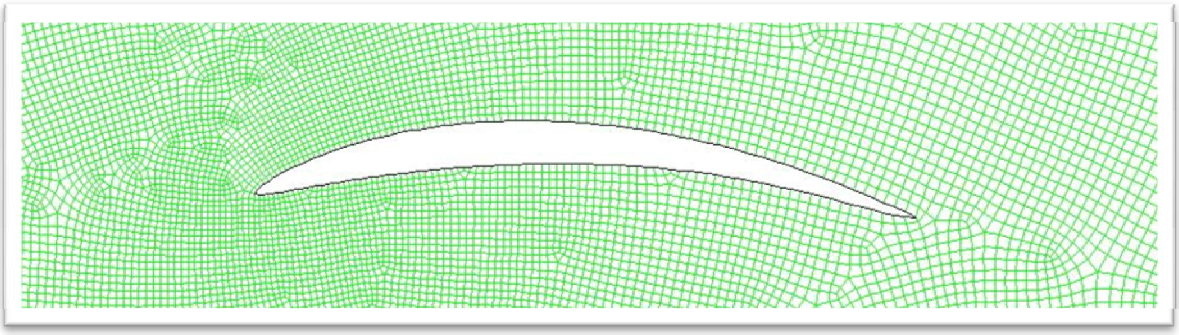

Figure 18(a): Mesh for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 2^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

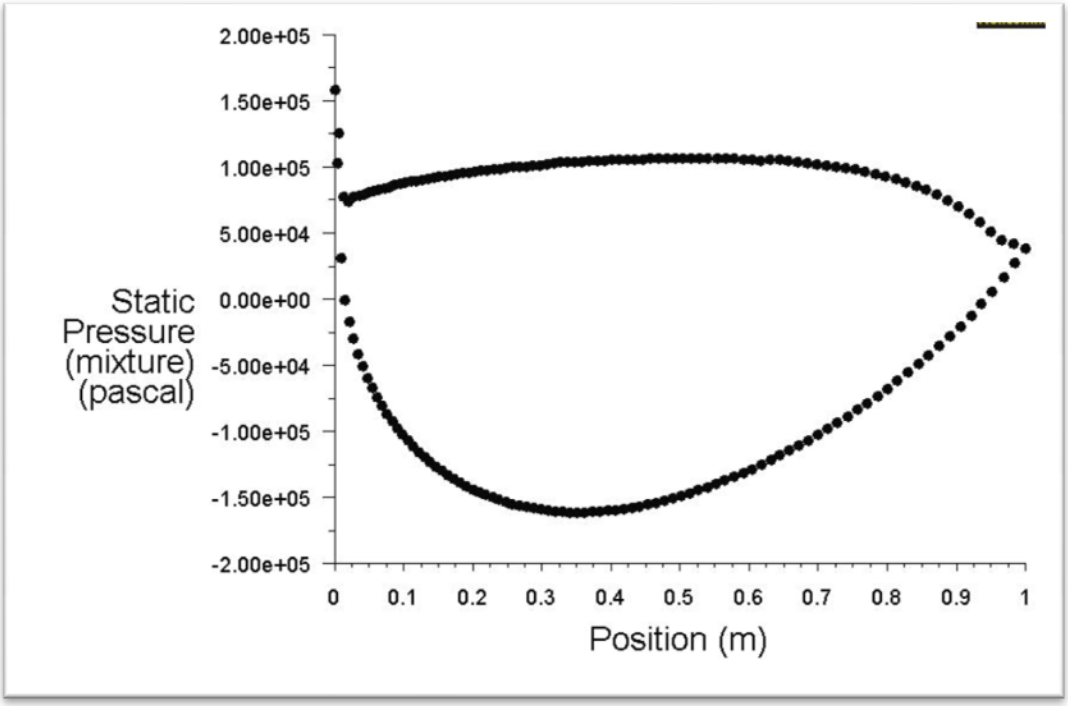

Figure 18(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 2^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

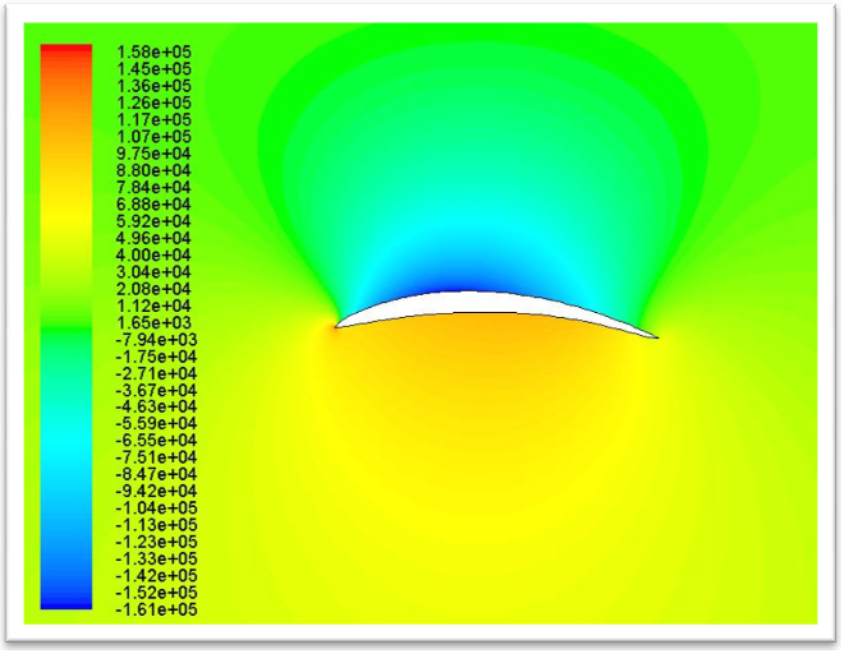

Figure 18(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 2^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

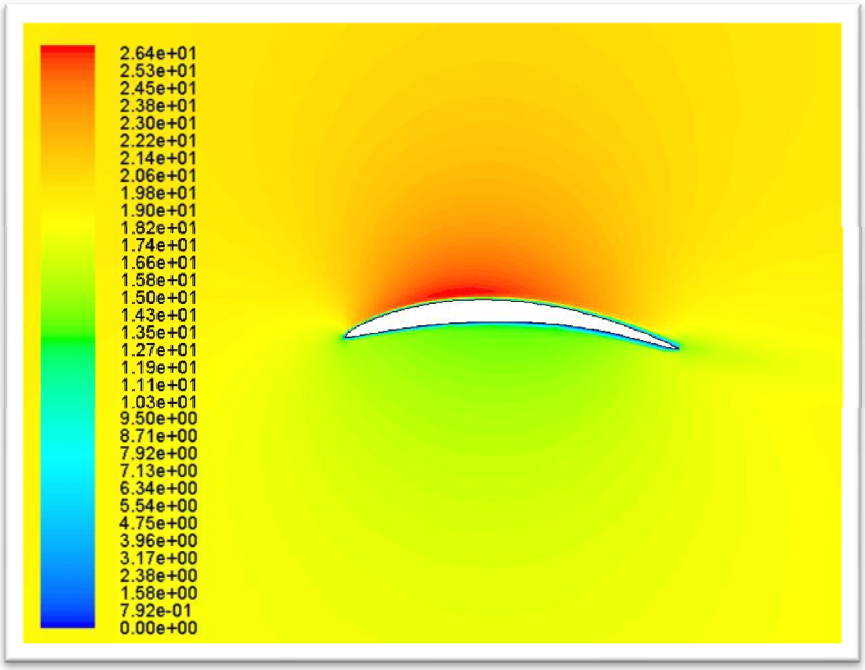

Figure 18(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 2^0$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

4.7 Optimized Hydrofoil Shape at V_∞ = 20 m/s, $\alpha = 4^o$ and $Re_\infty = 20 \times 10^6$, d/c = 0.5

Figures 19(a) – 19(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 48.70.

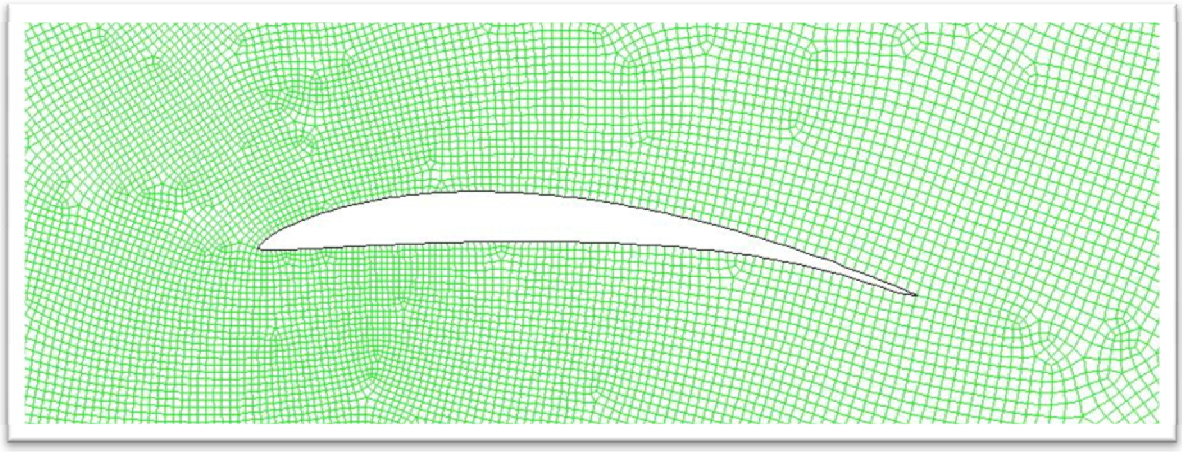

Figure 19(a): Mesh for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 4^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

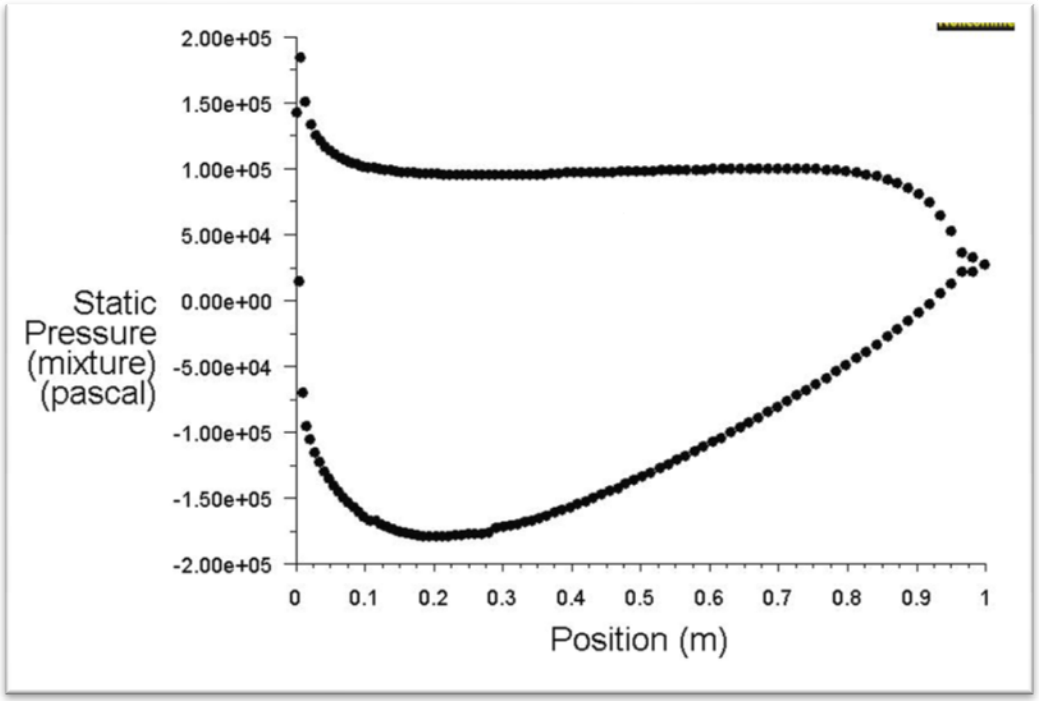

Figure 19(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 4^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

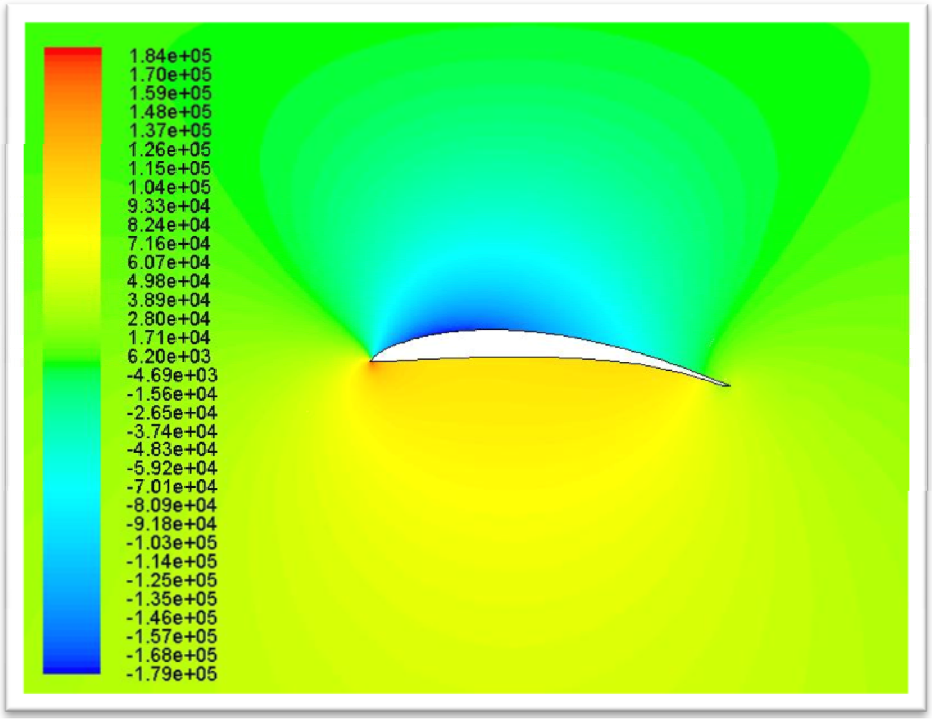

Figure 19(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 4^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

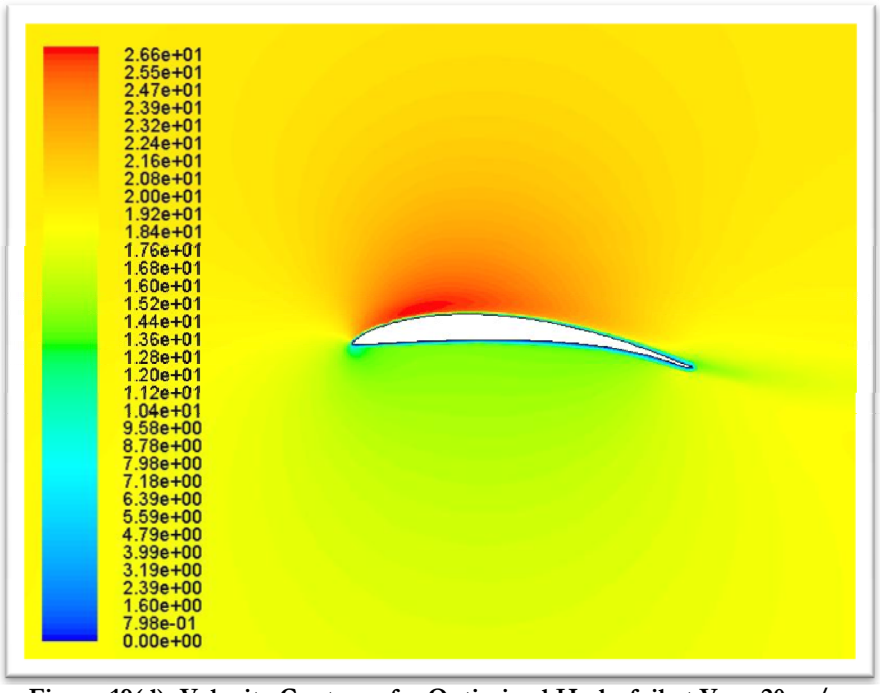

Figure 19(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 4^0$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

4.8 Optimized Hydrofoil Shape at V_∞ = 20 m/s, $\alpha = 6^o$ and $Re_\infty = 20 \times 10^6$, d/c = 0.5

Figures 20(a) – 20(d) show the mesh, the pressure distribution, pressure contours and velocity contours for the optimized hydrofoil respectively. It has a C_l/C_d = 43.72.

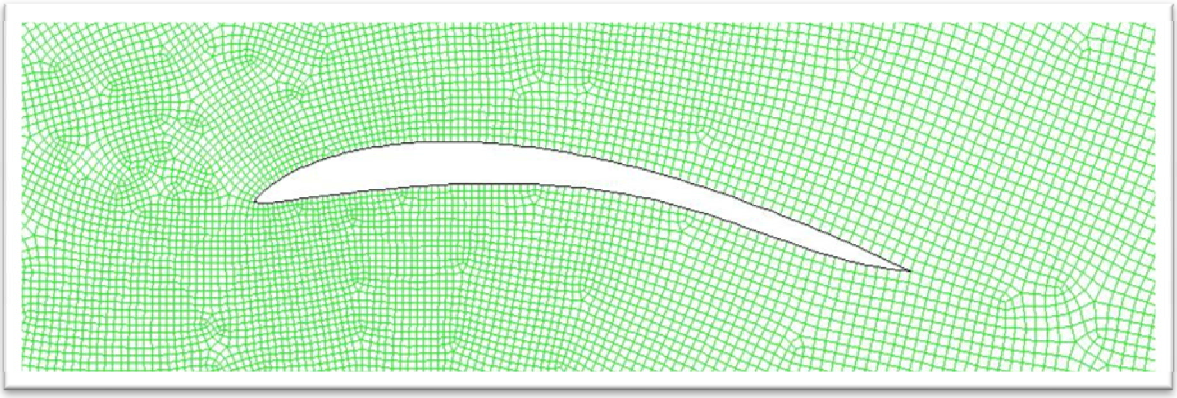

Figure 20(a): Mesh for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 6^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

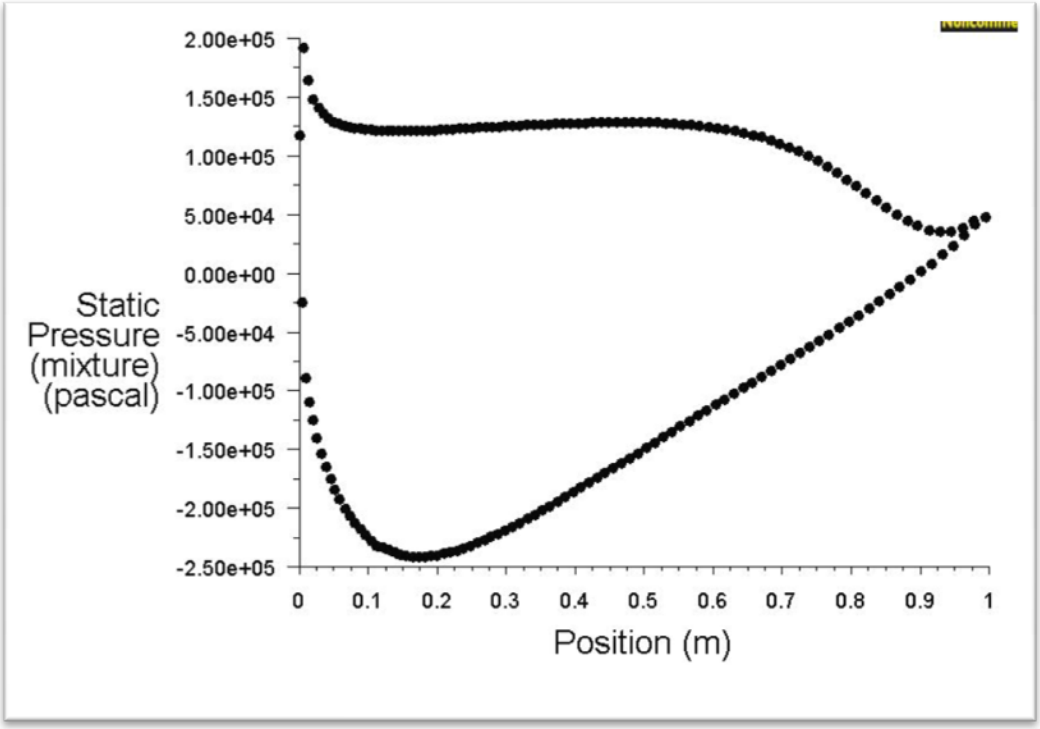

Figure 20(b): Pressure Distribution on the Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 6^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

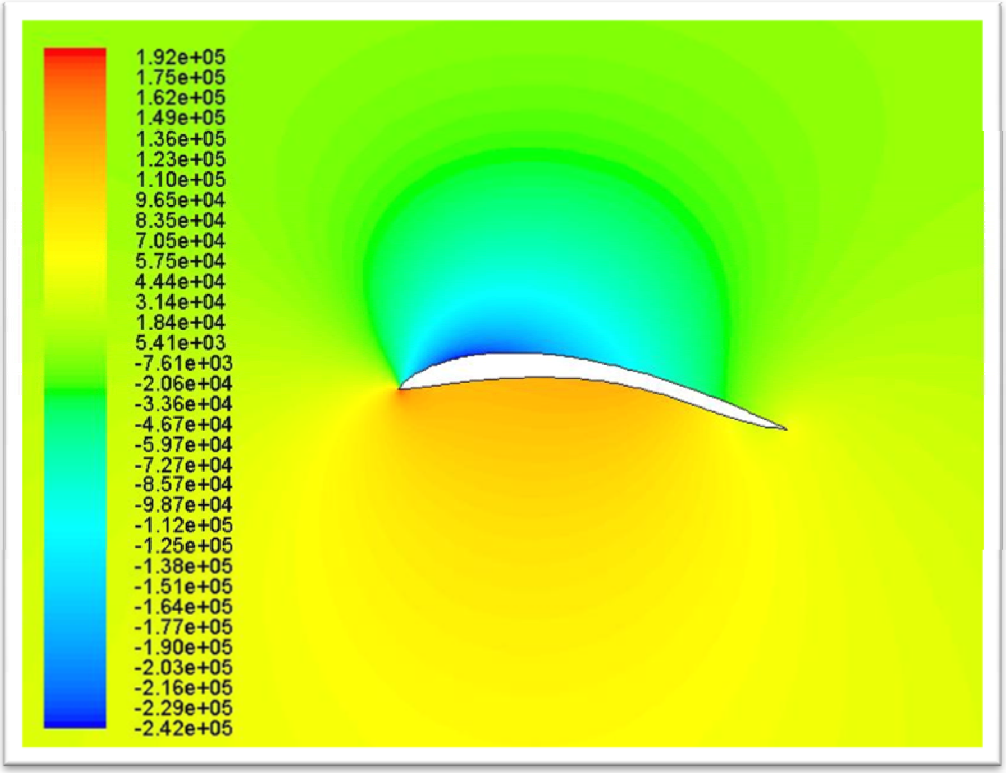

Figure 20(c): Pressure Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 6^o$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

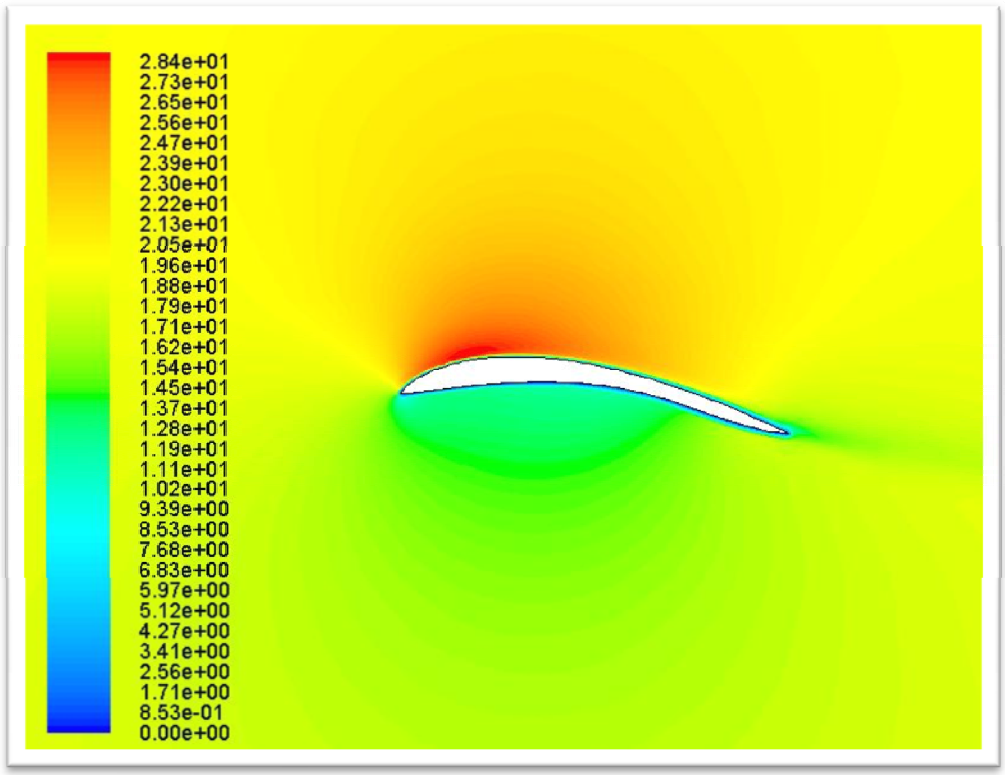

Figure 20(d): Velocity Contours for Optimized Hydrofoil at V_∞ = 20 m/s, $\alpha = 6^0$, $Re_\infty = 20 \times 10^6$, d/c = 0.5

Table 2 gives C_l/C_d values of the baseline hydrofoils and optimized hydrofoils at a distance of 0.5 meters below the free surface (d/c = 0.5) for $Re_\infty = 10 \times 10^6$ and 20×10^6 at angles of attack of $\alpha = 0$, 2, 4 and 6 degree.

Table 2: C_l/C_d for Baseline and Optimized Hydrofoil Shapes Including the Effect of Free Surface (d/c = 0.5)

(Cl/Cd)	Non-Optimized for Free Surface		Optimized for Free Surface	
	Re_∞ = 10 Million	Re_∞ = 20 Million	Re_∞ = 10 Million	Re_∞ = 20 Million
Zero Degrees	42.36	44.81	51.88	56.31
Two Degrees	43.45	47.57	52.03	54.47
Four Degrees	38.11	41.26	45.90	48.70
Six Degrees	30.07	31.69	40.06	43.72

4.9 Qualitative Comparison of Present Solutions with Previous Work for Hydrofoil Under a Free Surface

As shown in Figure 21 from Acosta [1], the influence of the depth of the hydrofoil from the free surface on its lift coefficient has been previously calculated from various Froude number by using the inviscid theory. In these calculations, hydrofoil has been assumed as a flat plate. $C_{l\infty}$ is the lift coefficient of the hydrofoil in deep water. We employed the RANS equations to determine the $C_l/C_{l\infty}$ of the optimized hydrofoil at $\alpha = 0^o$ and $d/c = 0.5$ as a function of Froude number as shown in Figure 22. Although our computations are for turbulent flow and not for flat plate, they show similar trend for the $C_l/C_{l\infty}$ vs F curve as shown in Figure 21. This calculation demonstrates that our computations reproduce the general features of hydrofoil flow fields including the lift and drag coefficient.

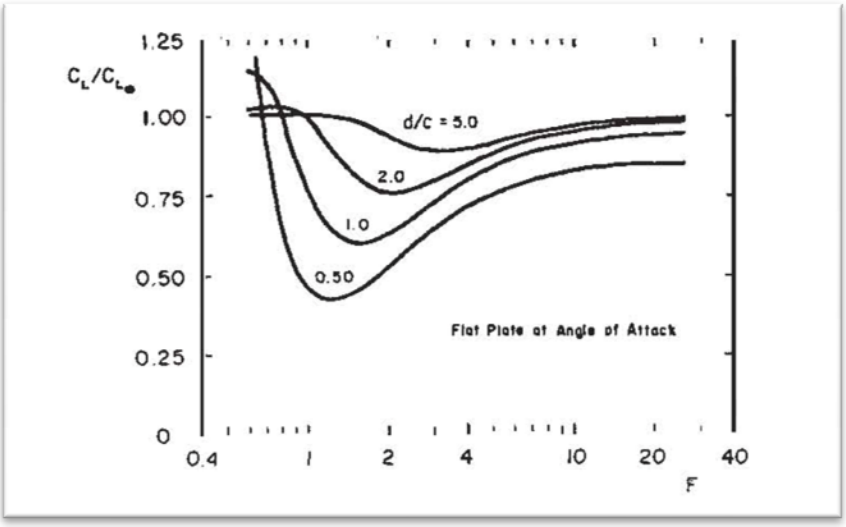

Figure 21: The effect of Froude number based on chord and submergence d beneath the surface on the lift of a fully wetted flat-plat hydrofoil [1]

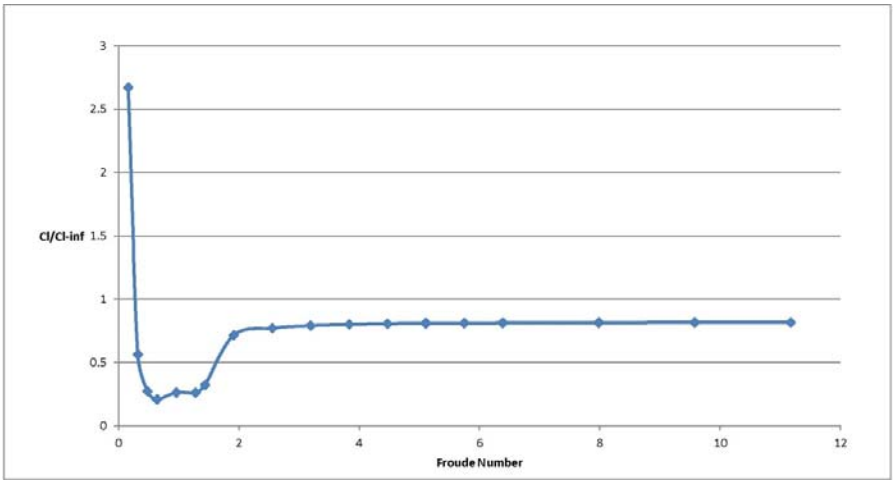

Figure 22: Free surface Lift Coefficient over Deep Water Lift Coefficient vs. Froude Number for d/c = 0.5

Chapter 5

5. Comparison of Optimized Hydrofoil Shapes in Deep Water and Close to Free Surface

In this chapter we compare the optimized hydrofoil shapes obtained by assuming that they are immersed in the deep water (Chapter 3) with those obtained by considering them at a depth d/c = 0.5 (Chapter 4). Eight cases corresponding to two Reynolds numbers of 10 and 20 million and with four angles of attack of 0, 2, 4 and 6 degree are compared. There are minor differences in the shapes as shown below; however they result in significant changes in C_l/C_d as shown in Table 3. In the following sections, for each case, ------- represents the hydrofoil optimized for deep water and ------- represents the hydrofoil optimized for a free surface.

5.1 V_∞ = 10 m/s, $Re_\infty = 10 \times 10^6$, $\alpha = 0^o$

Figure 23: Comparison between the Optimized Hydrofoil Shapes for Deep Water and with Free Surface for V_∞ = 10 m/s, $Re_\infty = 10 \times 10^6$, $\alpha = 0^o$

5.2 V_∞ = 10 m/s, $Re_\infty = 10 \times 10^6$, $\alpha = 2^o$

Figure 24: Comparison between the Optimized Hydrofoil Shapes for Deep Water and with Free Surface for V_∞ = 10 m/s, $Re_\infty = 10 \times 10^6$, $\alpha = 2^o$

5.3 V_∞ = 10 m/s, $Re_\infty = 10 \times 10^6$, $\alpha = 4^o$

Figure 25: Comparison between the Optimized Hydrofoil Shapes for Deep Water and with Free Surface for V_∞ = 10 m/s, $Re_\infty = 10 \times 10^6$, $\alpha = 4^o$

5.4 V_∞ = 10 m/s, $Re_\infty = 10 \times 10^6$, $\alpha = 6^o$

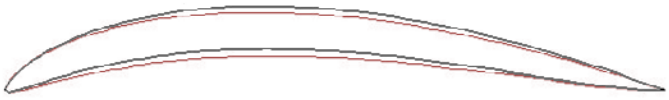

Figure 26: Comparison between the Optimized Hydrofoil Shapes for Deep Water and with Free Surface for V_∞ = 10 m/s, $Re_\infty = 10 \times 10^6$, $\alpha = 6^o$

5.5 V_∞ = 20 m/s, Re_∞ = 20 x 10^6, $\alpha = 0^o$

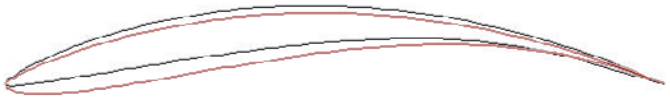

Figure 27: Comparison between the Optimized Hydrofoil Shapes for Deep Water and with Free Surface for V_∞ = 20 m/s, Re_∞ = 20 x 10^6, $\alpha = 0^o$

5.6 V_∞ = 20 m/s, Re_∞ = 20 x 10^6, $\alpha = 2^o$

Figure 28: Comparison between the Optimized Hydrofoil Shapes for Deep Water and with Free Surface for V_∞ = 20 m/s, Re_∞ = 20 x 10^6, $\alpha = 2^o$

5.7 V_∞ = 20 m/s, Re_∞ = 20 x 10^6, $\alpha = 4^o$

Figure 29: Comparison between the Optimized Hydrofoil Shapes for Deep Water and with Free Surface for V_∞ = 20 m/s, Re_∞ = 20 x 10^6, $\alpha = 4^o$

5.8 V_∞ = 20 m/s, $Re_\infty = 20 \times 10^6$, $\alpha = 6^o$

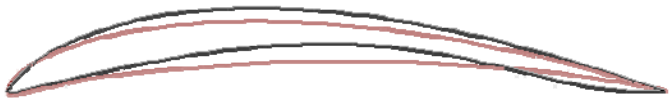

Figure 30: Comparison between the Optimized Hydrofoil Shapes for Deep Water and with Free Surface for V_∞ = 20 m/s, $Re_\infty = 20 \times 10^6$, $\alpha = 6^o$

Table 3: C_l/C_d for Optimized Hydrofoil Shapes in Deep Water and with Free Surface (d/c = 0.5)

(Cl/Cd)	Hydrofoil Optimized for Deep Water		Hydrofoil Optimized for Free Surface	
	10 m/s	20 m/s	10 m/s	20 m/s
Zero Degrees	51.04	54.22	51.88	56.31
Two Degrees	54.96	54.38	52.03	54.47
Four Degrees	45.71	49.92	45.90	48.70
Six Degrees	41.55	42.01	40.06	43.72

Chapter 6

6. Conclusions and Future Work

6.1 Conclusions

By combining the genetic algorithm with the ANSYS CFD solver GAMBIT/FLUENT, optimized shapes for hydrofoils in deep water and near the free surface (air/water interface) were computed. The optimized shapes were obtained by maximizing the lift to drag ratio for a given Reynolds number and angle of attack. For all the cases considered at Re = 10 million and Re = 20 million at angles of attack of 0, 2, 4 and 6 degrees, the optimized hydrofoils have a significantly greater value of C_l/C_d compared to the baseline hydrofoil. There is a slight difference in the optimized shape if the influence of free surface is taken into account, when compared to the shape of the hydrofoil in deep water. C_l/C_d of the optimized hydrofoil with free surface is lower than that for the optimized hydrofoil in deep water. On the whole, the results shown in the thesis clearly demonstrate that GA optimization technique is capable of accurately and efficiently finding the globally optimal hydrofoil.

6.2 Future Work

Since GA is a stochastic optimization tool, its application to shape optimization may not always result in a unique globally optimized hydrofoil. Therefore optimization for given flow conditions should be performed several times by varying different parameters in GA. This aspect of GA based optimization should be further investigated. In addition, various parameters in GA, for example the mutation rate, can influence its convergence. This aspect of GA should further investigated. Additional cases should be computer to evaluate the effect of depth from the free surface on C_l/C_d of the optimized hydrofoils. Finally, the effect of cavitation in the fluid flow model should be included.

References

1. Acosta, A. J. "Hydrofoils and Hydrofoil Craft," *Annual. Rev. of Fluid Mech.*, Vol. 5, 1973, pp. 161-84.

2. Arabshahi, A., Beddhu, M., Briley, W., Chen, J., Gaither, A., Janus, J., Jiang, M., Marcum, D., McGinley, J., Pankajakshan, R., Remotigue, M., Sheng, C., Sreenivas, K., Taylor, L., and Whitfield, D., "A Perspective on Naval Hydrodynamic Flow Simulations," 22nd Symposium on Naval Hydrodynamics, Washington D.C., 2000, pp. 920-932

3. Mulvany, N., Tu, J.Y., Chen, L. and Anderson, B., "Assessment of Two-Equation Turbulence Modeling for High Reynolds Number Hydrofoil Flows," *Int. J. of Num. Meth. in Fluids*, Vol. 45, 2004, pp. 275-299.

4. Goldberg, D. E., *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, 1989.

5. Tozzi, G. G., "Hydrofoil Shape Optimization by Gradient Methods," M.S. Thesis, MIT, 1998.

6. Morgan, B. and Maschmeyer, K., "Improving GA Efficiency with an Artificial Neural Network," Dept. of Computer Science Technical Report, Washington University in St. Louis, 2008.

7. FLUENT 6.3: Flow Modeling Software, Ansys Inc., 2007.

8. GAMBIT 6.2: Geometry and Mesh Generation Preprocessor, Ansys Inc., 2007.

9. Javaherchi, T., "Review of Spalart-Allmaras Turbulence Model and its Modifications," Mechanical Engineering Department, University of Washington, March, 2010.

10. Moscicki, Z. "Optimization of Hydrofoils in Deep Water using a Genetic Algorithm," Department of Mechanical Engineering and Material Science, Washington University in St. Louis, May 2010.

Appendix

Genetic Algorithm Code with Ancillary Files

The following is the complete code for the genetic algorithm used for this thesis, as well as all files called by the program. The section names are the file names for the program.

A.1 gaflatback.java

```
package gaflatback;
import java.io.File;

public class gaflatback {
	private int genSize, numGens, E, existingGenerations;
	private double removePercentage, mutRate;
	public static Airfoil bestAirfoil;
	private generation one;
	double t;
	double minThickness1 = 0.04;
	double maxThickness1 = 0.0625;
	double minThickness2 = 0.058;
	double maxThickness2 = 0.096;
	double minThickness3 = 0.03;
	double maxThickness3 = 0.073;


	public gaflatback(int genSize, int numGens, double
removePercentage, double mutRate){
		this.genSize = genSize;
		this.numGens = numGens;
		this.removePercentage = removePercentage;
		E = (int) Math.round(genSize * this.removePercentage);
		this.mutRate = mutRate;

	}

public Airfoil generateIndividual(){
	boolean x=true;
	Airfoil airfoil = new Airfoil();
	while(x){

	double X1=AirfoilModifier.minX1+Math.random() *
(AirfoilModifier.maxX1-AirfoilModifier.minX1);
	double X2=AirfoilModifier.minX2+Math.random() *
(AirfoilModifier.maxX2-AirfoilModifier.minX2);
	double Y1=AirfoilModifier.minY1+Math.random() *
(AirfoilModifier.maxY1-AirfoilModifier.minY1);
	double Y2=AirfoilModifier.minY2+Math.random() *
(AirfoilModifier.maxY2-AirfoilModifier.minY2);
```

```
      double M1=AirfoilModifier.minM1+Math.random() *
(AirfoilModifier.maxM1-AirfoilModifier.minM1);
      double M2=AirfoilModifier.minM2+Math.random() *
(AirfoilModifier.maxM2-AirfoilModifier.minM2);
      double M3=AirfoilModifier.minM3+Math.random() *
(AirfoilModifier.maxM3-AirfoilModifier.minN3);
//    double M4=AirfoilModifier.minM4+Math.random() *
(AirfoilModifier.maxM4-AirfoilModifier.minN4);

      double N1=AirfoilModifier.minN1+Math.random() *
(AirfoilModifier.maxN1-AirfoilModifier.minN1);
      double N2=AirfoilModifier.minN2+Math.random() *
(AirfoilModifier.maxN2-AirfoilModifier.minN2);
      double N3=AirfoilModifier.minN3+Math.random() *
(AirfoilModifier.maxN3-AirfoilModifier.minN3);
//    double N4=AirfoilModifier.minN4+Math.random() *
(AirfoilModifier.maxN4-AirfoilModifier.minN4);

      Airfoil airfoilgen = new Airfoil(X1, X2, Y1, Y2, M1, M2, M3, N1,
N2, N3 );
      AirfoilModifier.modAirfoil(airfoilgen);
      gambitAirfoils gt = new gambitAirfoils(airfoilgen);
      gt.buildAirfoil(0.02);

      if(gt.getThickness(0.02,0,12)<=(double)maxThickness1&&gt.getThick
ness(0.02,0,12)>=(double)minThickness1

      &&gt.getThickness(0.02,13,35)<=(double)maxThickness2&&gt.getThick
ness(0.02,13,35)>=(double)minThickness2

      &&gt.getThickness(0.02,36,50)<=(double)maxThickness3&&gt.getThick
ness(0.02,36,50)>=(double)minThickness3

      &&gt.getminThickness(0.02)>=0&&gt.getminThicknessInterval(0.02,8,
43)>=0.02){
            x=false;
            airfoil = airfoilgen;
      }else
            x=true;
      }
      return airfoil;
   }

public void runOptimization(){

      one = manyIndividuals();
      //one = existingIndividuals(134);
      bestAirfoil = new Airfoil();

      for (int i=0; i<numGens; i++){
            one.determineFitness(i);

            /**Find airfoil with highest coefficient of lift */
            System.out.println("***** Generation " + i + "*****");
            bestAirfoil = one.getBestAirfoil();
            System.out.println("Best airfoil Cl/Cd = " +
bestAirfoil.getFitness());
```

```
            /**Create new generations to find best airfoil*/
            advanceGen();
      }

}

public Airfoil reorganize(double X1, double Y1, double X2, double
Y2,double M1,double N1,double M2,double N2,double M3,double N3){
      return new Airfoil(X1,X2,Y1,Y2,M1,M2,M3,N1,N2,N3);
}

private generation manyIndividuals(){
      //new generation of airfoils
      generation airfoils = new generation(genSize);

      //put the initial foil in
      Airfoil A1 = new
Airfoil(reorganize(0.21248106035336295,0.060796393344218304,0.509840684
8141193,0.19531000819880937,0.02,-
0.03,0.48138507937291936,0.1049768428670589,0.6822165543341977,0.067960
29230541498));
      airfoils.addAirfoil(A1);
      //loop to generate 10 airfoils

      for(int i=0;i < genSize-1;i++){
            airfoils.addAirfoil(generateIndividual());
      }

      return airfoils;
}

/***** get the index of Parents according to the weights of the
airfoils***/
public int selectIndexofParents(){
      double sum = 0;
      double ratio[] = new double[genSize];
      double dice = Math.random();
      double area = 0;
      boolean whicharea = true;
      int indexofParents = 0;
      //System.out.println(" dice = " + dice);

      BubbleSorter.sort(one);   /*** sort the airfoils according to
fitness from lowest to highest***/
      /*** get the sum of the positive fitness***/
      for (int i=0; i<one.getAirfoilVectorSize(); i++ ){
            if(one.airfoils.elementAt(i).getFitness()>0){
                  sum = sum + one.airfoils.elementAt(i).getFitness();

            }
      //System.out.println(" fitness sum = " + sum);
      }

      /*** get the ratio of each fitness ***/
      for (int r=0 ; r< /**genSize**/one.getAirfoilVectorSize(); r++){
            if(one.airfoils.elementAt(r).getFitness()>0){
```

```
                  ratio[r] =
one.airfoils.elementAt(r).getFitness()/sum;
                  //System.out.println(" ratio = " + ratio[r]);
            }
      }

      /*** select the index if dice fits some special area ***/
      int currentindex = 0;
      for (int i=0; i<one.getAirfoilVectorSize(); i++){

            //System.out.println(" current index = " + i);
            if(area <= dice){
                  if (one.airfoils.elementAt(i).getFitness()>0){
                        area = area + ratio[i];
                        currentindex = i;
                        //System.out.println("area = " + area);
                  }
            }else if (area > dice&& area <10000){
                  indexofParents = currentindex;
                  area = 10000;
            }
      //    System.out.println(" current index = " + i);

            }

      //System.out.println(" indexofParents = " + indexofParents);
      return indexofParents;
      }


public void advanceGen(){
      System.out.println("Begin advance
gennnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn");
      generation nextGen = new generation(genSize);
      generation GenforFluent = new generation();
      generation GennoFluent = new generation();
      int count = 0;
      /**Create E number of new airfoils by using coordinates from two
random airfoils (with weights)from the previous generation*/
      System.out.println("Begin
crossoverrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr");

      /**Remove E number of airfoils from the previous generation with
the highest cl*/
      //naturalSelection();

      while(count<E){
            boolean x=true;
            while(x){

            Airfoil airfoil1 = one.getAirfoil(selectIndexofParents());
            Airfoil airfoil2 = one.getAirfoil(selectIndexofParents());

            /**
            Airfoil airfoil1 =
one.getAirfoil((int)Math.floor(Math.random()*one.getAirfoilVectorSize()
));
```

```
            Airfoil airfoil2 =
one.getAirfoil((int)Math.floor(Math.random()*one.getAirfoilVectorSize()
));
            **/


            if(airfoil1 != airfoil2){
                  Airfoil
airfoilCrossover=crossover(airfoil1,airfoil2);
                  gambitAirfoils gt = new
gambitAirfoils(airfoilCrossover);
                  gt.buildAirfoil(0.02);

      if(gt.getThickness(0.02,0,12)<=(double)maxThickness1&&gt.getThick
ness(0.02,0,12)>=(double)minThickness1

      &&gt.getThickness(0.02,13,35)<=(double)maxThickness2&&gt.getThick
ness(0.02,13,35)>=(double)minThickness2

      &&gt.getThickness(0.02,36,50)<=(double)maxThickness3&&gt.getThick
ness(0.02,36,50)>=(double)minThickness3

      &&gt.getminThickness(0.02)>=0&&gt.getminThicknessInterval(0.02,8,
43)>=0.02){
                        x=false;
                        //nextGen.addAirfoil(airfoilCrossover);
                        GenforFluent.addAirfoil(airfoilCrossover);

      //System.out.println("maxthickness2="+gt.getmaxThickness(0.02));
                        count++;
                  }else
                        x=true;
                  }
            }
      }

      /**Check to make sure it works*/
      System.out.print("***Next generation of airfoils go to Fluent***"
+ "\n");
      GenforFluent.outputAirfoils();

      /**Remove E number of airfoils from the previous generation with
the lowest cl*/
      naturalSelection();


      /**Add surviving airfoils to nextGen of airfoils*/
      for(int i=0;i<one.getAirfoilVectorSize();i++){
            //nextGen.addAirfoil(one.getAirfoil(i));
            GennoFluent.addAirfoil(one.getAirfoil(i));
      }

      /**Check to make sure it works*/
      System.out.println("***Next generation of airfoils without
Fluent***" + "\n");
      GennoFluent.outputAirfoils();
```

```
		/**Mutate*/
		mutate(GenforFluent,GennoFluent,nextGen);
		System.out.println("***testinggggggggggggggggggggggggggggggggggggggggggggggggg" + "\n");
		/**Original generation becomes nextGen*/

		one = nextGen;

	}

	/**Take coordinates of two airfoils and combine them to get a new airfoil*/
	private Airfoil crossover(Airfoil airfoil1, Airfoil airfoil2){


		//double x0_1,x0_2;
		double x1_1,x1_2;
		double y1_1,y1_2;
		double x2_1,x2_2;
		double y2_1,y2_2;
		double m1_1,m1_2;
		double n1_1,n1_2;
		double m2_1,m2_2;
		double n2_1,n2_2;
		double m3_1,m3_2;
		double n3_1,n3_2;


		if(airfoil1.getFitness() > airfoil2.getFitness()){
			x1_2 = airfoil1.X1;
			y1_2 = airfoil1.Y1;
			x2_2 = airfoil1.X2;
			y2_2 = airfoil1.Y2;
			m1_2 = airfoil1.M1;
			n1_2 = airfoil1.N1;
			m2_2 = airfoil1.M2;
			n2_2 = airfoil1.N2;
			m3_2 = airfoil1.M3;
			n3_2 = airfoil1.N3;

			x1_1 = airfoil2.X1;
			y1_1 = airfoil2.Y1;
			x2_1 = airfoil2.X2;
			y2_1 = airfoil2.Y2;
			m1_1 = airfoil2.M1;
			n1_1 = airfoil2.N1;
			m2_1 = airfoil2.M2;
			n2_1 = airfoil2.N2;
			m3_1 = airfoil2.M3;
			n3_1 = airfoil2.N3;

		} else {
			x1_1 = airfoil1.X1;
			y1_1 = airfoil1.Y1;
			x2_1 = airfoil1.X2;
			y2_1 = airfoil1.Y2;
			m1_1 = airfoil1.M1;
```

```
            n1_1 = airfoil1.N1;
            m2_1 = airfoil1.M2;
            n2_1 = airfoil1.N2;
            m3_1 = airfoil1.M3;
            n3_1 = airfoil1.N3;

            x1_2 = airfoil2.X1;
            y1_2 = airfoil2.Y1;
            x2_2 = airfoil2.X2;
            y2_2 = airfoil2.Y2;
            m1_2 = airfoil2.M1;
            n1_2 = airfoil2.N1;
            m2_2 = airfoil2.M2;
            n2_2 = airfoil2.N2;
            m3_2 = airfoil2.M3;
            n3_2 = airfoil2.N3;
      }

      /**Create new airfoils coordinates with crossover of two old
airfoils biasing towards airfoil with smaller Cl*/
      //double x0 = Math.random()*(x0_2-x0_1) + x0_2;
      double x1 = Math.random()*(x1_2-x1_1) + x1_2;
      double y1 = Math.random()*(y1_2-y1_1) + y1_2;
      double x2 = Math.random()*(x2_2-x2_1) + x2_2;
      double y2 = Math.random()*(y2_2-y2_1) + y2_2;
      double m1 = Math.random()*(m1_2-m1_1) + m1_2;
      double n1 = Math.random()*(n1_2-n1_1) + n1_2;
      double m2 = Math.random()*(m2_2-m2_1) + m2_2;
      double n2 = Math.random()*(n2_2-n2_1) + n2_2;
      double m3 = Math.random()*(m3_2-m3_1) + m3_2;
      double n3 = Math.random()*(n3_2-n3_1) + n3_2;
//    double m4 = Math.random()*(m4_2-m4_1) + m4_2;
//    double n4 = Math.random()*(n4_2-n4_1) + n4_2;

      /**
      Airfoil airfoil = new Airfoil(x1,y1,x2,y2,m1,n1,m2,n2,m3,n3);
      **/
      Airfoil airfoil = new Airfoil(x1,x2,y1,y2,m1,m2,m3,n1,n2,n3);
      AirfoilModifier.modAirfoil(airfoil);

      return airfoil;


}

private void naturalSelection(){
      System.out.println("Begin natural selection");
      BubbleSorter.sort(one);
      one.removeAirfoils(E);
}

private void mutate(generation GenforFluent,generation
GennoFluent,generation nextGen){
      System.out.println("**********************Begin
mutation*********************************");
      System.out.println(GenforFluent.getAirfoilVectorSize());
      int counter = 0;
```

```
      int counter2 = 0;
      for(int i=0;i<GenforFluent.getAirfoilVectorSize();i++){

            if(Math.random() <= mutRate) {

                  boolean x=true;
                  Airfoil airfoiladd = new Airfoil();
                  Airfoil airfoilremove = new Airfoil();

                  while(x){
                  Airfoil airfoil = GenforFluent.getAirfoil(i);

                  double x1=AirfoilModifier.minX1+Math.random() *
(AirfoilModifier.maxX1-AirfoilModifier.minX1);
                  double x2=AirfoilModifier.minX2+Math.random() *
(AirfoilModifier.maxX2-AirfoilModifier.minX2);
                  double y1=AirfoilModifier.minY1+Math.random() *
(AirfoilModifier.maxY1-AirfoilModifier.minY1);
                  double y2=AirfoilModifier.minY2+Math.random() *
(AirfoilModifier.maxY2-AirfoilModifier.minY2);

                  double m1=AirfoilModifier.minM1+Math.random() *
(AirfoilModifier.maxM1-AirfoilModifier.minM1);
                  double m2=AirfoilModifier.minM2+Math.random() *
(AirfoilModifier.maxM2-AirfoilModifier.minM2);
                  double m3=AirfoilModifier.minM3+Math.random() *
(AirfoilModifier.maxM3-AirfoilModifier.minM3);
//                double m4=AirfoilModifier.minM4+Math.random() *
(AirfoilModifier.maxM4-AirfoilModifier.minM4);
                  double n1=AirfoilModifier.minN1+Math.random() *
(AirfoilModifier.maxN1-AirfoilModifier.minN1);
                  double n2=AirfoilModifier.minN2+Math.random() *
(AirfoilModifier.maxN2-AirfoilModifier.minN2);
                  double n3=AirfoilModifier.minN3+Math.random() *
(AirfoilModifier.maxN3-AirfoilModifier.minN3);
//                double n4=AirfoilModifier.minN4+Math.random() *
(AirfoilModifier.maxN4-AirfoilModifier.minN4);

                  Airfoil mutant = new
Airfoil(x1,x2,y1,y2,m1,m2,m3,n1,n2,n3);
                  AirfoilModifier.modAirfoil(mutant);
                  gambitAirfoils gt = new gambitAirfoils(mutant);
                  gt.buildAirfoil(0.02);



      if(gt.getThickness(0.02,0,12)<=(double)maxThickness1&&gt.getThick
ness(0.02,0,12)>=(double)minThickness1

      &&gt.getThickness(0.02,13,35)<=(double)maxThickness2&&gt.getThick
ness(0.02,13,35)>=(double)minThickness2

      &&gt.getThickness(0.02,36,50)<=(double)maxThickness3&&gt.getThick
ness(0.02,36,50)>=(double)minThickness3

      &&gt.getminThickness(0.02)>=0&&gt.getminThicknessInterval(0.02,8,
43)>=0.02){
```

```
                        x=false;
                        airfoiladd = mutant;
                        GenforFluent.removeAirfoilAt(counter);
                        GenforFluent.addAirfoilAt(airfoiladd, counter);
                        counter ++;

      //System.out.println("maxthickness3="+gt.getmaxThickness(0.02));
                  }else
                        x=true;
                  }
                  //GenforFluent.removeAirfoil(airfoilremove);
                  System.out.println("Airfoil removed for mutation");
                  //GenforFluent.addAirfoilAt(airfoiladd);
                  System.out.println("Airfoil added for mutation");
            }
      }


      for(int j=0;j<GennoFluent.getAirfoilVectorSize();j++){

            if(Math.random() <= mutRate) {
                  boolean x=true;
                  Airfoil airfoiladd = new Airfoil();
                  Airfoil airfoilremove = new Airfoil();

                  while(x){
                  Airfoil airfoil = GennoFluent.getAirfoil(j);

                  double x1=AirfoilModifier.minX1+Math.random() *
(AirfoilModifier.maxX1-AirfoilModifier.minX1);
                  double x2=AirfoilModifier.minX2+Math.random() *
(AirfoilModifier.maxX2-AirfoilModifier.minX2);
                  double y1=AirfoilModifier.minY1+Math.random() *
(AirfoilModifier.maxY1-AirfoilModifier.minY1);
                  double y2=AirfoilModifier.minY2+Math.random() *
(AirfoilModifier.maxY2-AirfoilModifier.minY2);

                  double m1=AirfoilModifier.minM1+Math.random() *
(AirfoilModifier.maxM1-AirfoilModifier.minM1);
                  double m2=AirfoilModifier.minM2+Math.random() *
(AirfoilModifier.maxM2-AirfoilModifier.minM2);
                  double m3=AirfoilModifier.minM3+Math.random() *
(AirfoilModifier.maxM3-AirfoilModifier.minM3);
//                double m4=AirfoilModifier.minM4+Math.random() *
(AirfoilModifier.maxM4-AirfoilModifier.minM4);
                  double n1=AirfoilModifier.minN1+Math.random() *
(AirfoilModifier.maxN1-AirfoilModifier.minN1);
                  double n2=AirfoilModifier.minN2+Math.random() *
(AirfoilModifier.maxN2-AirfoilModifier.minN2);
                  double n3=AirfoilModifier.minN3+Math.random() *
(AirfoilModifier.maxN3-AirfoilModifier.minN3);
//                double n4=AirfoilModifier.minN4+Math.random() *
(AirfoilModifier.maxN4-AirfoilModifier.minN4);

                  /**
                  double x1=m1;
                  double x2=m2;
```

```
                  double y1=n1;
                  double y2=n2;
                  **/

                  /**Airfoil mutant = new
Airfoil(x1,y1,x2,y2,m1,n1,m2,n2,m3,n3);
                   **/
                  Airfoil mutant = new
Airfoil(x1,x2,y1,y2,m1,m2,m3,n1,n2,n3);
                  AirfoilModifier.modAirfoil(mutant);
                  gambitAirfoils gt = new gambitAirfoils(mutant);
                  gt.buildAirfoil(0.02);


      if(gt.getThickness(0.02,0,12)<=(double)maxThickness1&&gt.getThick
ness(0.02,0,12)>=(double)minThickness1

      &&gt.getThickness(0.02,13,35)<=(double)maxThickness2&&gt.getThick
ness(0.02,13,35)>=(double)minThickness2

      &&gt.getThickness(0.02,36,50)<=(double)maxThickness3&&gt.getThick
ness(0.02,36,50)>=(double)minThickness3

      &&gt.getminThickness(0.02)>=0&&gt.getminThicknessInterval(0.02,8,
43)>=0.02){
                        x=false;
                        airfoiladd = mutant;
                        GennoFluent.removeAirfoilAt(counter2);
                        GennoFluent.addAirfoilAt(airfoiladd, counter2);
                        counter2 ++;
                        /**
                        //System.out.println("Computing fitness for
mutation individuals");
                        //mutant.fitness = gt.getScoreWithFluent();
                        airfoiladd = mutant;
                        airfoilremove=airfoil;

      //System.out.println("maxthickness3="+gt.getmaxThickness(0.02))
                         **/
                  }else
                        x=true;
                  }
                  //GennoFluent.removeAirfoil(airfoilremove);
                  System.out.println("Airfoil removed for mutation");
            //    GenforFluent.addAirfoil(airfoiladd);
                  System.out.println("Airfoil added for mutation");
            }
      }

      for(int k=0;k<GenforFluent.getAirfoilVectorSize();k++){
            //nextGen.addAirfoil(one.getAirfoil(i));
            nextGen.addAirfoil(GenforFluent.getAirfoil(k));
      }

      for(int l=0;l<GennoFluent.getAirfoilVectorSize();l++){
            //nextGen.addAirfoil(one.getAirfoil(i));
            nextGen.addAirfoil(GennoFluent.getAirfoil(l));
```

```
	}

	/**part = GenforFluent;
	nofluent = GennoFluent;
	System.out.println(GenforFluent.getAirfoilVectorSize());
	System.out.println(GennoFluent.getAirfoilVectorSize());
	System.out.println(part.getAirfoilVectorSize());**/
}
public void EvaIndividual(double X1, double Y1, double X2, double
Y2,double M1,double N1,double M2,double N2,double M3,double N3){
	double timeStep=0.02;
	Airfoil af=new Airfoil(X1,X2,Y1,Y2,M1,M2,M3,N1,N2,N3);
	gambitAirfoils gt=new gambitAirfoils(af);
	gt.buildAirfoil(timeStep);
	gt.getmaxThickness(timeStep);
	System.out.println("gt.getThickness(0.02,0,12)="+gt.getThickness(
0.02,0,12));
	System.out.println("gt.getThickness(0.02,13,35)="+gt.getThickness
(0.02,13,35));
	System.out.println("gt.getThickness(0.02,36,50)="+gt.getThickness
(0.02,36,50));
	gt.getScoreWithFluent(1, 1, af.getFitness());
	gt.publishFile("airfoil.dat");
}


public static void main(String[] args){
	gaflatback firstTry = new gaflatback(20,300,0.5,.04);
	firstTry.EvaIndividual(0.0,0.04054234068441003,0.2881659110574578
7,0.26740633856534124,0.21382533412454174,-
0.03143924893289855,0.3023275506446589,0.18683151956640545,0.7415214483
646052,0.003089133055364832);
//	firstTry.runOptimization();


}
}
```

A.2 gambitAirfoils.java

```
package gaflatback;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.Vector;


public class gambitAirfoils {
	public static final double X0=0;
	public static final double Y0=0;
	public static final double X1=0;
	public static final double X3=1;
//	public static final double X3=0.998629;
	public static final double M1=0;
	public static final double Y3=0;
//	public static final double Y3=-0.0523359;
	public static final double M4=1;
	public static final double N4=0;
//	public static final double M4=0.998629;
//	public static final double N4=-0.0523359;

	public static final int extraPoints = 1;
	public double
X2,Y1,Y2,N1,M2,N2,M3,N3,Ax,Bx,Cx,Ay,By,Cy,Dm,Em,Fm,Gm,Dn,En,Fn,Gn,fitne
ss, thickness;
	public double[] xPoints, yPoints, mPoints, nPoints;
	public int iterations;
	public double max=-5;
	public double min=1;


	public gambitAirfoils(Airfoil af){

		this.Y1 = af.Y1;
		this.X2 = af.X2;
		this.Y2 = af.Y2;
		this.N1 = af.N1;
		this.M2 = af.M2;
		this.N2 = af.N2;
		this.M3 = af.M3;
		this.N3 = af.N3;

		this.Cx = getCx();
		this.Bx = getBx();
		this.Ax = getAx();
		this.Cy = getCy();
		this.By = getBy();
		this.Ay = getAy();
```

```
            this.Dm = getDm();
            this.Em = getEm();
            this.Fm = getFm();
            this.Gm = getGm();
            this.Dn = getDn();
            this.En = getEn();
            this.Fn = getFn();
            this.Gn = getGn();

      }

       public void buildAirfoil(double timeStep1){
                  double timeStep =1*timeStep1;
                  iterations = (int) (1/timeStep);
                  xPoints = new double[iterations + extraPoints];
                  yPoints = new double[iterations + extraPoints];
                  mPoints = new double[iterations + extraPoints];
                  nPoints = new double[iterations + extraPoints];
                  double t = 0;

                  //bezier curve first
                  for(int i=0;i<iterations;i++){
                        xPoints[i] = (Ax*Math.pow(t,3)) +
(Bx*Math.pow(t,2)) + (Cx*t) + X0;
                        yPoints[i] = (Ay*Math.pow(t,3)) +
(By*Math.pow(t,2)) + (Cy*t) + Y0;
                        mPoints[i] = (Dm*Math.pow(t,4)) +
(Em*Math.pow(t,3)) + (Fm*Math.pow(t, 2)) + Gm*t+X0;
                        nPoints[i] = (Dn*Math.pow(t,4)) +
(En*Math.pow(t,3)) + (Fn*Math.pow(t, 2)) + Gn*t+Y0;
                        t = t + timeStep;
                  }

                  //Add tail points
                  for(int j=iterations;j<iterations+extraPoints;j++){
                        if(j==iterations){
                              xPoints[iterations] = X3;
                              yPoints[iterations] = Y3;
                              mPoints[iterations] = M4;
                              nPoints[iterations] = N4;
                              /**
                              xPoints[iterations] = M4;
                              yPoints[iterations] = N4;
                              **/

      }
                  }
      }
       public double getThickness(double timeStep, int vertexBegin, int
vertexEnd){
            iterations = (int) (1/timeStep);
            double range = 0.02;
            max=-5;
            for(int k=vertexBegin; k<=vertexEnd ;k++){
                  for(int c=0;c<iterations+extraPoints;c++){
                        if (Math.abs(xPoints[k]- mPoints[c])<range){
                              thickness=yPoints[k]-nPoints[c];
```

```
                              //System.out.println("thickness = "+
thickness);
                              if(thickness>=max){
                                    max=thickness;
                              }
                        }
                  }
            }
//           System.out.println("thickness for " + vertexBegin +  "-" +
vertexEnd + " " + max);
                  return max;
      }

      public double getminThicknessInterval(double timeStep, int
vertexBegin, int vertexEnd){
            iterations = (int) (1/timeStep);
            double range = 0.02;
            min=2;
            for(int k=vertexBegin; k<=vertexEnd ;k++){
                  for(int c=0;c<iterations+extraPoints;c++){
                        if (Math.abs(xPoints[k]- mPoints[c])<range){
                              thickness=yPoints[k]-nPoints[c];
                              //System.out.println("thickness = "+
thickness);
                              if(thickness<=min){
                                    min=thickness;
                              }
                        }
                  }
            }
            return min;
      }



      public double getmaxThickness(double timeStep){
            iterations = (int) (1/timeStep);

//System.out.println("ypoints"+yPoints.length+"npoints"+nPoints.length)
;
                  for(int k=0;k<iterations+extraPoints;k++){
                        //System.out.println(yPoints[k] + " ," +
nPoints[k]);
                        thickness=yPoints[k]-nPoints[k];
                        //thickness=-2*nPoints[k];
                        if(thickness>=max)
                              max=thickness;
                  }
                  return max;
      }

      public double getminThickness(double timeStep){
            iterations = (int) (1/timeStep);
            double range = 0.02;
            min=1;
            for(int k=0;k<iterations+extraPoints;k++){
                  for(int c=0;c<iterations+extraPoints;c++){
```

```
                        if (Math.abs(xPoints[k]- mPoints[c])<range){
                              thickness=yPoints[k]-nPoints[c];
                              //System.out.println("thickness = "+
thickness);
                              if(thickness<=min){
                                    min=thickness;
                              }
                        }
                  }
            }
                  return min;
      }



      public double getFitness(){
                        return fitness;
                  }

      public double getScoreWithFluent(int generation, int iteration,
double fitness){
            //System.out.println("Fitness: " + fitness);
                  if (fitness != 1000000){
                        System.out.println("Fitness: " + fitness);

                        return fitness;
                  }

                  else

                  publishFile("airfoil.dat");

                  long maximumWait = 1000 * 60 * 60;
                  try{

                        File clhist = new File("cl-history");
                        File cdhist = new File("cd-history");
                        File trans = new File("trans.jou");
                        File d1 = new File("default_id.dbs");
                        File d2 = new File("default_id.jou");
                        File d3 = new File("default_id.trn");
                        File mesh = new
File("Free_Surface_Hydrofoilmesh");

                        while(clhist.exists() || cdhist.exists() ||
trans.exists() || d1.exists() || d2.exists() || d3.exists()){
                              Process cleanupProc2 =
Runtime.getRuntime().exec("cleanup.bat");
                              cleanupProc2.waitFor();
                        }
                        long gambitwait = 1000 * 60 * 10;
                        long gambittimestart =
System.currentTimeMillis();
                        boolean x = true;

                        @SuppressWarnings("unused")
```

```
                        Process gambitProc =
Runtime.getRuntime().exec("gambitTest.bat");
                        while (x){
                              if (gambitwait <
System.currentTimeMillis() - gambittimestart){
                                    Process killGambit =
Runtime.getRuntime().exec("gambitKill.bat");
                                    killGambit.waitFor();
                                    x = false;
                              }
                              if (mesh.exists()){
                                    x = false;
                              }
                        }

                        Process fluentProc =
Runtime.getRuntime().exec("fluentTest.bat");
                        fluentProc.waitFor();

                        File transcript = new File("trans.jou");
                  //wait and check for Fluent's return at one second
intervals
                        long time = System.currentTimeMillis();
                        while (true) {
                              long interval =
System.currentTimeMillis() - time;
                              if( interval > maximumWait){
                                    publishFile("trans.jou");
                               System.out.println("FLUENT DID NOT RETURN A
RESULT FOR THIS CASE: #" + iteration);
                               Process fluentKill =
Runtime.getRuntime().exec("fluentKill.bat");
                               fluentKill.waitFor();
                               return -100;
                                    }

                                    if(transcript.exists()){
                                    BufferedReader cdInput =  new
BufferedReader(new FileReader(new File("cd-history")));
                                    BufferedReader clInput =  new
BufferedReader(new FileReader(new File("cl-history")));

                                    double cd = 0;
                                    double cl = 0;
                                    String linecl;
                                    String linecd;

                                    //skip the first two lines
                                    for(int i=0;i < 1501; i++){
                                          clInput.readLine();
                                          cdInput.readLine();
                                    }
                                    linecl = clInput.readLine();
                                    linecd = cdInput.readLine();
                                    linecl =
linecl.substring(linecl.indexOf("\t")+1);
```

```
                                    linecd =
linecd.substring(linecd.indexOf("\t")+1);

                                    cl = Double.parseDouble(linecl);
                                    cd = Double.parseDouble(linecd);

                                    System.out.println("cl = " + cl +
"," + "cd = " + cd);
                                    System.out.println("Cl/Cd = " +
cl/cd);
                                    if (cl/cd > 75) {
                                          return -200;
                                    }
                                    return cl/cd;
                              }

                        }

                  }

                  catch(Exception e){
                        e.printStackTrace();
                        return -1;
                  }
            }


            public boolean publishFile(String filename){
                  try{
                        // Create file
                        FileWriter fstream = new
FileWriter(filename);
                        BufferedWriter out = new
BufferedWriter(fstream);
                        out.write(2*xPoints.length+ " 2\n");
                        for(int i=0;i<xPoints.length;i++){
                          out.write(xPoints[i]+ " " + yPoints[i] +
" 0\n");
                        }


                        for(int i=0;i<mPoints.length;i++){
                          out.write(mPoints[i]+ " " + nPoints[i] +
" 0\n");
                        }
                        out.close();
                        return true;
                  }
                  catch (Exception e){
                        System.err.println("Error: " +
e.getMessage());
                        return false;
                  }
            }


            public double getCx(){
```

```
            return 3*(X1-X0);
        }

        public double getBx(){
            return 3*(X2-X1)-3*(X1-X0);
        }

        public double getAx(){
            return X3-X0-3*(X1-X0)-3*(X2-X1)+3*(X1-X0);
        }

        public double getCy(){
            return 3*(Y1-Y0);
        }

        public double getBy(){
            return 3*(Y2-Y1)-3*(Y1-Y0);
        }

        public double getAy(){
            return Y3-Y0-3*(Y1-Y0)-3*(Y2-Y1)+3*(Y1-Y0);
        }
        public double getGm(){
            return 4*(M1-X0);
        }

        public double getFm(){
            return -1.5*4*(M1-X0)+6*(M2-M1);
        }

        public double getEm(){
            return 4*(M3-X0)-2*(-1.5*4*(M1-X0)+6*(M2-M1))-
3*4*(M1-X0);
        }

        public double getDm(){
            return M4-M3-3*(M3-M2)+0.5*(-1.5*4*(M1-X0)+6*(M2-
M1))+0.5*4*(M1-X0);
        }

      public double getGn(){
        return 4*(N1-Y0);
      }

      public double getFn(){
        return -1.5*4*(N1-Y0)+6*(N2-N1);
      }

      public double getEn(){
        return 4*(N3-Y0)-2*(-1.5*4*(N1-Y0)+6*(N2-N1))-3*4*(N1-Y0);
      }

      public double getDn(){
        return N4-N3-3*(N3-N2)+0.5*(-1.5*4*(N1-Y0)+6*(N2-
N1))+0.5*4*(N1-Y0);
      }
      }
```

A.3 generation.java

```
package gaflatback;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.util.Vector;

public class generation {
	public Vector<Airfoil> airfoils;
	private int genSize;
	private double timeStep = 0.02;

	public generation(int genSize){
		this.genSize = genSize;
		this.airfoils = new Vector<Airfoil>();
	}

	public generation(){
		this.airfoils = new Vector<Airfoil>();
	}

	public void addAirfoil(Airfoil airfoil){
		airfoils.add(airfoil);
	}

	public void addAirfoilAt(Airfoil airfoil,int i){
		airfoils.insertElementAt(airfoil, i);
	}

	public void removeAirfoilAt(int i){
		airfoils.removeElementAt(i);
	}


	public void replaceAirfoil(Airfoil airfoil, int i){
		airfoils.removeElementAt(i);
		airfoils.insertElementAt(airfoil,i);
	}


	public Airfoil getAirfoil(int i){
		return airfoils.elementAt(i);
	}


	public int getGenSize(){
		return genSize;
	}


	public int getAirfoilVectorSize(){
		return airfoils.size();
	}
```

```
	public void removeAirfoils(int num){
		for(int i=0;i<num;i++){
			airfoils.removeElementAt(0);
		}
	}


	public boolean removeAirfoil(Airfoil airfoil){
		return airfoils.remove(airfoil);
	}


	/**Get the airfoils and their X1,X2,Y1,Y2,M1,M2,M3,N1,N2,N3
values*/
	public void outputAirfoils(){
		for(int i=0;i<airfoils.size();i++){
			Airfoil af = airfoils.elementAt(i);
			System.out.println("Airfoil variables: " + af.X1 + ",
" + af.Y1 + ", " + af.X2 + ", " + af.Y2 + ", " + af.M1+ ", " + af.N1 +
", " + af.M2
					+ ", " + af.N2 + ", " + af.M3+ ", " +
af.N3 + ", " +"fitness: " + af.fitness);
		}
	}

	/**create the coordinates for this generation of airfoils and
store in vector "airfoils"*/
	public void determineFitness(int generation){
		for(int i=0;i<airfoils.size();i++){
			Airfoil airfoil = airfoils.elementAt(i);
			gambitAirfoils gt = new gambitAirfoils(airfoil);
			gt.buildAirfoil(timeStep);
			airfoils.elementAt(i).fitness =
gt.getScoreWithFluent(generation,i ,airfoils.elementAt(i).fitness);
			try{
				BufferedWriter recordWriter =  new
BufferedWriter(new FileWriter(new File("record.txt"), true));
				recordWriter.write("Generation: " + generation
+  "; ("  + airfoils.elementAt(i).getX1() + "," +
airfoils.elementAt(i).getY1() + "," + airfoils.elementAt(i).getX2() +
"," + airfoils.elementAt(i).getY2() +
					","+
airfoils.elementAt(i).getM1()+","+ airfoils.elementAt(i).getN1()+","+
airfoils.elementAt(i).getM2()+ "," + airfoils.elementAt(i).getN2()+ ","
+ airfoils.elementAt(i).getM3()+ "," +
					airfoils.elementAt(i).getN3()+ ")
fitness: " + airfoils.elementAt(i).fitness + "\n");
				recordWriter.close();
			}
			catch(Exception e){
				e.printStackTrace();
			}
		}
	}

	/**Get airfoil of the generation that has the highest coefficient
of lift*/
```

```
	public Airfoil getBestAirfoil(){
		//Airfoil bestAirfoil = new Airfoil();
		for(int i=0;i<airfoils.size();i++){
			System.out.println("Currently running airfoil cl = "
+ airfoils.elementAt(i).getFitness());
			if(airfoils.elementAt(i).getFitness() >
gaflatback.bestAirfoil.getFitness()){
				gaflatback.bestAirfoil = airfoils.elementAt(i);
				System.out.println("New best airfoil cl = " +
gaflatback.bestAirfoil.getFitness());
			}
			else{
				System.out.println("ELSECurrent best airfoil cl
= " + gaflatback.bestAirfoil.getFitness());
			}
		}

		return gaflatback.bestAirfoil;
	}
}
```

A.4 airfoil.java

```
package gaflatback;

public class Airfoil {

	public double X1,X2,Y1,Y2,M1,M2,M3,N1,N2,N3,Cx,Bx,Ax,
	Cy,By,Ay,Dm,Em,Fm,Gm,Dn,En,Fn,Gn;
	/**public static final double
M3=0.81,X1=0.12,X2=0.9,Y1=0.36,Y2=0.06**/;
	public double fitness;
	public double X0=0, Y0=0, X3=1, Y3=0.04375, M4=1, N4=0.0;

	/**Constructor for copying an existing airfoil**/
	public Airfoil(Airfoil af){
		this(af.X1, af.X2, af.Y1, af.Y2,af.M1, af.M2, af.M3, af.N1,
af.N2, af.N3 );
		}

	/**Default constructor.  Initialize everything to 0.**/
	public Airfoil(){
		X1=0;
		X2=0;
		Y1=0;
		Y2=0;
		M1 = 0;
		M2 = 0;

		N1 = 0;
		N2 = 0;
		N3 = 0;
		fitness = 0;
	}

	/**Main constructor**/
	public Airfoil(double X1, double X2, double Y1, double Y2,
			double M1, double M2, double M3, double N1,
			double N2, double N3 ){
		this.X1=X1;
		this.X2=X2;
		this.Y1=Y1;
		this.Y2=Y2;
		this.M1 = M1;
		this.M2 = M2;
		this.M3 = M3;
		this.N1 = N1;
		this.N2 = N2;
		this.N3 = N3;
		this.fitness = 1000000;
	}

	public Airfoil(double X1, double X2, double Y1, double Y2,
	double M1, double M2, double M3, double N1,
	double N2, double N3,double fitness ){
    this.X1=X1;
    this.X2=X2;
    this.Y1=Y1;
```

```
this.Y2=Y2;

this.M1 = M1;
this.M2 = M2;
this.M3 = M3;
this.N1 = N1;
this.N2 = N2;
this.N3 = N3;
this.fitness = fitness;
  }



  public double getX1(){
        return X1;
  }
  public double getY1(){
        return Y1;
  }
  public double getX2(){
        return X2;
  }
  public double getY2(){
        return Y2;
  }

  public double getM1(){
        return M1;
  }

  public double getN1(){
        return N1;
  }

  public double getM2(){
        return M2;
  }

  public double getN2(){
        return N2;
  }

  public double getM3(){
        return M3;
  }

  public double getN3(){
        return N3;
  }

  public double getFitness(){
        return fitness;
  }


        }
```

A.5 airfoil.Modifier.java

```
package gaflatback;

public class AirfoilModifier {

		//symetric

		public static final double minX1=0;
		public static final double maxX1=0.95;
		public static final double minX2=0.0;
		public static final double maxX2=0.98;

		public static final double minY1=0.01;
		public static final double maxY1=0.3;
		public static final double minY2=-0.1;
		public static final double maxY2=0.4;

		public static final double minM1=0.02;
		public static final double maxM1=0.8;
		public static final double minM2=0.01;
		public static final double maxM2=0.98;
		public static final double minM3=0.1;
		public static final double maxM3=0.98;

		public static final double minN1=-0.3;
		public static final double maxN1=0;
		public static final double minN2=-0.2;
		public static final double maxN2=0.4;
		public static final double minN3=-0.2;
		public static final double maxN3=0.4;


		public static void modAirfoil(Airfoil af){
			// make sure all the variables in the right range


			if(af.X1 < minX1)
				af.X1 = minX1;
			else if(af.X1 > maxX1)
				af.X1 = maxX1;
			else
				af.X1 = af.X1;

			if(af.X2 < minX2)
				af.X2 = minX2;
			else if(af.X2 > maxX2)
				af.X2 = maxX2;
			else
				af.X2 = af.X2;

			if(af.Y1 < minY1)
				af.Y1 = minY1;
			else if(af.Y1 > maxY1)
				af.Y1 = maxY1;
			else
				af.Y1 = af.Y1;
```

```
if(af.Y2 < minY2)
      af.Y2 = minY2;
else if(af.Y2 > maxY2)
      af.Y2 = maxY2;
else
      af.Y2 = af.Y2;


if(af.M1 < minM1)
      af.M1 = minM1;
else if(af.M1 > maxM1)
      af.M1 = maxM1;
else
      af.M1 = af.M1;

if(af.M2 < minM2)
      af.M2 = minM2;
else if(af.M2 > maxM2)
      af.M2 = maxM2;
else
      af.M2 = af.M2;

if(af.M3 < minM3)
      af.M3 = minM3;
else if(af.M3 > maxM3)
      af.M3 = maxM3;
else
      af.M3 = af.M3;


if(af.N1 < minN1)
      af.N1 = minN1;
else if(af.N1 > maxN1)
      af.N1 = maxN1;
else
      af.N1 = af.N1;

if(af.N2 < minN2)
      af.N2 = minN2;
else if(af.N2 > maxN2)
      af.N2 = maxN2;
else
      af.N2 = af.N2;

if(af.N3 < minN3)
      af.N3 = minN3;
else if(af.N3 > maxN3)
      af.N3 = maxN3;
else
      af.N3 = af.N3;


//make x1,x2,m1,m2 in order
```

```
                        if(af.X2<af.X1)
                              af.X2=af.X1+Math.random() * (maxX2-af.X1);


                        if(af.M2<af.M1)
                              af.M2=af.M1+Math.random() * (maxM2-af.M1);

                        if(af.M3<af.M2)
                              af.M3=af.M2+Math.random() * (maxM3-af.M2);

            }
      }
```

A.6 bubbleSort.java

```
package gaflatback;
import java.util.Vector;

/*
 * @(#)BubbleSortAlgorithm.java     1.6 95/01/31 James Gosling
 *
 * Copyright (c) 1994 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */

/**
 * A bubble sort demonstration algorithm
 * SortAlgorithm.java, Thu Oct 27 10:32:35 1994
 *
 * @author James Gosling
 * @version      1.6, 31 Jan 1995
 *
 * Modified 23 Jun 1995 by Jason Harrison@cs.ubc.ca:
 *   Algorithm completes early when no items have been swapped in the
 *   last pass.
 *
 *
 */
public class BubbleSorter {
       public static void sort(generation a){
                  for (int i = a.getGenSize(); --i>=0; ) {
                  boolean flipped = false;
                  for (int j = 0; j<i; j++) {

                        if (a.getAirfoil(j).getFitness() >
a.getAirfoil(j+1).getFitness()) {
                              Airfoil T = a.getAirfoil(j);
                              a.replaceAirfoil(a.getAirfoil(j+1), j);
                              a.replaceAirfoil(T,j+1);
                              flipped = true;
                        }
                  }
                  if (!flipped) {
                        return;
                  }
            }
         }
}
```

A.7 gambittest.bat

```
C:\Fluent.Inc\ntbin\ntx86\gambit.exe -inputfile
"C:\Travis\Thesis\Free_Surface_Hydrofoil\Hydrofoil_Gambit.jou"
```

A.8 fluenttest.bat

```
Start C:\Travis\Thesis\Free_Surface_Hydrofoil\FLUENT 2ddp -i
"C:\Travis\Thesis\Free_Surface_Hydrofoil\Hydrofoil_Fluent.jou"
```

A.9 gambitkill.bat

```
taskkill /f /im gambit.exe

taskkill /f /im exceed.exe
```

A.10 fluentkill.bat

```
taskkill /f /im fl1214s.exe
```

A.11 Hydrofoil_Gambit.jou

```
/ Journal File for GAMBIT 2.4.6, Database 2.4.4, ntx86 SP2007051421
/ Identifier "default_id544"
/ File opened for write Wed Sep 07 11:19:15 2011.
/ERROR occurred in the next command!
coordinate create cartesian oldsystem "c_sys.1" offset 0 0 0 axis1 "x"
angle1 \
  0 axis2 "y" angle2 0 axis3 "z" angle3 0 rotation
import iceminput
"C:\\Travis\\Thesis\\Free_Surface_Hydrofoil\\airfoil.dat" \
  face net
coordinate activate "c_sys.1"
edge create "t" nurbs "vertex.1" "vertex.2" "vertex.3" "vertex.4"
"vertex.5" \
  "vertex.6" "vertex.7" "vertex.8" "vertex.9" "vertex.10" "vertex.11" \
  "vertex.12" "vertex.13" "vertex.14" "vertex.15" "vertex.16"
"vertex.17" \
  "vertex.18" "vertex.19" "vertex.20" "vertex.21" "vertex.22"
"vertex.23" \
  "vertex.24" "vertex.25" "vertex.26" "vertex.27" "vertex.28"
"vertex.29" \
  "vertex.30" "vertex.31" "vertex.32" "vertex.33" "vertex.34"
"vertex.35" \
  "vertex.36" "vertex.37" "vertex.38" "vertex.39" "vertex.40"
"vertex.41" \
  "vertex.42" "vertex.43" "vertex.44" "vertex.45" "vertex.46"
"vertex.47" \
  "vertex.48" "vertex.49" "vertex.50" "vertex.51" interpolate
edge create "b" nurbs "vertex.52" "vertex.53" "vertex.54" "vertex.55" \
  "vertex.56" "vertex.57" "vertex.58" "vertex.59" "vertex.60"
"vertex.61" \
  "vertex.62" "vertex.63" "vertex.64" "vertex.65" "vertex.66"
"vertex.67" \
```

```
  "vertex.68" "vertex.69" "vertex.70" "vertex.71" "vertex.72"
"vertex.73" \
  "vertex.74" "vertex.75" "vertex.76" "vertex.77" "vertex.78"
"vertex.79" \
  "vertex.80" "vertex.81" "vertex.82" "vertex.83" "vertex.84"
"vertex.85" \
  "vertex.86" "vertex.87" "vertex.88" "vertex.89" "vertex.90"
"vertex.91" \
  "vertex.92" "vertex.93" "vertex.94" "vertex.95" "vertex.96"
"vertex.97" \
  "vertex.98" "vertex.99" "vertex.100" "vertex.101" "vertex.102"
interpolate
face create "h" wireframe "t" "b" real
vertex create "a" coordinates -5 -5 0
vertex create "b" coordinates -5 5 0
vertex create "c" coordinates 7 5 0
vertex create "d" coordinates 7 -5 0
edge create "ab" straight "a" "b"
edge create "bc" straight "b" "c"
edge create "cd" straight "c" "d"
edge create "da" straight "a" "d"
face create "face" wireframe "ab" "bc" "cd" "da" real
face subtract "face" faces "h"
undo begingroup
edge modify "edge.7" backward
edge picklink "edge.7"
edge mesh "edge.7" successive ratio1 0.99 intervals 90
undo endgroup
undo begingroup
edge picklink "edge.8"
edge mesh "edge.8" successive ratio1 0.99 intervals 90
undo endgroup
face mesh "face" pave size 0.25
physics create "inlet" btype "PRESSURE_INLET" edge "ab"
physics create "ceiling" btype "WALL" edge "bc"
physics create "outlet" btype "PRESSURE_OUTLET" edge "cd"
physics create "floor" btype "WALL" edge "da"
physics create "hydrofoil" btype "WALL" edge "edge.7" "edge.8"
export fluent5 \

"C:\\Travis\\Thesis\\Free_Surface_Hydrofoil\\Free_Surface_Hydrofoilmesh
" \
  nozval
/ File closed at Wed Sep 07 11:26:09 2011, 75.58 cpu second(s),
28023160 maximum memory.
```

A.12 Hydrofoil_Fluent.jou

```
(cx-gui-do cx-activate-item "MenuBar*ReadSubMenu*Mesh...")
(cx-gui-do cx-set-text-entry "Select File*Text"
"C:\Travis\Thesis\Free_Surface_Hydrofoil\Free_Surface_Hydrofoilmesh")
(cx-gui-do cx-activate-item "Select File*OK")
(cx-use-window-id 1)
(cx-set-camera '(0.506374 0.101696 2.86072) '(0.506374 0.101696 0)
'(1.3374e-007 1 1.16415e-010) 1.14429 1.14429 "perspective")

(cx-gui-do cx-activate-item "MenuBar*FileMenu*Save Picture...")
(cx-gui-do cx-set-toggle-button "Save
Picture*Frame1(Format)*ToggleBox1(Format)*JPEG" #f)
(cx-gui-do cx-activate-item "Save
Picture*Frame1(Format)*ToggleBox1(Format)*JPEG")
(cx-gui-do cx-set-toggle-button "Save
Picture*Frame2(Coloring)*ToggleBox2(Coloring)*Color" #f)
(cx-gui-do cx-activate-item "Save
Picture*Frame2(Coloring)*ToggleBox2(Coloring)*Color")
(cx-gui-do cx-activate-item "Save
Picture*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-text-entry "Select File*FilterText"
"c:\travis\thesis\free_surface_hydrofoil\pictures\*")
(cx-gui-do cx-activate-item "Select File*Apply")
(cx-gui-do cx-set-text-entry "Select File*Text" "Mesh.jpg")
(cx-gui-do cx-activate-item "Select File*OK")
(cx-gui-do cx-activate-item "Warning*OK")
(cx-gui-do cx-activate-item "Save
Picture*PanelButtons*PushButton2(Cancel)")
(cx-gui-do cx-set-toggle-button
"General*Frame1*Table1*Frame3*Frame1*CheckButton1(Gravity)" #f)
(cx-gui-do cx-activate-item
"General*Frame1*Table1*Frame3*Frame1*CheckButton1(Gravity)")
(cx-gui-do cx-set-real-entry-list
"General*Frame1*Table1*Frame3*Frame1*Frame2(Gravitational
Acceleration)*RealEntry2(Y)" '( -9.81))
(cx-gui-do cx-activate-item
"General*Frame1*Table1*Frame3*Frame1*Frame2(Gravitational
Acceleration)*RealEntry2(Y)")
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton3(Models)")
(cx-gui-do cx-set-list-selections
"Models*Frame1*Table1*Frame1*List1(Models)" '( 0))
(cx-gui-do cx-activate-item
"Models*Frame1*Table1*Frame1*List1(Models)")
(cx-gui-do cx-activate-item "Models*Frame1*Table1*PushButton2(Edit)")
(cx-gui-do cx-set-toggle-button "Multiphase
Model*Frame1*Table1*Frame1(Model)*ToggleBox1(Model)*Volume of Fluid"
#f)
(cx-gui-do cx-activate-item "Multiphase
Model*Frame1*Table1*Frame1(Model)*ToggleBox1(Model)*Volume of Fluid")
(cx-gui-do cx-set-toggle-button "Multiphase
Model*Frame1*Table1*Frame5(Volume Fraction Parameters)*Table5(Volume
Fraction Parameters)*CheckButton3(Open Channel Flow)" #f)
(cx-gui-do cx-activate-item "Multiphase
Model*Frame1*Table1*Frame5(Volume Fraction Parameters)*Table5(Volume
Fraction Parameters)*CheckButton3(Open Channel Flow)")
```

```
(cx-gui-do cx-set-toggle-button "Multiphase
Model*Frame1*Table1*Frame5(Volume Fraction Parameters)*Table5(Volume
Fraction Parameters)*CheckButton4(Open Channel Wave BC)" #f)
(cx-gui-do cx-activate-item "Multiphase
Model*Frame1*Table1*Frame5(Volume Fraction Parameters)*Table5(Volume
Fraction Parameters)*CheckButton4(Open Channel Wave BC)")
(cx-gui-do cx-set-toggle-button "Multiphase
Model*Frame1*Table1*Frame5(Volume Fraction Parameters)*Table5(Volume
Fraction Parameters)*Frame1(Scheme)*ToggleBox1(Scheme)*Explicit" #f)
(cx-gui-do cx-activate-item "Multiphase
Model*Frame1*Table1*Frame5(Volume Fraction Parameters)*Table5(Volume
Fraction Parameters)*Frame1(Scheme)*ToggleBox1(Scheme)*Explicit")
(cx-gui-do cx-set-toggle-button "Multiphase
Model*Frame1*Table1*Frame5(Volume Fraction Parameters)*Table5(Volume
Fraction Parameters)*Frame1(Scheme)*ToggleBox1(Scheme)*Implicit" #f)
(cx-gui-do cx-activate-item "Multiphase
Model*Frame1*Table1*Frame5(Volume Fraction Parameters)*Table5(Volume
Fraction Parameters)*Frame1(Scheme)*ToggleBox1(Scheme)*Implicit")
(cx-gui-do cx-activate-item "Multiphase
Model*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-list-selections
"Models*Frame1*Table1*Frame1*List1(Models)" '( 2))
(cx-gui-do cx-activate-item
"Models*Frame1*Table1*Frame1*List1(Models)")
(cx-gui-do cx-activate-item "Models*Frame1*Table1*PushButton2(Edit)")
(cx-gui-do cx-set-toggle-button "Viscous
Model*Frame1*Table1*Frame1(Model)*ToggleBox1(Model)*Spalart-Allmaras (1
eqn)" #f)
(cx-gui-do cx-activate-item "Viscous
Model*Frame1*Table1*Frame1(Model)*ToggleBox1(Model)*Spalart-Allmaras (1
eqn)")
(cx-gui-do cx-activate-item "Viscous
Model*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton4(Materials)")
(cx-gui-do cx-set-list-selections
"Materials*Frame1*Table1*Frame1*List1(Materials)" '( 0))
(cx-gui-do cx-activate-item
"Materials*Frame1*Table1*Frame1*List1(Materials)")
(cx-gui-do cx-activate-item
"Materials*Frame1*Table1*Frame2*ButtonBox2*PushButton1(Create/Edit)")
(cx-gui-do cx-activate-item "Create/Edit
Materials*Frame1*Table1*Frame1*Frame3*ButtonBox3*PushButton1(FLUENT
Database)")
(cx-gui-do cx-set-list-selections "Database
Materials*Frame1*Table1*Frame1*Frame1*List1(Materials)" '( 552))
(cx-gui-do cx-activate-item "Database
Materials*Frame1*Table1*Frame1*Frame1*List1(Materials)")
(cx-gui-do cx-set-position "Database Materials" '(x 106 y 224))
(cx-gui-do cx-activate-item "Database
Materials*PanelButtons*PushButton1(Copy)")
(cx-gui-do cx-activate-item "Database
Materials*PanelButtons*PushButton1(Close)")
(cx-gui-do cx-activate-item "Create/Edit
Materials*PanelButtons*PushButton1(Close)")
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton5(Phases)")
```

```
(cx-gui-do cx-set-list-selections
"Phases*Frame1*Table1*Frame1*List1(Phases)" '( 0))
(cx-gui-do cx-activate-item
"Phases*Frame1*Table1*Frame1*List1(Phases)")
(cx-gui-do cx-set-list-selections
"Phases*Frame1*Table1*Frame1*List1(Phases)" '( 0))
(cx-gui-do cx-activate-item
"Phases*Frame1*Table1*Frame1*List1(Phases)")
(cx-gui-do cx-set-list-selections
"Phases*Frame1*Table1*Frame1*List1(Phases)" '( 0))
(cx-gui-do cx-activate-item
"Phases*Frame1*Table1*Frame1*List1(Phases)")
(cx-gui-do cx-activate-item
"Phases*Frame1*Table1*Frame2*ButtonBox2*PushButton1(Edit)")
(cx-gui-do cx-set-text-entry "phase-domain-2*TextEntry1(Name)" "air")
(cx-gui-do cx-activate-item "phase-domain-
2*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-list-selections
"Phases*Frame1*Table1*Frame1*List1(Phases)" '( 0))
(cx-gui-do cx-activate-item
"Phases*Frame1*Table1*Frame1*List1(Phases)")
(cx-gui-do cx-set-list-selections
"Phases*Frame1*Table1*Frame1*List1(Phases)" '( 1))
(cx-gui-do cx-activate-item
"Phases*Frame1*Table1*Frame1*List1(Phases)")
(cx-gui-do cx-activate-item
"Phases*Frame1*Table1*Frame2*ButtonBox2*PushButton1(Edit)")
(cx-gui-do cx-set-list-selections "phase-domain-
3*Frame2*Table2*Frame1*Table1*DropDownList1(Phase Material)" '( 0))
(cx-gui-do cx-activate-item "phase-domain-
3*Frame2*Table2*Frame1*Table1*DropDownList1(Phase Material)")
(cx-gui-do cx-set-text-entry "phase-domain-3*TextEntry1(Name)" "water")
(cx-gui-do cx-activate-item "phase-domain-
3*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-list-selections
"Phases*Frame1*Table1*Frame1*List1(Phases)" '( 1))
(cx-gui-do cx-activate-item
"Phases*Frame1*Table1*Frame1*List1(Phases)")
(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton7(Boundary
Conditions)")
(cx-gui-do cx-set-list-selections "Boundary
Conditions*Frame1*Table1*Frame1*List1(Zone)" '( 0))
(cx-gui-do cx-activate-item "Boundary
Conditions*Frame1*Table1*Frame1*List1(Zone)")
(cx-gui-do cx-activate-item "Boundary
Conditions*Frame1*Table1*Frame2*Table2*Frame4*Table4*Frame1*ButtonBox1*
PushButton1(Edit)")
(cx-gui-do cx-set-toggle-button "wall-6-
1*Frame4*Frame3(Momentum)*Frame1*Table1*Frame2*Frame1*Frame1*Table1*Fra
me1(Shear Condition)*ToggleBox1(Shear Condition)*Specified Shear" #f)
(cx-gui-do cx-activate-item "wall-6-
1*Frame4*Frame3(Momentum)*Frame1*Table1*Frame2*Frame1*Frame1*Table1*Fra
me1(Shear Condition)*ToggleBox1(Shear Condition)*Specified Shear")
(cx-gui-do cx-activate-item "wall-6-1*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-list-selections "Boundary
Conditions*Frame1*Table1*Frame1*List1(Zone)" '( 2))
```

```
(cx-gui-do cx-activate-item "Boundary
Conditions*Frame1*Table1*Frame1*List1(Zone)")
(cx-gui-do cx-activate-item "Boundary
Conditions*Frame1*Table1*Frame2*Table2*Frame4*Table4*Frame1*ButtonBox1*
PushButton1(Edit)")
(cx-gui-do cx-set-toggle-button "wall-4-
1*Frame4*Frame3(Momentum)*Frame1*Table1*Frame2*Frame1*Frame1*Table1*Fra
me1(Shear Condition)*ToggleBox1(Shear Condition)*Specified Shear" #f)
(cx-gui-do cx-activate-item "wall-4-
1*Frame4*Frame3(Momentum)*Frame1*Table1*Frame2*Frame1*Frame1*Table1*Fra
me1(Shear Condition)*ToggleBox1(Shear Condition)*Specified Shear")
(cx-gui-do cx-activate-item "wall-4-1*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-list-selections "Boundary
Conditions*Frame1*Table1*Frame1*List1(Zone)" '( 4))
(cx-gui-do cx-activate-item "Boundary
Conditions*Frame1*Table1*Frame1*List1(Zone)")
(cx-gui-do cx-activate-item "Boundary
Conditions*Frame1*Table1*Frame2*Table2*Frame4*Table4*Frame1*ButtonBox1*
PushButton1(Edit)")
(cx-gui-do cx-set-toggle-button "pressure-inlet-7-
1*Frame4*Frame8(Multiphase)*Frame1*Table1*CheckButton2(Open Channel)"
#f)
(cx-gui-do cx-activate-item "pressure-inlet-7-
1*Frame4*Frame8(Multiphase)*Frame1*Table1*CheckButton2(Open Channel)")
(cx-gui-do cx-set-real-entry-list "pressure-inlet-7-
1*Frame4*Frame8(Multiphase)*Frame1*Table1*RealEntry6(Free Surface
Level)" '( 0.5))
(cx-gui-do cx-set-real-entry-list "pressure-inlet-7-
1*Frame4*Frame8(Multiphase)*Frame1*Table1*RealEntry8(Bottom Level)" '(
-5))
(cx-gui-do cx-set-real-entry-list "pressure-inlet-7-
1*Frame4*Frame8(Multiphase)*Frame1*Table1*Frame9*Table9*RealEntry2(Velo
city Magnitude)" '( 10))
(cx-gui-do cx-activate-item "pressure-inlet-7-
1*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton7(Boundary
Conditions)")
(cx-gui-do cx-set-list-selections "Boundary
Conditions*Frame1*Table1*Frame1*List1(Zone)" '( 5))
(cx-gui-do cx-activate-item "Boundary
Conditions*Frame1*Table1*Frame1*List1(Zone)")
(cx-gui-do cx-activate-item "Boundary
Conditions*Frame1*Table1*Frame2*Table2*Frame4*Table4*Frame1*ButtonBox1*
PushButton1(Edit)")
(cx-gui-do cx-set-toggle-button "pressure-outlet-5-
1*Frame4*Frame8(Multiphase)*Frame1*Table1*CheckButton2(Open Channel)"
#f)
(cx-gui-do cx-activate-item "pressure-outlet-5-
1*Frame4*Frame8(Multiphase)*Frame1*Table1*CheckButton2(Open Channel)")
(cx-gui-do cx-set-real-entry-list "pressure-outlet-5-
1*Frame4*Frame8(Multiphase)*Frame1*Table1*RealEntry5(Free Surface
Level)" '( 0.5))
(cx-gui-do cx-set-real-entry-list "pressure-outlet-5-
1*Frame4*Frame8(Multiphase)*Frame1*Table1*RealEntry6(Bottom Level)" '(
-5))
(cx-gui-do cx-activate-item "pressure-outlet-5-
1*PanelButtons*PushButton1(OK)")
```

```
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton10(Reference Values)")
(cx-gui-do cx-set-list-selections "Reference
Values*DropDownList1(Compute from)" '( 5))
(cx-gui-do cx-activate-item "Reference Values*DropDownList1(Compute
from)")
(cx-gui-do cx-set-real-entry-list "Reference Values*Frame2(Reference
Values)*Table2(Reference Values)*RealEntry2(Density)" '( 998.2))
(cx-gui-do cx-activate-item "Reference Values*Frame2(Reference
Values)*Table2(Reference Values)*RealEntry2(Density)")
(cx-gui-do cx-set-real-entry-list "Reference Values*Frame2(Reference
Values)*Table2(Reference Values)*RealEntry9(Viscosity)" '( 0.001003))
(cx-gui-do cx-activate-item "Reference Values*Frame2(Reference
Values)*Table2(Reference Values)*RealEntry9(Viscosity)")
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton12(Solution Methods)")
(cx-gui-do cx-set-list-selections "Solution
Methods*Frame1*Table1*Frame2(Pressure-Velocity
Coupling)*Table2(Pressure-Velocity Coupling)*DropDownList1(Scheme)" '(
0))
(cx-gui-do cx-activate-item "Solution
Methods*Frame1*Table1*Frame2(Pressure-Velocity
Coupling)*Table2(Pressure-Velocity Coupling)*DropDownList1(Scheme)")
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton14(Monitors)")
(cx-gui-do cx-set-list-selections
"Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force
Monitors)" '( 2))
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force
Monitors)")
(cx-gui-do cx-activate-item "Monitors*Frame1*Table1*PushButton2(Edit)")
(cx-gui-do cx-set-toggle-button
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton1(Print to Console)" #f)
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton1(Print to Console)")
(cx-gui-do cx-set-toggle-button
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton2(Plot)" #f)
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton2(Plot)")
(cx-gui-do cx-set-list-selections
"Monitors*Frame1*Table1*Frame2*Table2*Frame1*List1(Wall Zones)" '( 2))
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame2*Table2*Frame1*List1(Wall Zones)")
(cx-gui-do cx-set-toggle-button
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton4(Write)" #f)
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton4(Write)")
(cx-gui-do cx-set-text-entry
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*F
```

```
rame5*Table5*TextEntry2(File Name)"
"C:\Travis\Thesis\Free_Surface_Hydrofoil\cd-history")
(cx-gui-do cx-activate-item "Monitors*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-list-selections
"Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force
Monitors)" '( 3))
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force
Monitors)")
(cx-gui-do cx-activate-item "Monitors*Frame1*Table1*PushButton2(Edit)")
(cx-gui-do cx-set-list-selections
"Monitors*Frame1*Table1*Frame2*Table2*Frame1*List1(Wall Zones)" '( 2))
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame2*Table2*Frame1*List1(Wall Zones)")
(cx-gui-do cx-set-toggle-button
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton1(Print to Console)" #f)
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton1(Print to Console)")
(cx-gui-do cx-set-toggle-button
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton2(Plot)" #f)
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton2(Plot)")
(cx-gui-do cx-set-toggle-button
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton4(Write)" #f)
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*C
heckButton4(Write)")
(cx-gui-do cx-set-text-entry
"Monitors*Frame1*Table1*Frame1*Table1*Frame1(Options)*Table1(Options)*F
rame5*Table5*TextEntry2(File Name)"
"C:\Travis\Thesis\Free_Surface_Hydrofoil\cl-history")
(cx-gui-do cx-activate-item "Monitors*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-list-selections
"Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force
Monitors)" '( 0))
(cx-gui-do cx-activate-item
"Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force
Monitors)")
(cx-gui-do cx-activate-item "Monitors*Frame1*Table1*PushButton2(Edit)")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton10" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton10")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton16" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton16")
```

```
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton22" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton22")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton28" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton28")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton8" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton8")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton14" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton14")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton20" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton20")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton26" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton26")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton34" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton34")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton40" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton40")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton38" #t)
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton38")
(cx-gui-do cx-set-toggle-button "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton32" #t)
```

```
(cx-gui-do cx-activate-item "Residual
Monitors*Frame1*Table1*Frame2*Table2*Frame1(Equations)*Table1(Equations
)*CheckButton32")
(cx-gui-do cx-activate-item "Residual
Monitors*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton15(Solution Initialization)")
(cx-gui-do cx-set-list-selections "Solution
Initialization*Frame1*Table1*DropDownList1(Compute from)" '( 5))
(cx-gui-do cx-activate-item "Solution
Initialization*Frame1*Table1*DropDownList1(Compute from)")
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton15(Solution Initialization)")
(cx-gui-do cx-set-list-selections "Solution
Initialization*Frame1*Table1*DropDownList3(Open channel Initialization
Method)" '( 1))
(cx-gui-do cx-activate-item "Solution
Initialization*Frame1*Table1*DropDownList3(Open channel Initialization
Method)")
(cx-gui-do cx-activate-item "Solution
Initialization*Frame1*Table1*Frame6*ButtonBox6*PushButton1(Initialize)"
)
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton2(General)")
(cx-gui-do cx-set-toggle-button
"General*Frame1*Table1*Frame2(Solver)*Table2(Solver)*Frame5(Time)*Butto
nBox5(Time)*Transient" #f)
(cx-gui-do cx-activate-item
"General*Frame1*Table1*Frame2(Solver)*Table2(Solver)*Frame5(Time)*Butto
nBox5(Time)*Transient")
(cx-gui-do cx-set-toggle-button
"General*Frame1*Table1*Frame2(Solver)*Table2(Solver)*Frame5(Time)*Butto
nBox5(Time)*Steady" #f)
(cx-gui-do cx-activate-item
"General*Frame1*Table1*Frame2(Solver)*Table2(Solver)*Frame5(Time)*Butto
nBox5(Time)*Steady")
(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton17(Run
Calculation)")
(cx-gui-do cx-set-integer-entry "Run
Calculation*Frame1*Table1*IntegerEntry8(Number of Iterations)" 1000)
(cx-gui-do cx-activate-item "Run
Calculation*Frame1*Table1*IntegerEntry8(Number of Iterations)")
(cx-gui-do cx-activate-item "Run
Calculation*Frame1*Table1*PushButton18(Calculate)")
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton19(Graphics and Animations)")
(cx-gui-do cx-set-list-selections "Graphics and
Animations*Frame1*Table1*Frame1*List1(Graphics)" '( 1))
(cx-gui-do cx-activate-item "Graphics and
Animations*Frame1*Table1*Frame1*List1(Graphics)")
(cx-gui-do cx-activate-item "Graphics and
Animations*Frame1*Table1*PushButton2(Set Up)")
(cx-gui-do cx-set-list-selections
"Contours*Frame2*Table2*DropDownList3(Phase)" '( 0))
(cx-gui-do cx-activate-item
"Contours*Frame2*Table2*DropDownList3(Phase)")
```

```
(cx-gui-do cx-set-toggle-button
"Contours*Frame1*Frame1(Options)*ToggleBox1(Options)*CheckButton1(Fille
d)" #f)
(cx-gui-do cx-activate-item
"Contours*Frame1*Frame1(Options)*ToggleBox1(Options)*CheckButton1(Fille
d)")
(cx-gui-do cx-set-integer-entry "Contours*Frame1*IntegerEntry3(Levels)"
100)
(cx-gui-do cx-activate-item "Contours*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-position "Contours" '(x 465 y 224))
(cx-gui-do cx-set-position "Contours" '(x 386 y 224))
(cx-gui-do cx-set-position "Contours" '(x 286 y 213))
(cx-gui-do cx-set-position "Contours" '(x 230 y 204))
(cx-gui-do cx-set-position "Contours" '(x 188 y 198))
(cx-gui-do cx-set-position "Contours" '(x 152 y 194))
(cx-gui-do cx-set-position "Contours" '(x 127 y 189))
(cx-gui-do cx-set-position "Contours" '(x 114 y 189))
(cx-gui-do cx-set-position "Contours" '(x 110 y 187))
(cx-gui-do cx-set-position "Contours" '(x 102 y 186))
(cx-gui-do cx-set-position "Contours" '(x 97 y 185))
(cx-use-window-id 3)
(cx-set-camera '(0.486008 0.00040853 4.83628) '(0.486008 0.00040853
0.00187743) '(-1.26042e-009 1 2.68726e-013) 1.93376 1.93376
"perspective")

(cx-gui-do cx-activate-item "ToolBar*General Tools*savepicture")
(cx-gui-do cx-activate-item "Save
Picture*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-text-entry "Select File*FilterText"
"c:\travis\thesis\free_surface_hydrofoil\pictures\*")
(cx-gui-do cx-activate-item "Select File*Apply")
(cx-gui-do cx-set-text-entry "Select File*Text" "Pressure.jpg")
(cx-gui-do cx-activate-item "Select File*OK")
(cx-gui-do cx-activate-item "Warning*OK")
(cx-gui-do cx-activate-item "Save
Picture*PanelButtons*PushButton2(Cancel)")
(cx-gui-do cx-set-list-selections
"Contours*Frame2*Table2*DropDownList1(Contours of)" '( 11))
(cx-gui-do cx-activate-item
"Contours*Frame2*Table2*DropDownList1(Contours of)")
(cx-gui-do cx-set-list-selections
"Contours*Frame2*Table2*DropDownList3(Phase)" '( 0))
(cx-gui-do cx-activate-item
"Contours*Frame2*Table2*DropDownList3(Phase)")
(cx-gui-do cx-activate-item "Contours*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-activate-item "ToolBar*General Tools*savepicture")
(cx-gui-do cx-activate-item "Save
Picture*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-text-entry "Select File*Text" "Phase.jpg")
(cx-gui-do cx-activate-item "Select File*OK")
(cx-gui-do cx-activate-item "Warning*OK")
(cx-gui-do cx-activate-item "Save
Picture*PanelButtons*PushButton2(Cancel)")
(cx-gui-do cx-set-list-selections
"Contours*Frame2*Table2*DropDownList1(Contours of)" '( 2))
(cx-gui-do cx-activate-item
"Contours*Frame2*Table2*DropDownList1(Contours of)")
```

```
(cx-gui-do cx-set-list-selections
"Contours*Frame2*Table2*DropDownList3(Phase)" '( 0))
(cx-gui-do cx-activate-item
"Contours*Frame2*Table2*DropDownList3(Phase)")
(cx-gui-do cx-activate-item "Contours*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-activate-item "ToolBar*General Tools*savepicture")
(cx-gui-do cx-activate-item "Save
Picture*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-text-entry "Select File*Text" "Velocity.jpg")
(cx-gui-do cx-activate-item "Select File*OK")
(cx-gui-do cx-activate-item "Warning*OK")
(cx-gui-do cx-activate-item "Save
Picture*PanelButtons*PushButton2(Cancel)")
(cx-gui-do cx-activate-item
"Contours*PanelButtons*PushButton2(Cancel)")
(cx-gui-do cx-activate-item
"NavigationPane*Frame1*PushButton20(Plots)")
(cx-gui-do cx-set-list-selections
"Plots*Frame1*Table1*Frame1*List1(Plots)" '( 0))
(cx-gui-do cx-activate-item "Plots*Frame1*Table1*Frame1*List1(Plots)")
(cx-gui-do cx-activate-item "Plots*Frame1*Table1*PushButton2(Set Up)")
(cx-gui-do cx-set-list-selections "Solution XY
Plot*Frame8*Table8*DropDownList3(Phase)" '( 0))
(cx-gui-do cx-activate-item "Solution XY
Plot*Frame8*Table8*DropDownList3(Phase)")
(cx-gui-do cx-set-list-selections "Solution XY
Plot*Frame7*Table7*DropDownList3(Phase)" '( 0))
(cx-gui-do cx-activate-item "Solution XY
Plot*Frame7*Table7*DropDownList3(Phase)")
(cx-gui-do cx-set-list-selections "Solution XY
Plot*Frame9*Frame1*List1(Surfaces)" '( 3))
(cx-gui-do cx-activate-item "Solution XY
Plot*Frame9*Frame1*List1(Surfaces)")
(cx-gui-do cx-activate-item "Solution XY
Plot*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-activate-item "ToolBar*General Tools*savepicture")
(cx-gui-do cx-activate-item "Save
Picture*PanelButtons*PushButton1(OK)")
(cx-gui-do cx-set-text-entry "Select File*Text" "PressurePlot.jpg")
(cx-gui-do cx-activate-item "Select File*OK")
(cx-gui-do cx-activate-item "Warning*OK")
(cx-gui-do cx-activate-item "Save
Picture*PanelButtons*PushButton2(Cancel)")
(cx-gui-do cx-activate-item "Solution XY
Plot*PanelButtons*PushButton2(Cancel)")
(cx-gui-do cx-activate-item "MenuBar*WriteSubMenu*Stop Transcript")
(cx-gui-do cx-set-text-entry "Select File*FilterText"
"c:\travis\thesis\free_surface_hydrofoil\*")
(cx-gui-do cx-activate-item "Select File*Apply")
(cx-gui-do cx-set-text-entry "Select File*Text" "trans.jou")
(cx-gui-do cx-activate-item "Select File*OK")
(%cx-warning-dialog "OK to quit?" #f)
(cx-gui-do cx-activate-item "Warning*OK")
/exit y
```

Vita

William Travis Cocke

Date of Birth	March 23, 1989
Place of Birth	Nashville, TN
Degrees	B.S. Mechanical Engineering, May 2012 M.S. Mechanical Engineering, May 2012 Minor in Computer Science, May 2012

May 2012