İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

GENETIC ALGORITHM AND RESIDUAL CORRECTION METHOD FOR INVERSE DESIGN OF AIRFOILS

M. Sc. Thesis by

Bülent TUTKUN, B. Sc.

Department : Space Engineering

Programme: Space Engineering

MAY 2003

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

GENETIC ALGORITHM AND RESIDUAL CORRECTION METHOD FOR INVERSE DESIGN OF AIRFOILS

M.Sc. Thesis by
Bülent TUTKUN, B.Sc.

(511001208)

Date of submission : 5 May 2003

Date of defence examination: 27 May 2003

Supervisor (Chairman): Prof. Dr. Rıdvan MERİÇ

Members of the Examining Committee Prof. Dr. Süleyman TOLUN

Prof. Dr. A. Rüstem ASLAN

MAY 2003

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GENETİK ALGORİTMA VE ARTIK DÜZELTME YÖNTEMİYLE PROFİL TERS TASARIM

YÜKSEK LİSANS TEZİ
Uzay Müh. Bülent TUTKUN
(511001208)

Tezin Enstitüye Verildiği Tarih : 5 Mayıs 2003

Tezin Savunulduğu Tarih : 27 Mayıs 2003

Tez Danışmanı : Prof. Dr. R. Alsan MERİÇ

Diğer Jüri Üyeleri Prof. Dr. Süleyman TOLUN

Prof. Dr. A. Rüstem ASLAN

MAYIS 2003

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Dr. R Alsan MERİÇ for his valuable support and encouragement during this study.

Of course, I would also like to thank to my family. Their support is always with me whenever I need it.

And I would like to send special thanks to a special person for me.

May 2003, ISTANBUL Bülent TUTKUN

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

L	: Lift
D	: Drag
N	: Component of R(normal to the airfoil chord)
A	: Component of R(Tangential to the airfoil chord)
M	: Moment
R	: Force
$\mathbf{V}_\infty$	: Free stream velocity
c	: Chord of airfoil
$\mathbf{c_l}$	: Lift coefficient
$\mathbf{c_d}$	: Drag Coefficient
$\mathbf{c_p}$	: Pressure coefficient
$\mathbf{Tc_p}$	: Target pressure coefficient
$\mathbf{c_m}$	: Moment coefficient
p	: Pressure
u	: Velocity in horizontal direction
v	: Velocity in vertical direction
$\mathbf{u}_{sij}$, $\mathbf{v}_{sij}$	: Velocity components at the midpoint of panel i induced by a source of unit strength at the midpoint of panel j.
$\mathbf{u}_{vij}$, $\mathbf{v}_{vij}$	: Velocity components at the midpoint of panel i induced by a vortex of unit strength at the midpoint of panel j.
$\mathbf{u}^*, \mathbf{v}^*$	: Local velocities
q	: Source strength
s	: arc-length coordinate
$\mathbf{r_{ij}}$	: Distance from the midpoint of panel i to the j_{th} node
τ	: Shear stress
α	: Angle of attack
ρ_∞	: Free stream density
$\mathbf{q}_\infty$	: $\frac{1}{2}\rho_\infty V_\infty^2$
ϕ	: potential function
ϕ_∞	: Free stream potential function
ϕ_s	: Source distribution potential function
ϕ_v	: Vortex distribution potential function
ψ	: stream function

γ : Vortex strength

θ : Panel angle

$\hat{\mathbf{n}}$: vector normal to the panel

$\hat{\mathbf{t}}$: vector tangential to the panel

$\mathbf{r}_{ij}$: Distance from the midpoint of panel i to the j_{th} node

β_{ij} : Angle subtended by the j_{th} panel at the midpoint of panel i

GENETİK ALGORİTMA VE ARTIK DÜZELTME YÖNTEMİYLE PROFİL TERS TASARIM

ÖZET

Hava-uzay alanında tasarımlanan kanat profil şekilleri performans ve ihtiyaçların karşılanması açısından hayati rol oynar. Bu nedenle bu konuda bir çok çaba harcanmaktadır. Yeni bir kanat profil şekli tasarlarken, araştırmacılar genellikle optimizasyon veya ters tasarım tekniklerini kullanırlar. Optimizasyonda taşıma, sürükleme ve moment gibi profile ait bazı parametreler minimize veya maksimize edilmeye çalışılır. Halbuki, ters tasarımda ise verilen bir parametre için, (bu genellikle basınç dağılımıdır) o parametreyi sağlayan profil şekli bulunmaya çalışılır.

Bu çalışmada, verilen hedef değerleri sağlayan bir profil geometrisi iki farklı ters tasarım yöntemi ile elde edilmiştir. İki yöntemin sahip olduğu algoritmalarda farklıdır. İlk yöntem genetik algoritma kullanmaktadır. Bu yöntemde ayrıca, şekil parametrelerini azaltmak için B-spline eğrilerinden yararlanılmıştır. Bu yöntemin amacı $\sum_{i=1}^{n} -\left[(Tc_{p_i})^2 - (c_{p_i})^2\right]$ değerini maksimize etmektir. Buradaki Tc_{p_i} hedef basınç dağılımı ve c_{p_i} tasarlanan basınç dağılımıdır. Tasarlanan profil geometrisinin analizinde, Smith-Hess panel yöntemi kullanılmıştır.

İkinci ters tasarım yönteminde, artık düzeltme algoritması kullanılmıştır. Belli bir profil geometrisi ile başlayarak (NACA 0012), her adımda hesaplanan ΔY'lerin yardımıyla hedef profil geometrisine ulaşılır. ΔY'ler $A\Delta Y + B\frac{d\Delta Y}{dx} + C\frac{d^2\Delta Y}{dx^2} = V_t^2 - V^2$ diferansiyel denklemi kullanılarak hesaplanırlar. İlk olarak bu diferansiyel denklem sonlu farklar yaklaşımı kullanılarak ayrıklaştırılır. Daha sonra, elde edilen üç-bant katsayılar matrisi Thomas algoritmasının yardımıyla çözülür ve ΔY'ler elde edilir. Bu yöntemde birincisi gibi analiz için Smith-Hess panel yöntemini kullanır.

GENETIC ALGORITHM AND RESIDUAL CORRECTION METHOD FOR INVERSE DESIGN OF AIRFOILS

SUMMARY

In the field of aerospace, designed airfoil shapes play a crucial role in terms of performance and meeting the requirements. So, many efforts are put on this subject in aerospace. While designing a new airfoil shape, researchers generally use optimization or inverse design techniques. In optimization, some parameters (lift, drag, moment, etc.) of the airfoil are tried to be minimized or maximized. However, in inverse design, an airfoil shape is designed for a given parameter (generally pressure distribution).

In this work, two inverse design method with different algorithms are used to design an airfoil geometry that fits to given target values. First method utilizes a genetic algorithm which is a search method. In the first method, also B-spline curves are used to decrease shape parameters. This method's purpose is to maximize the $\sum_{i=1}^{n} -(c_{p_i} - Tc_{p_i})^2$. Where Tc_{p_i} is the target pressure distribution and c_{p_i} is the designed pressure distribution. To analyze the designed airfoil geometries, Smith-Hess panel method is used.

In second inverse design method, residual correction algorithm is utilized. Starting with an initial airfoil geometry (NACA 0012), target airfoil geometry is reached with the help of ΔY's coming from the $A\Delta Y + B\frac{d\Delta Y}{dx} + C\frac{d^2\Delta Y}{dx^2} = V_t^2 - V^2$. First, this differential equation is discritized with finite differences. Then obtained tri-diagonal coefficient matrix is solved with the Thomas Algorithm to give ΔY's. This method also uses Smith-Hess panel method to analyze the airfoils.

1. INTRODUCTION

With the advent of successful powered flight at the turn of the twentieth century, the importance of aerodynamics rose suddenly. So, interest grew in the understanding of the aerodynamic action of such lifting surfaces as fixed wings on airplanes and, later, rotors on helicopters [1]. Consider a wing as drawn in perspective in Figure (1.1).

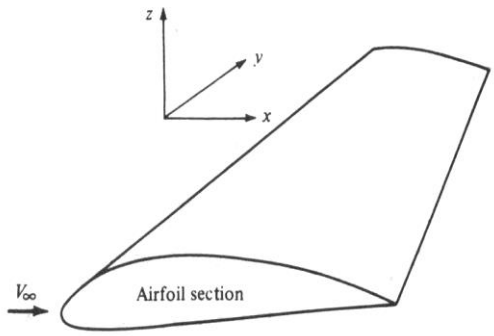


Figure 1.1 Definition of an airfoil

The wing extends in the y direction. The free stream velocity V_∞ is parallel to the xz plane. Any section of the wing cut by a plane parallel to the xz plane is called an airfoil. The lift and moments on the airfoil are due mainly to the pressure distribution.

The first Patented airfoil shapes were developed by Horatio F. Phillips in 1884 [1]. Clearly, in the early days of powered flight, airfoil design was basically customized and personalized. Just mentioned above, the aerodynamic forces and moments on the airfoil are due to only two basic sources:

a) Pressure distribution over the body surface

b) Shear stress distribution over the body surface

The net effect of the p and τ distributions integrated over the complete body surface is a resultant aerodynamic force R and moment M on the body (Figure 1.2)

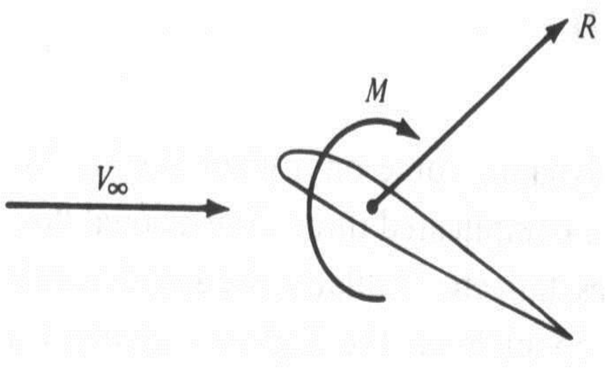


Figure 1.2 Aerodynamic force and moment on the body

Then the resultant R can be split into components. (Figure 1.3)

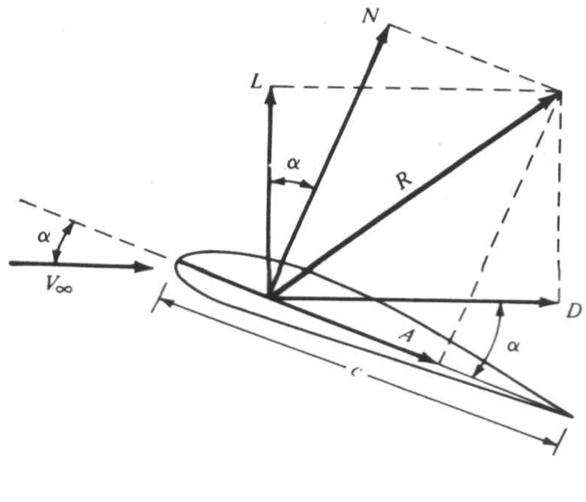


Figure 1.3 Aerodynamic force and its components

The angle of attack α is defined as the angle between c and V_∞. From geometrical relations:

$$\begin{aligned} L &= N\cos\alpha - A\sin\alpha \\ D &= N\sin\alpha + A\cos\alpha \end{aligned} \tag{1.1}$$

As this formulation shows, L and D values of an airfoil are determined by directly pressure distribution. If we use dimensionless quantities, they are defined as follows:
Lift coefficient;

$$c_l = \frac{L}{q_\infty c} \tag{1.2}$$

Drag coefficient;

$$c_d = \frac{D}{q_\infty c} \tag{1.3}$$

Pressure coefficient

$$c_p = \frac{p - p_\infty}{q_\infty} \tag{1.4}$$

Moment coefficient

$$c_m = \frac{M}{q_\infty c^2} \tag{1.5}$$

$$\text{where } q_\infty = \frac{1}{2}\rho_\infty V_\infty^2 \tag{1.6}$$

Since lift and moments come from the pressure distribution on the airfoil, to create the required lift or moment, airfoil geometry must form a specified pressure distribution. Therefore, for many years researchers have studied hard on airfoil design techniques.
The aerodynamic design of aircraft components is often carried by means of one of the following four approaches:

1. ''Cut and Try'' analysis
2. Indirect Methods
3. Optimization Techniques
4. Inverse Design Techniques

This work includes mainly inverse design and partly optimization.
First part of this work is composed of finding suitable airfoil shape, which gives a specified pressure distribution by the help of an optimization algorithm. This

constitutes an inverse design problem. In solution of this problem, optimization with a genetic algorithm is used. That is, inverse problem is transformed into an optimization problem. With the help of genetic algorithm, the optimum set of 20 control points is found. Then this set is used to form the airfoil shape by utilizing an algorithm for drawing a B-spline curve [2].

Genetic algorithms are search methods used in recent years. They differ in conception from other search methods, including traditional optimization methods and other stochastic search methods. The basic difference is that while other methods always process single points in the search space, genetic algorithms maintain a population of potential solutions [3].

Shape optimization based on genetic algorithm [4], or based on evolutionary algorithms in general, is a relatively young and potential field of research. The interest towards researching evolutionary shape optimization techniques appears to be just started to grow, rather than reached a stable and mature state.

Currently the most popular application area of genetic algorithms-based shape optimization seems to be the shape optimization in connection with computational fluid dynamics (CFD), especially aerodynamic shape optimization in the field of aircraft design, for example [5-12].

Using B-splines in an optimization problem is very helpful in a way that it lessens the design parameters. As a result of this, cost of the algorithm is also lessened. This kind of application of B-splines may be seen in literature, for example [13,14].

Another part of this work includes an inverse design method in which a residual correction algorithm is used. With this algorithm, an airfoil shape that gives the target pressure distribution is reached. Inverse design method is a very popular method in aerospace, for example [15-18].

2. GENETIC ALGORITHMS

Genetic algorithms constitute a class of search methods especially suited for solving complex optimization problems [3]. Search algorithms in general consist of systematically walking through the search space of possible solutions until an acceptable solution is found. Genetic algorithms transpose the notions of natural evolution to the world of computers, and imitate natural evolution. They were initially introduced by John Holland [4] for explaining the adaptive processes of natural systems and for creating new artificial systems that work on similar bases. In nature, new organisms adapted to their environment develop through evolution. Genetic algorithms evolve solutions to the given problem in a similar way. They maintain a collection of solutions-—a population of individuals-—and so perform a multidirectional search. The individuals are represented by chromosomes composed of genes. Genetic algorithms operate on the chromosomes, which represent the inheritable properties of the individuals. By analogy with Nature, through selection the fit individuals-—potential solutions to the optimization problem-—live to reproduce, and the weak individuals, which are not so fit, die off. New individuals are created from one or two parents by mutation and crossover, respectively. They replace old individuals in the population and they are usually similar to their parents. In other words, in a new generation there will be individuals that resemble the fit individuals from the previous generation. The individuals survive if they are fitted to the given environment.

In table 1 the analogy of terms between nature and artificial evolutionary systems in general.

Table 2.1 The correspondence of terms between natural and artificial evolution

Nature	**Evolutionary computation**
Individual	Solution to a problem
Population	Collection of solution
Fitness	Quality of solution
Chromosome	Representation of a solution
Gene	Part of representation of a solution
Crossover	Binary search operator
Mutation	Unary search operator
Reproduction	Reuse of solutions
Selection	Keeping good sub-solutions

Evolution is an emergent property of artificial evolutionary systems. The computer is only told to (1) maintain a population of solutions, (2) allow the fitter individuals to reproduce, and (3) let the less fit individuals die off. The new individuals inherit the properties of their parents, and the fitter ones survive for the next generation. The final solutions will be much better than their ancestors from the first generation.

This evolution is directed by fitness. The evolutionary search is conducted towards better regions of the search space on the basis of the fitness measure. Each solution in a population is evaluated based on how well it solves the given problem. Correspondingly, each member of the population is assigned a fitness value. Genetic algorithms use a separate search space and solution space. The search space is the space of coded solutions, i.e. genotypes or chromosomes consisting of genes. More exactly, a genotype may consist of several chromosomes, but in most practical applications genotypes are made of one chromosome. The solution space is the space of actual solutions, i.e. phenotypes. Any genotype must be transformed into the corresponding phenotype before its fitness is evaluated.

2.1 The outline of a genetic algorithm

When solving a problem using genetic algorithms, first a proper representation and fitness measure must be designed. Many representations are possible, and will work. Some are better than the others, however. Devising the termination criterion should be the next step. The termination criterion usually allows at most some predefined number of iterations and verifies whether an acceptable solution has been found. The genetic algorithm then works as follows (also shown in Figure 2.1):

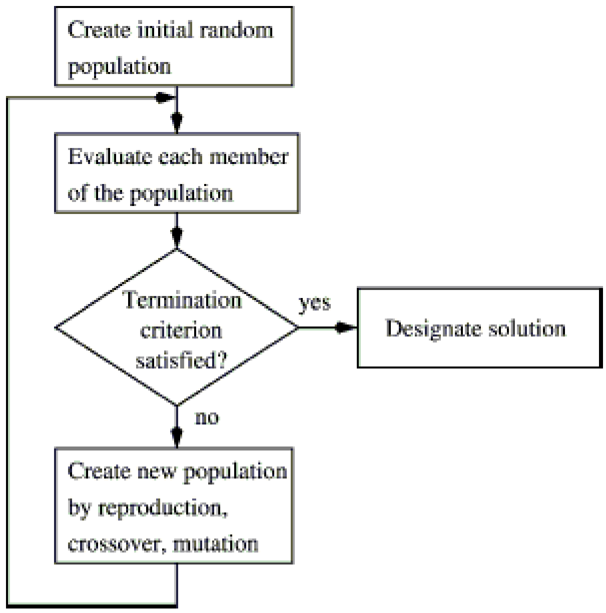


Figure 2.1 Genetic algorithm flowchart

1. The initial population is filled with individuals that are generally created at random. Sometimes, the individuals in the initial population are the solutions found by some method determined by the problem domain. In this case, the scope of the genetic algorithm is to obtain more accurate solutions.

2. Each individual in the current population is evaluated using the fitness measure.

3. If the termination criterion is met, the best solution is returned.

4. From the current population, individuals are selected based on the previously computed fitness values. A new population is formed by applying the genetic

operators (reproduction, crossover, mutation) to these individuals. The selected individuals are called parents and the resulting individuals offspring. Implementations of genetic algorithms differ in the way of constructing the new population. Some implementations extend the current population by adding the new individuals and then create the new population by omitting the least fit individuals. Other implementations create a separate population of new individuals by applying the genetic operators. Moreover, there are genetic algorithms that do not use generations at all, but continuous replacement.

5. Actions starting from step 2 are repeated until the termination criterion is satisfied. An iteration is called generation.

2.2 Genetic operators

In each generation, the genetic operators are applied to selected individuals from the current population in order to create a new population. Generally, the three main genetic operators of reproduction, crossover and mutation are employed. By using different probabilities for applying these operators, the speed of convergence can be controlled. Crossover and mutation operators must be carefully designed, since their choice highly contributes to the performance of the whole genetic algorithm.

Reproduction: A part of the new population can be created by simply copying without change selected individuals from the present population. This gives the possibility of survival for already developed fit solutions.

Crossover: New individuals are generally created as offspring of two parents (as such, crossover being a binary operator). One or more so-called crossover points are selected (usually at random) within the chromosome of each parent, at the same place in each. The parts delimited by the crossover points are then interchanged between the parents.

Mutation: A new individual is created by making modifications to one selected individual. The modifications can consist of changing one or more values in the representation or in adding/deleting parts of the representation. In genetic algorithms mutation is a source of variability, and is applied in addition to crossover and reproduction. At different stages of evolution, one may use different mutation operators. At the beginning mutation operators resulting in bigger jumps in the search space might be preferred. Later on, when the solution is close by, a mutation operator leading to slighter shifts in the search space could be favored.

2.3 Fitness assignment

The probability of survival of any individual is determined by its fitness: through evolution the fitter individuals overtake the less fit ones. In order to evolve good solutions, the fitness assigned to a solution must directly reflect its 'goodness', i.e. the fitness function must indicate how well a solution fulfills the requirements of the given problem. Fitness assignment can be performed in several different ways:

- We define a fitness function and incorporate it in the genetic algorithm. When evaluating any individual, this fitness function is computed for the individual.
- Fitness evaluation is performed by dedicated separate analysis software. In such cases evaluation can be time-consuming, thus slowing down the whole evolutionary algorithm.
- Sometimes there is no explicit fitness function, but a human evaluator assigns a fitness value to the solutions presented to him/her.
- Fitness can be assigned by comparing the individuals in the current population.

2.4 Selection methods

Only selected individuals of a population are allowed to have offspring. Selection is based on fitness: individuals with better fitness values are picked more frequently than individuals with worse fitness values. The most commonly used selection schemes:

Fitness-proportional selection: When using this selection method, a solution has a probability of selection directly proportional to its fitness. The mechanism that allows fitness proportional selection is similar to a roulette wheel that is partitioned into slices. Each individual has a share directly proportional to its fitness. When the roulette wheel is rotated, an individual has a chance of being selected corresponding to its share.

Ranked selection: The problem of fitness-proportional selection is that it is directly based on fitness. In most cases, we cannot define an accurate measure of goodness of a solution, so the assigned fitness value does not express exactly the quality of a solution. Still, an individual with better fitness value is a better individual. In rank based selection, the individuals are ordered according to their fitness. The individuals are then selected with a probability based on some linear function of their rank.

Tournament selection: In tournament selection, a set of n individuals are chosen from the population at random. Then the best of the pool is selected. For n =1, the method is equivalent to random selection. The higher is the value of n, the more directed the selection is towards better individuals.

3. B-SPLINE CURVES

Formula for a cubic B-spline in terms of parametric equations whose parameter is u. Given the points $p_i = (x_i, y_i),\ \ i = 0,1,....,n$, the cubic B-spline for the interval $(p_i, p_{i+1}),\ \ i = 1,2,......., n-1$, is

$$B_i(u) = \sum_{k=-1}^{2} b_k p_{i+k}\ , \text{ where} \tag{3.1}$$

$$b_{-1} = \frac{(1-u)^3}{6},$$

$$b_0 = \frac{u^3}{2} - u^2 + \frac{2}{3}, \tag{3.2}$$

$$b_1 = -\frac{u^3}{2} + \frac{u^2}{2} + \frac{u}{2} + \frac{1}{6},$$

$$b_2 = \frac{u^3}{6}, \qquad 0 \le u \le 1$$

p_i refers to the point (x_i , y_i); it is a two-component vector. The coefficients, the b_k's, serve as a basis and do not change as we move from one set of points to the next. Observe that they can be considered weighting factors applied to the coordinates of a set of four points. The weighted sum, as u varies from 0 to 1, generates the B-spline curve.

If we write out the equations for x and y from Equation (3.2), we get

$$x_i(u) = \frac{1}{6}(1-u)^3 x_{i-1} + \frac{1}{6}(3u^3 - 6u^2 + 4)x_i$$
$$+ \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)x_{i+1} + \frac{1}{6}u^3 x_{i+2}$$

(3.3)

$$y_i(u) = \frac{1}{6}(1-u)^3 y_{i-1} + \frac{1}{6}(3u^3 - 6u^2 + 4)y_i$$
$$+ \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)y_{i+1} + \frac{1}{6}u^3 y_{i+2}$$

Note the notation here: $x_i(u)$ and $y_i(u)$ are functions of u and x_i, y_i are components of the point p. The u-cubics act as weighting factors on the coordinates of the four successive points to generate the curve. For example, at u = 0, the weights applied are 1/6, 2/3, 1/6 and 0. At u =1, they are 0, 1/6, 2/3, and 1/6. These values vary throughout the interval from u = 0 to u = 1.

Now we can examine two B-splines determined from a set of exactly four points. Figure 3.1a and 3.1b show the effect of varying just one of the points. As you would expect, when p_2 is moved upward and to the left, the curve tends to follow; in fact, it is pulled to the opposite side of p_1. You may be surprised to see that the curve is never very close to the two intermediate points, though it begins and ends at positions somewhat adjacent. It will be helpful to think of the curve generated from the defining equation for B_1 as associated with a curve that goes from near p_1 to p_2. It is also helpful to remember that points p_0, p_1, p_2, and p_3 are used to get B_1.

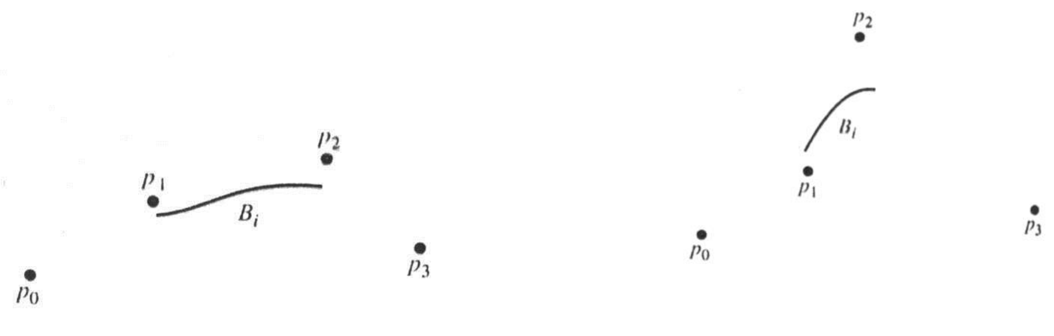


Figure 3.1 Effects of varying just one of the points on B-splines

Because a set of four points is required to generate only a portion of the B-spline, that associated with the two inner points, we must consider how to get the B-spline for more than four points as well as how to extend the curve into the region outside of the middle pair. For this we can march along one point at a time, forming new sets of four. We abandon the first of the old set when we add the new one.

The conditions that we want to impose on the B-spline are: continuity of the curve and its first and second derivatives. It turns out that the equations for the weighting factors (the u-polynomials, the b_k) are such that these requirements are met. Figure 3.2 shows how three successive parts of a B-spline might look.

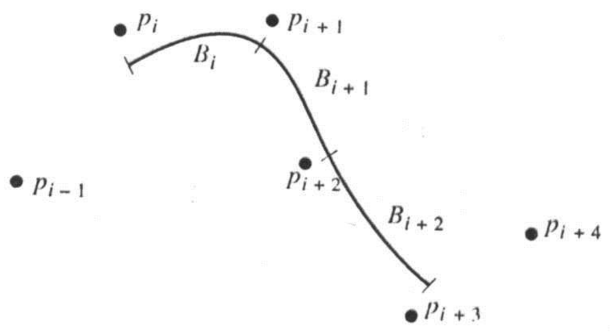


Figure 3.2 Successive B-splines joined together

We can summarize the properties of B-splines as follows:

1. B-splines are pieced together so they agree at their joints in three ways:

a. $B_i = B_{i+1}(0) = \dfrac{p_i + 4p_{i+1} + p_{i+2}}{6}$

b. $B_i^{'}(1) = B_{i+1}^{'}(0) = \dfrac{-p_i + p_{i+2}}{2}$

c. $B_i^{''}(1) = B_{i+1}^{''}(0) = p_i - 2p_{i+1} + p_{i+2}$

2. The portion of the curve determined by each group of four points is within the convex hull of these points.

Now we consider how to generate the ends of the joined B-spline. If we have points from p_0 to p_n, we already can construct B-splines B_1 through B_{n-2}. We need B_0 and

B_{n-1}. Our problem is that, using the procedure already defined, we would need additional points outside the domain of the given points.

First, we can add more points without artificiality by making the added points coincide with the given extreme points. If we add not just a single fictitious point at each end of the set, but two at each end, we will find that the new curves not only join properly with the portions already made, but start and end at the extreme points as we wanted. In summary, we add fictitious points p_{-2}, p_{-1}, p_{n+1}, and p_{n+2}, with the first two identical with p_0 and the last two identical with p_n.

The matrix formulation for cubic B-spline is:

$$B_i(u) = \frac{1}{6}\begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{bmatrix} = \frac{u^T M_b p}{6} \tag{3.4}$$

4. POTENTIAL FLOW AND PANEL METHOD

4.1 Governing Equations

Conservation of mass (continuity equation):

$$\nabla \cdot \mathbf{V} = 0 \tag{4.1}$$

For an irrotational flow:

$$\mathbf{V} = \nabla\phi \tag{4.2}$$

Therefore, for a flow that is both incompressible and irrotational equation 1 and 2 can be combined to yield

$\nabla \cdot (\nabla\phi) = 0$ then

$$\nabla^2\phi = 0 \tag{4.3}$$

Equation 3 is Laplace's equation

For a two dimensional incompressible flow, a stream function ψ can be defined such that

$$u = \frac{\partial\psi}{\partial y} \tag{4.4}$$

$$v = -\frac{\partial\psi}{\partial x} \tag{4.5}$$

The continuity equation, $\nabla \cdot \mathbf{V} = 0$, expressed in cartesian coordinates, is

$$\nabla \cdot \mathbf{V} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{4.6}$$

Substituting equation (4.4) and (4.5) into equation (4.6) we obtain

$$\frac{\partial}{\partial x}\left(\frac{\partial\psi}{\partial y}\right)+\frac{\partial}{\partial y}\left(-\frac{\partial\psi}{\partial x}\right)=\frac{\partial^2\psi}{\partial x\,\partial y}-\frac{\partial^2\psi}{\partial y\,\partial x}=0 \qquad (4.7)$$

Since the flow is irrotational:

$$\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}=0 \qquad (4.8)$$

Substituting equation (4.4) and (4.5) into equation (4.8):

$$\frac{\partial}{\partial x}\left(-\frac{\partial\psi}{\partial x}\right)-\frac{\partial}{\partial y}\left(\frac{\partial\psi}{\partial y}\right)=0 \qquad \text{then}$$

$$\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}=0 \qquad (4.9)$$

This is also Laplace's equation. So, the stream function also satisfies Laplace's equation.

4.2 Hess-Smith Panel Method

There are many choices as to how to formulate a panel method (singularity solutions, variation within a panel, singularity strength and distribution, etc.) The simplest and first truly practical method was due to Hess and Smith [19]. It is based on a distribution of sources and vortices on the surface of the geometry. In their method:

$$\phi=\phi_\infty+\phi_s+\phi_v \qquad (4.10)$$

where, ϕ is the total potential function and its three components are the potentials corresponding to the free stream, the source distribution, and the vortex distribution. These last two distributions have potentially locally varying strengths $q(s)$ and $\gamma(s)$, where s is an arc-length coordinate which spans the complete surface of the airfoil in any way you want.

The potentials created by the distribution of sources/sinks and vortices are given by:

$$\phi_s=\int\frac{q(s)}{2\pi}\ln r\,ds$$

$$\phi_v=-\int\frac{\gamma(s)}{2\pi}\theta\,ds \qquad (4.11)$$

where the various quantities are defined in the Figure below:

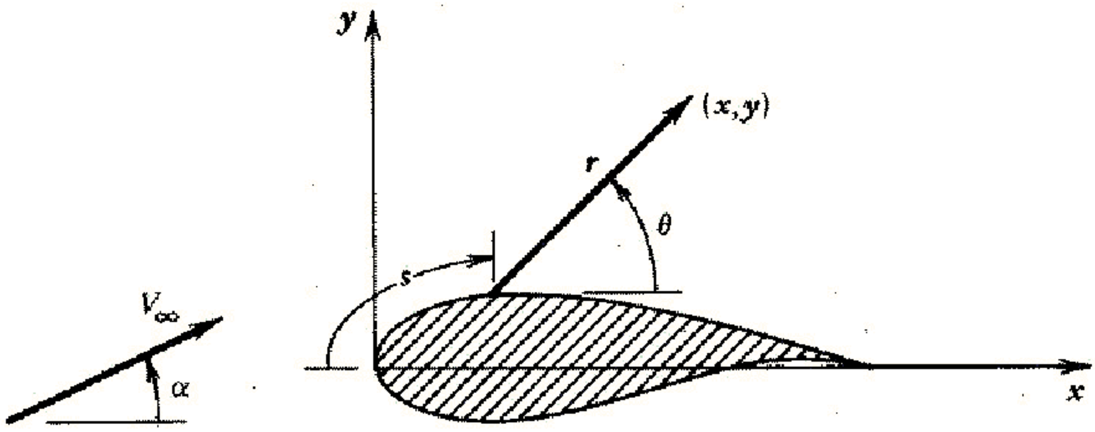


Figure 4.1 Airfoil Analysis Nomenclature for Panel Methods

Notice that in these formulae, the integration is to be carried out along the complete surface of the airfoil. Using the superposition principle, any such distribution of sources/sinks and vortices satisfies Laplace's equation, but we will need to find conditions for q(s) and γ (s) such that the flow tangency boundary condition and the Kutta condition are satisfied.

Notice that we have multiple options. In theory, we could:

- Use the source strength distribution to satisfy flow tangency and the vortex distribution to satisfy the Kutta condition.

- Use arbitrary combinations of both sources/sinks and vortices to satisfy both boundary conditions simultaneously.

Hess and Smith made the following valid simplification:

Take the vortex strength to be constant over the whole airfoil and use the Kutta condition to fix its value, while allowing the source strength to vary from panel to panel so that, together with the constant vortex distribution, the flow tangency boundary condition is satisfied everywhere.

Alternatives to this choice are possible and result in different types of panel methods. Ask if you want to know more about them. Using the panel decomposition from the figure below,

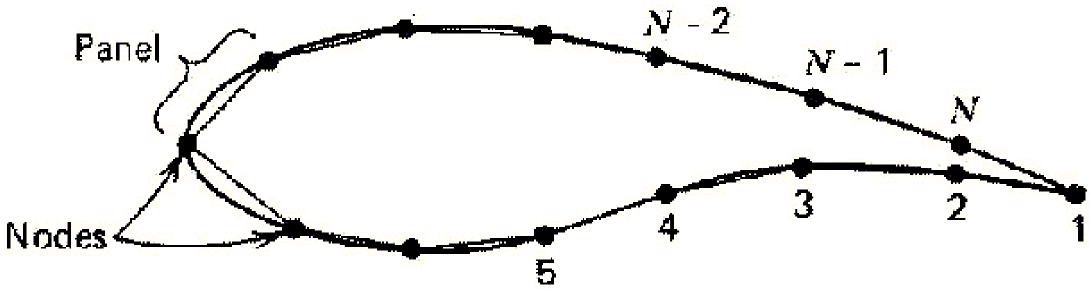


Figure 4.2 Definition of Nodes and Panels

we can ''discretize'' Equation (10) in the following way:

$$\phi = V_{\infty}(x\cos\alpha + y\sin\alpha) + \sum_{j=1}^{N} \int_{panel_j} \left[\frac{q(s)}{2\pi} \ln r - \frac{\gamma}{2\pi}\theta \right] ds \tag{4.12}$$

Since Equation (4.12) involves integrations over each discrete panel on the surface of the airfoil, we must somehow parameterize the variation of source and vortex strength within each of the panels. Since the vortex strength was considered to be a constant, we only need worry about the source strength distribution within each panel.

This is the major approximation of the panel method. However, you can see how the importance of this approximation should decrease as the number of panels, $N \to \infty$ (of course this will increase the cost of the computation considerably, so there are more efficient alternatives.)

If we take the simplest possible approximation, that is, to take the source strength to be constant on each of the panels:

$q(s) = q_i$ on panel I, $\qquad i = 1, \ldots, N$

Therefore, we have $N + 1$ unknowns to solve for in our problem: the N panel source strengths q_i and the constant vortex strength γ. Consequently, we will need $N + 1$ independent equations which can be obtained by formulating the flow tangency boundary condition at each of the N panels, and by enforcing the Kutta condition discussed previously. The solution of the problem will require the inversion of a matrix of size $(N+1) \times (N+1)$.

The final question that remains is: where should we impose the flow tangency boundary condition? The following options are available:

- The nodes of the surface panelization.

- The points on the surface of the actual airfoil, halfway between each adjacent pair of nodes.

- The points located at the midpoint of each of the panels.

We will see in a moment that the velocities are infinite at the nodes of our panelization, which makes them a poor choice for boundary condition imposition.

The second option is reasonable, but rather difficult to implement in practice.

The last option is the one Hess and Smith chose. Although it suffers from a slight alteration of the surface geometry, it is easy to implement and yields fairly accurate results for a reasonable number of panels. This location is also used for the imposition of the Kutta condition (on the last panels on upper and lower surfaces of the airfoil, assuming that their midpoints remain at equal distances from the trailing edge as the number of panels is increased).
If we want to implement the method; consider the ith panel to be located between the ith and (i + 1)th nodes, with its orientation to the x-axis given by

$$\sin\theta_i = \frac{y_{i+1} - y_i}{l_i}$$

$$\cos\theta_i = \frac{x_{i+1} - x_i}{l_i} \tag{4.13}$$

where l_i is the length of the panel under consideration. The normal and tangential vectors to this panel, are then given by

$$\hat{\mathbf{n}}_i = -\sin\theta_i \hat{\mathbf{i}} + \cos\theta_i \hat{\mathbf{j}}$$

(4.14)

$$\hat{\mathbf{t}}_i = \cos\theta_i \hat{\mathbf{i}} + \sin\theta_i \hat{\mathbf{j}}$$

The tangential vector is oriented in the direction from node i to node i +1, while the normal vector, if the airfoil is traversed clockwise, points into the fluid.

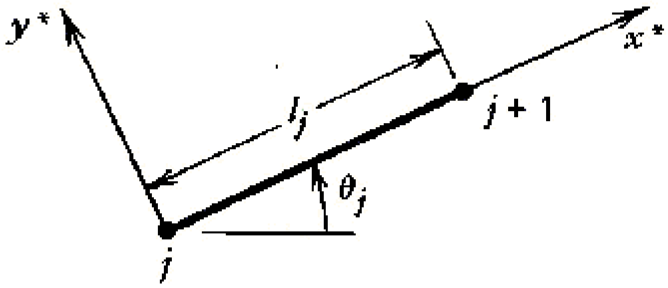


Figure 4.3 Local Panel Coordinate System

Furthermore, the coordinates of the midpoint of the panel are given by

$$\bar{x}_i = \frac{x_i + x_{i+1}}{2}$$

$$\bar{y}_i = \frac{y_i + y_{i+1}}{2} \tag{4.15}$$

and the velocity components at these midpoints are given by

$$u_i = u(\bar{x}_i, \bar{y}_i)$$
$$v_i = v(\bar{x}_i, \bar{y}_i)$$

The flow tangency boundary condition can then be simply written as $(\vec{u} \cdot \vec{n}) = 0$, or, for each panel

$$-u_i \sin\theta_i + v_i \cos\theta_i = 0 \qquad \text{for } i = 1, \ldots, N \tag{4.16}$$

while the Kuttta condition is simply given by

$$u_1 \cos\theta_1 + v_1 \sin\theta_1 = -u_N \cos\theta_N - v_N \sin\theta_N$$
(4.17)

where the negative signs are due to the fact that the tangential vectors at the first and last panels have nearly opposite directions.

Now, the velocity at the midpoint of each panel can be computed by superposition of the contributions of all sources and vortices located at the midpoint of every panel (including itself). Since the velocity induced by the source or vortex on a panel is proportional to the source or vortex strength in that panel, q_i and γ can be pulled out of the integral in Equation (4.12) to yield

$$u_i = V_\infty \cos\alpha + \sum_{j=1}^{N} q_j u_{sij} + \gamma \sum_{j=1}^{N} u_{vij}$$

(4.18)

$$v_i = V_\infty \sin\alpha + \sum_{j=1}^{N} q_j v_{sij} + \gamma \sum_{j=1}^{N} v_{vij}$$

where u_{sij}, v_{sij} are the velocity components at the midpoint of panel i induced by a source of unit strength at the midpoint of panel j. A similar interpretation can be found for u_{vij}, v_{vij}. In a coordinate system tangential and normal to the panel, we can perform the integrals in Equation (4.12) by noticing that the local velocity components can be expanded into absolute ones according to the following transformation:

$$u = u^* \cos\theta_j - v^* \sin\theta_j$$

(4.19)

$$v = u^* \sin\theta_j + v^* \cos\theta_j$$

Now, the local velocity components at the midpoint of the ith panel due to a unit-strength source distribution on this jth panel can be written as

$$u^*_{sij} = \frac{1}{2\pi}\int_0^{l_j} \frac{x^* - t}{(x^* - t)^2 + y^{*2}}\,dt$$

(4.20)

$$v^*_{sij} = \frac{1}{2\pi}\int_0^{l_j} \frac{y^*}{(x^* - t)^2 + y^{*2}}\,dt$$

where (x^*, y^*) are the coordinates of the midpoint of panel i in the local coordinate system of panel j. Carrying out the integrals in Equation (4.20) we find that

$$u^*_{sij} = \frac{-1}{2\pi}\ln\left[(x^* - t)^2 + y^{*2}\right]^{\frac{1}{2}}\Bigg|_{t=0}^{t=l_j}$$

(4.21)

$$v^*_{sij} = \frac{1}{2\pi}\tan^{-1}\frac{y^*}{x^* - t}\Bigg|_{t=0}^{t=l_j}$$

These results have a simple geometric interpretation that can be discerned by looking at the figure below. One can say that

$$u^*_{sij} = \frac{-1}{2\pi}\ln\frac{r_{ij+1}}{r_{ij}}$$

(4.22)

$$v^*_{sij} = \frac{\nu_1 - \nu_0}{2\pi} = \frac{\beta_{ij}}{2\pi}$$

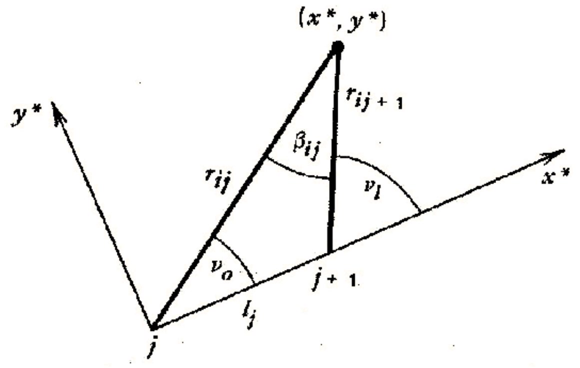


Figure 4.4 Geometric Interpretation of Source and Vortex Induced Velocities

r_{ij} is the distance from the midpoint of panel i to the j_{th} node, while β_{ij} is the angle subtended by the j_{th} panel at the midpoint of panel i. Notice that $u^*_{sii} = 0$, but the value of v^*_{sii} is not so clear. When the point of interest approaches the midpoint of the panel from the outside of the airfoil, this angle $\beta_{ii} \to \pi$. However, when the midpoint of the panel is approached from the inside of the airfoil, $\beta_{ii} \to -\pi$. Since we are interested in the flow outside of the airfoil only, we will always take $\beta_{ii} = \pi$.

Similarly, for the velocity field induced by the vortex on panel j at the midpoint of panel i we can simply see that

$$u^*_{vij} = -\frac{1}{2\pi}\int_0^{l_j} \frac{y^*}{(x^* - t)^2 + y^{*2}}\,dt = \frac{\beta_{ij}}{2\pi}$$

$$v^*_{vij} = -\frac{1}{2\pi}\int_0^{l_j} \frac{x^* - t}{(x^* - t)^2 + y^{*2}}\,dt = \frac{1}{2\pi}\ln\frac{r_{ij+1}}{r_{ij}} \tag{4.23}$$

and finally, the flow tangency boundary condition, using Equation (4.18), and undoing the local coordinate transformation of Equation (4.19) can be written as

$$\sum_{j=1}^{N} A_{ij} q_j + A_{iN+1}\gamma = b_i \tag{4.24}$$

where

$$A_{ij} = -u_{sij}\sin\theta_i + v_{sij}\cos\theta_i$$
$$= -u^*_{sij}(\cos\theta_j\sin\theta_i - \sin\theta_j\cos\theta_i) + v^*_{sij}(\sin\theta_j\sin\theta_i + \cos\theta_j\cos\theta_i) \qquad (4.25)$$

which yields

$$2\pi A_{ij} = \sin(\theta_i - \theta_j)\ln\frac{r_{ij+1}}{r_{ij}} + \cos(\theta_i - \theta_j)\beta_{ij} \qquad (4.26)$$

Similarly for the vortex strength coefficient

$$2\pi A_{iN+1} = \sum_{j=1}^{N}\cos(\theta_i - \theta_j)\ln\frac{r_{ij+1}}{r_{ij}} - \sin(\theta_i - \theta_j)\beta_{ij} \qquad (4.27)$$

The right hand side of this matrix equation is given by

$$b_i = V_\infty\sin(\theta_i - \alpha) \qquad (4.28)$$

The flow tangency boundary condition gives us N equations. We need an additional one provided by the Kutta condition in order to obtain a system that can be solved. According to Equation (4.17)

$$\sum_{j=1}^{N} A_{N+1,j}q_j + A_{N+1,N+1}\gamma = b_{N+1} \qquad (4.29)$$

After similar manipulations we find that

$$2\pi A_{N+1,j} = \sum_{k=1,N}\sin(\theta_k - \theta_j)\beta_{kj} - \cos(\theta_k - \theta_j)\ln\frac{r_{kj+1}}{r_{kj}}$$
$$(4.30)$$
$$2\pi A_{N+1,N+1} = \sum_{k=1,N}\sum_{j=1}^{N}\sin(\theta_k - \theta_j)\ln\frac{r_{kj+1}}{r_{kj}} + \cos(\theta_k - \theta_j)\beta_{kj}$$

$$b_{N+1} = -V_\infty \cos(\theta_1 - \alpha) - V_\infty \cos(\theta_N - \alpha) \tag{4.31}$$

where the sum $\sum_{k=1,N}$ are carried out only over the first and last panels, and not the range $[1, N]$. These various expressions set up a matrix problem of the kind

$$Ax = b$$

where the matrix A is of size (N + 1)x(N + 1). This system can be sketched as follows:

$$\begin{bmatrix} A_{11} & \cdots & A_{1i} & \cdots & A_{1N} & A_{1,N+1} \\ \vdots & & \vdots & & \vdots & \vdots \\ A_{i1} & \cdots & A_{ii} & \cdots & A_{iN} & A_{i,N+1} \\ \vdots & & \vdots & & \vdots & \vdots \\ A_{N1} & \cdots & A_{Ni} & \cdots & A_{NN} & A_{N,N+1} \\ A_{N+1,1} & \cdots & A_{N+1,i} & \cdots & A_{N+1,N} & A_{N+1,N+1} \end{bmatrix} \begin{bmatrix} q_1 \\ \vdots \\ q_i \\ \vdots \\ q_N \\ \gamma \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_N \\ b_{N+1} \end{bmatrix} \tag{4.32}$$

Notice that the cost of inversion of a full matrix such as this one is $O(N+1)^3$, so that, as the number of panels increases without bounds, the cost of solving the panel problem increases rapidly.

Finally, once you have solved the system for the unknowns of the problem, it is easy to construct the tangential velocity at the midpoint of each panel according to the following formula

$$\begin{aligned} V_{ti} = {} & V_\infty \cos(\theta_i - \alpha) + \sum_{j=1}^{N} \frac{q_j}{2\pi}\left[\sin(\theta_i - \theta_j)\beta_{ij} - \cos(\theta_i - \theta_j)\ln\frac{r_{ij+1}}{r_{ij}}\right] \\ & + \frac{\gamma}{2\pi}\sum_{j=1}^{N}\left[\sin(\theta_i - \theta_j)\ln\frac{r_{ij+1}}{r_{ij}} + \cos(\theta_i - \theta_j)\beta_{ij}\right] \end{aligned} \tag{4.33}$$

And knowing the tangential velocity component, we can compute the pressure coefficient (no approximation since $V_{ni} = 0$) at the midpoint of each panel according to the following formula

$$C_p(\bar{x}_i, \bar{y}_i) = 1 - \frac{V_{ti}^2}{V_\infty^2}$$

from which the force and moment coefficients can be computed assuming that this value of Cp is constant over each panel and by performing the discrete sum

5. INVERSE DESIGN OF AIRFOILS

In inverse design, purpose is to find a proper airfoil shape, which gives the pre-specified pressure distribution. In this work two methods used to implement inverse design. First method used is based on a genetic algorithm. Second method is based on a residual correction algorithm.

5.1 Inverse design with Genetic Algorithm

In this method, Fortran code of a genetic algorithm written by David L. Carroll (University of Illinois) is used. Inverse design problem is solved as an optimization problem such that, the value of the below equation is maximized.

$$\text{Funcval} = \sum_{i=1}^{n} -(c_{p_i} - Tc_{p_i})^2 \tag{5.1}$$

So an inverse problem may transform into an optimization problem with this formulation. This genetic algorithm code is compiled together with Smith-Hess Panel Method Code and B-spline Curve Generator Code. Flow chart of this code is as follow:

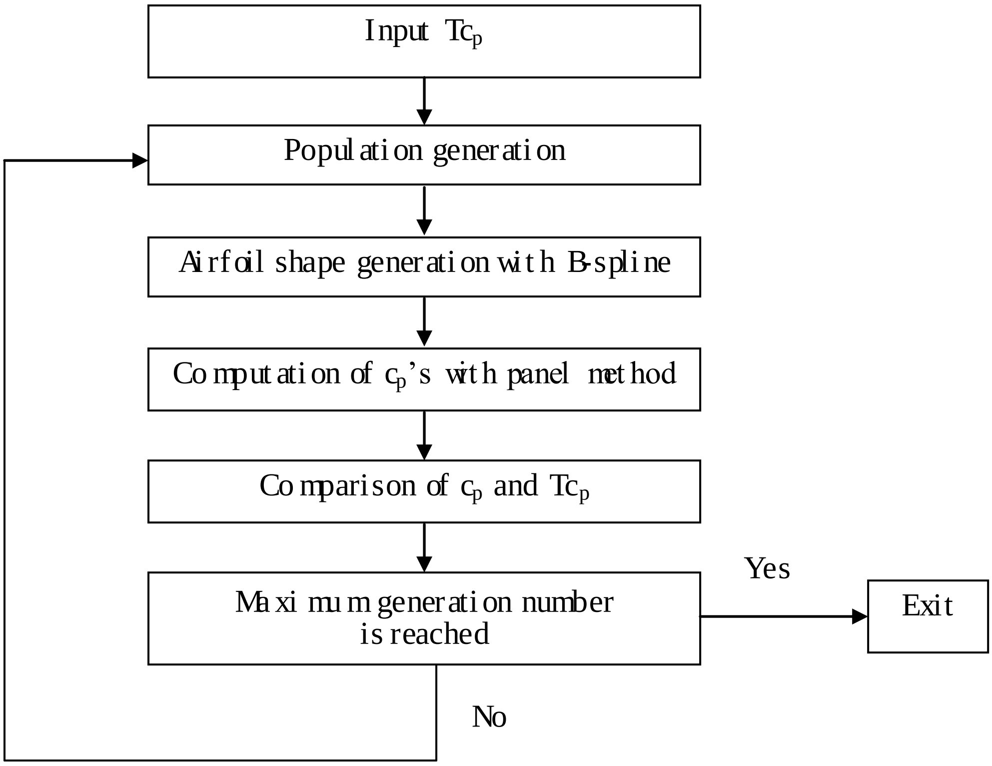


Figure 5.1 Inverse design with genetic algorithm flow chart

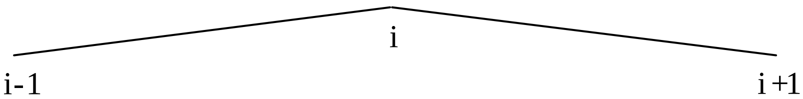
5.2 Inverse Design with Residual Correction Algorithm

In this method, a residual correction method is used [17]. Corresponding differential equation:

$$A\Delta Y + B\frac{d\Delta Y}{dx} + C\frac{d^2\Delta Y}{dx^2} = V_t^2 - V^2 \tag{5.2}$$

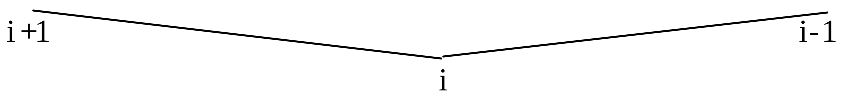
A, B, C are arbitrary constants determining the rate of change of the airfoil.

If we use finite differences to discritize the equation, approximation of $\frac{d\Delta Y}{dx}$ on the upper surface:



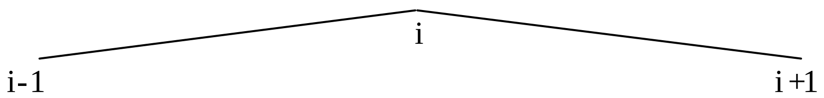
$$\frac{\partial \Delta Y}{\partial x} = \frac{\Delta Y_{i+1} - \Delta Y_i}{x_{i+1} - x_i} \tag{5.3}$$

Approximation of $\dfrac{d\Delta Y}{dx}$ on the lower surface:



$$\frac{\partial \Delta Y}{\partial x} = \frac{\Delta Y_i - \Delta Y_{i-1}}{x_i - x_{i-1}} \tag{5.4}$$

Then approximation of $\dfrac{d^2\Delta Y}{dx^2}$ on the upper surface:



$$\frac{\partial^2 \Delta Y}{\partial x^2} = \frac{\dfrac{\Delta Y_{i+1} - \Delta Y_i}{x_{i+1} - x_i} - \dfrac{\Delta Y_i - \Delta Y_{i-1}}{x_i - x_{i-1}}}{\left(\dfrac{x_{i+1} - x_{i-1}}{2}\right)} \tag{5.5}$$

A similar equation occurs on the lower surface. At the leading edges and the trailing edges ΔY is accepted as zero. With these approximations, we can write following formulation:

$$A\Delta Y_i + B\frac{\Delta Y_{i+1} - \Delta Y_i}{x_{i+1} - x_i} - 2C\frac{\dfrac{\Delta Y_{i+1} - \Delta Y_i}{x_{i+1} - x_i} - \dfrac{\Delta Y_i - \Delta Y_{i-1}}{x_i - x_{i-1}}}{x_{i+1} - x_{i-1}} = V_{t_i}^2 - V_i^2 \tag{5.6}$$

Then the coefficients of the system $K_i\Delta Y_{i+1} + L_i\Delta Y_i + M_i\Delta Y_{i-1} = V_{t_i}^2 - V_i$ are:

$$K_i = \frac{B}{x_{i+1} - x_i} - \frac{2C}{(x_{i+1} - x_i)(x_{i+1} - x_{i-1})}$$

$$L_i = A - \frac{B}{x_{i+1} - x_i} + \frac{2C}{x_{i+1} - x_{i-1}}\left(\frac{1}{x_{i+1} - x_i} + \frac{1}{x_i - x_{i-1}}\right) \tag{5.7}$$

$$M_i = -\frac{2C}{(x_i - x_{i-1})(x_{i+1} - x_{i-1})}$$

As a result of this formulation, we can constitute tri-diagonal NxN coefficient matrix. To solve this system Thomas algorithm is used. So we can have a ΔY for each point on the airfoil. Then, using this ΔY we can find new Y value for each point, such that:

$$Y_{new} = Y_{old} + \Delta Y \tag{5.8}$$

A, B, and C constants determine the sensitivity of ΔY; the bigger constants lead to smaller ΔY values. This algorithm code is also compiled together with Smith-Hess Panel Method Code. Flow chart is as follows:

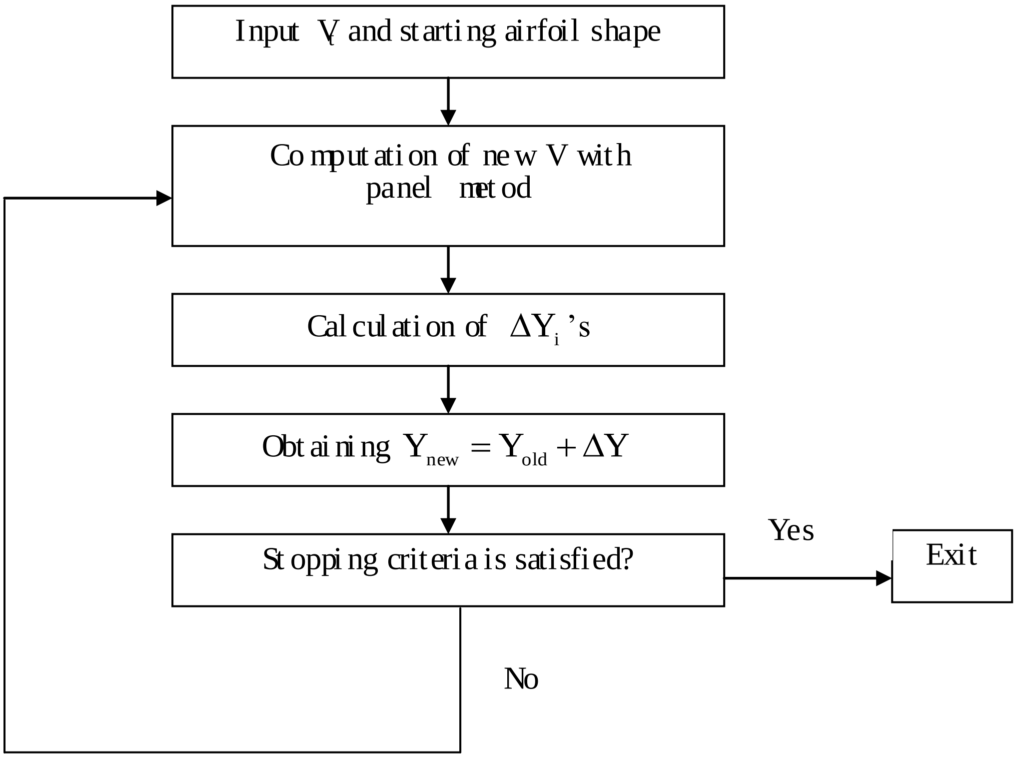
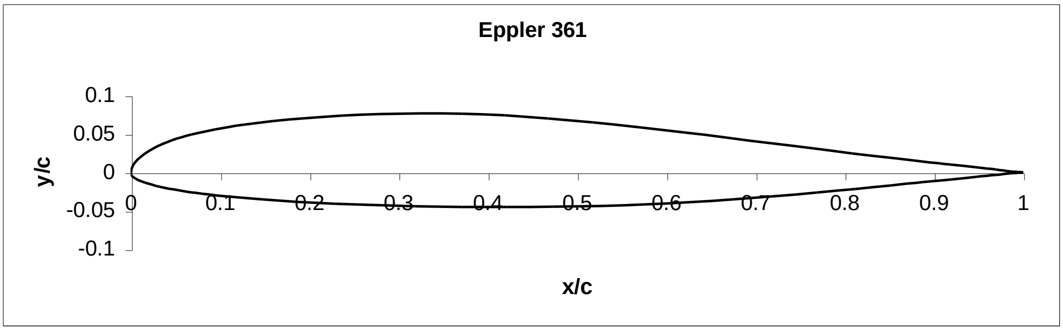


Figure 5.2 Inverse design with residual correction algorithm flow chart

6. RESULTS AND CONCLUSION

To test previously mentioned two methods, a test case is utilized. Eppler 361 airfoil that is designed for rotorcrafts is used for test case. Flow around Eppler 361 airfoil with 5-degree angle of attack is analyzed. As a result of this analysis, pressure distribution and velocity distribution on the airfoil are obtained. Below figures show Eppler 261 airfoil geometry and pressure distribution on the airfoil respectively.



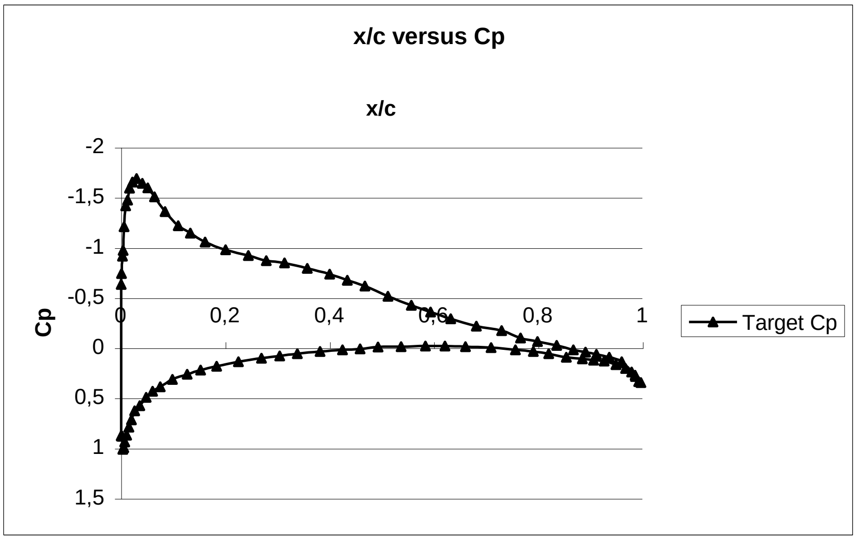
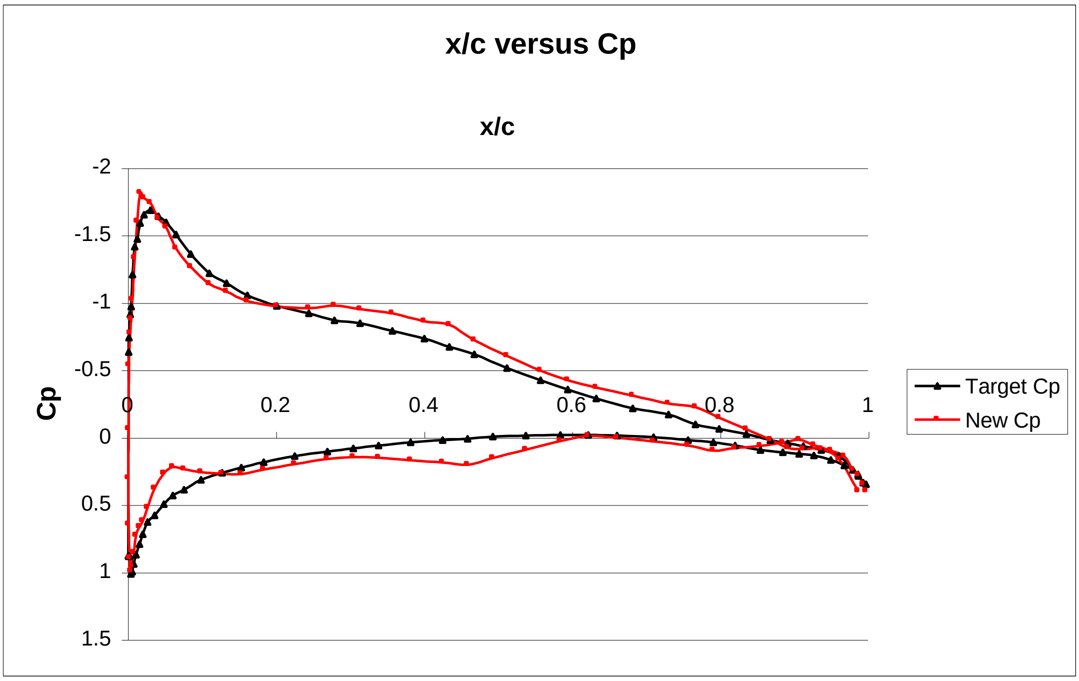


Figure 6.1 Target geometry and pressure distribution

6.1 Results of Inverse Design with Genetic Algorithm

In inverse design with genetic algorithm, results of pressure distribution and airfoil geometry obtained from different iteration number are presented with the target pressure distribution and airfoil geometry.

For generation number 200, results may be seen below:



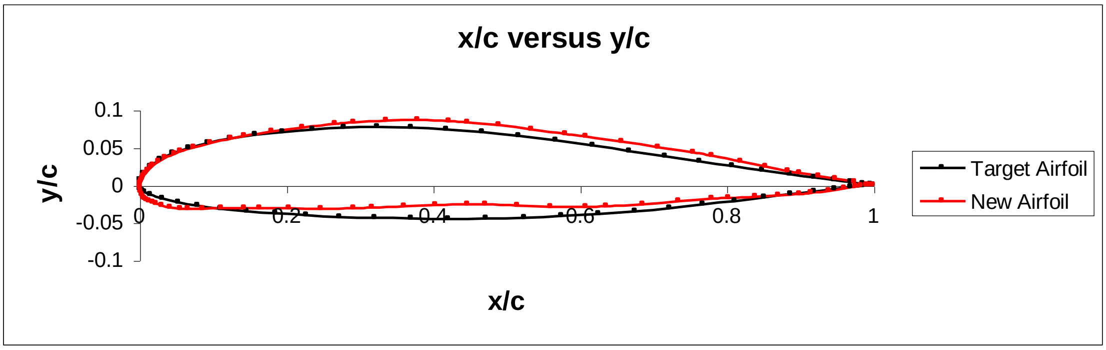
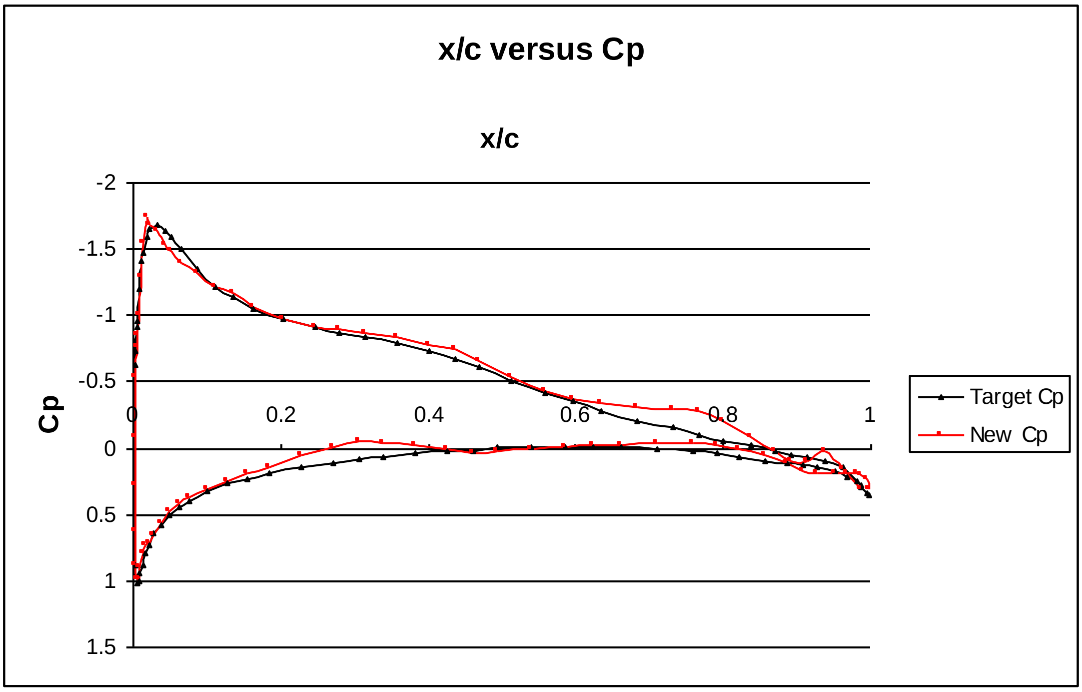


Figure 6.2 Results of genetic algorithm after generation 200

For generation number 200, it is seen that results is not so good and there are remarkable differences in pressure distributions and, of course, airfoil geometry. While iteration number increases, results obtained start to resemble to target values. For example in generation number 1000, results may be seen below:



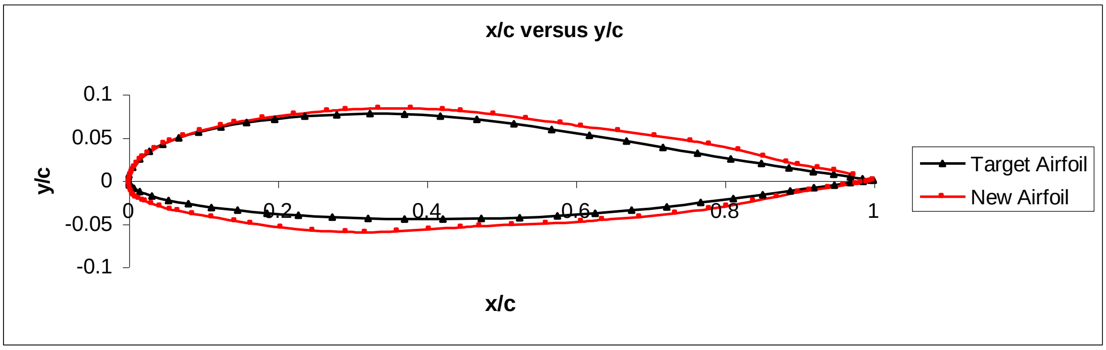
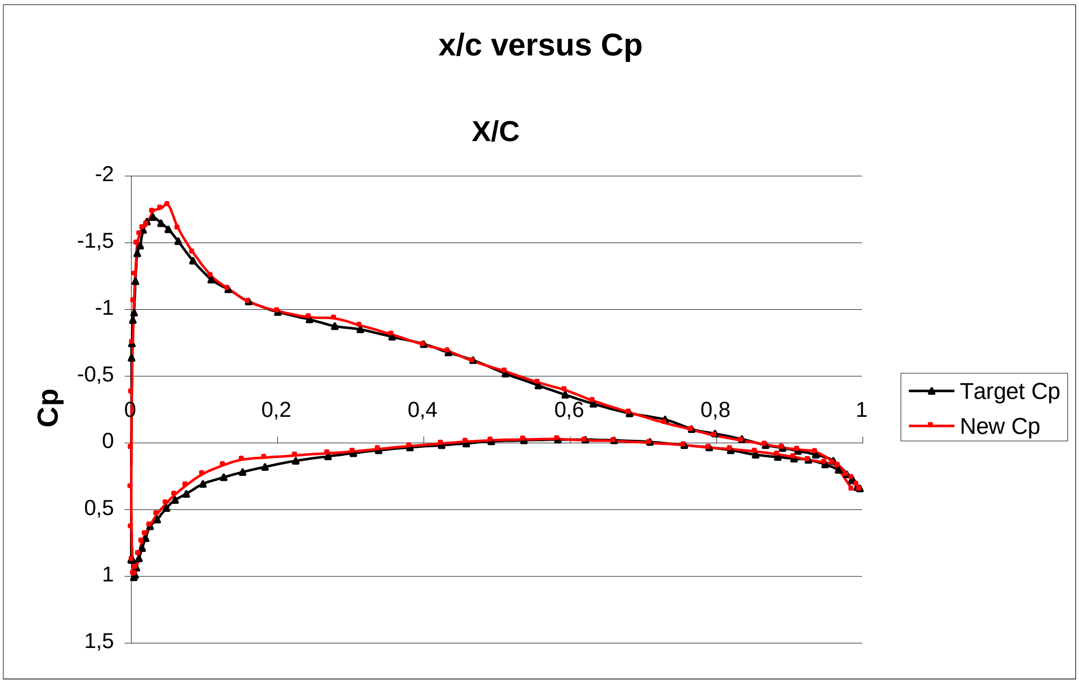


Figure 6.3 Results of genetic algorithm after generation 1000

Although results seem better with respect to generation number 200, differences from target values are not negligible. So if we continue to iterate, we can come up with good results. As an example, obtained results in generation number 4000 may be seen below:



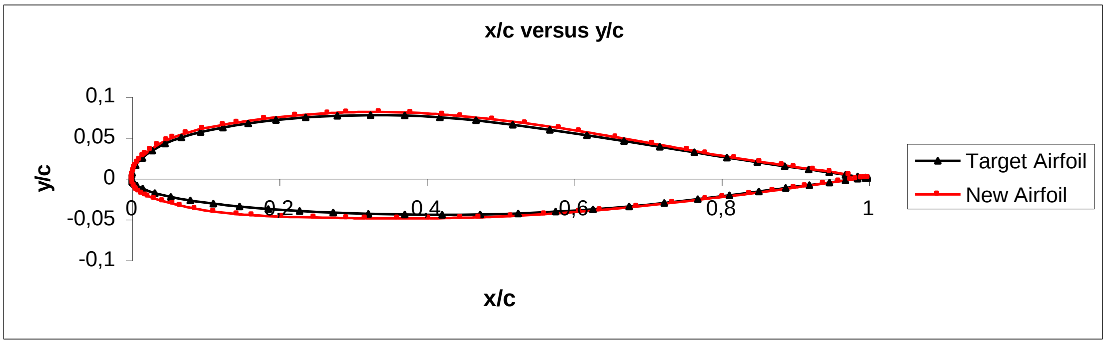


Figure 6.4 Results of genetic algorithm after generation 4000

As it is seen in the figures above, at the generation number 4000, target pressure distributions and airfoil geometry are obtained with slight differences. More results from different generation numbers are available in the appendix. At first, looking at generation number 4000, cost of this computation may be regarded as high. But, since the time consumed by the algorithm for one generation is too low, actually it is not so costly.

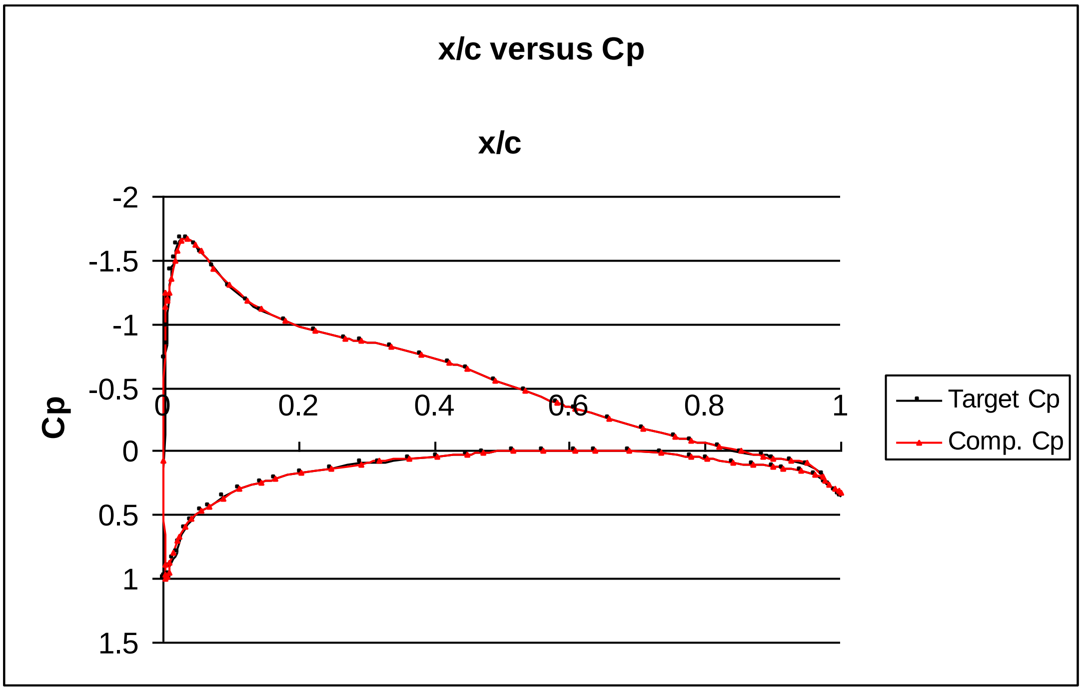
6.2 Results of Inverse Design with Residual Correction Algorithm

While implementing this method, different A, B, and C constant values are used to see the effects of these constants on results and iteration number. In addition, for this method two different criteria are defined. These are:

a) $\sum_{i=1}^{n}\left(\left|Tc_{p_i}\right|-\left|c_{p_i}\right|\right)^2 \leq 1.5$ and

b) $\sum_{i=1}^{n}\left(\left|Tc_{p_i}\right|-\left|c_{p_i}\right|\right)^2 \leq 1$

When these criteria are satisfied, algorithm stops. For example, for the case of A = 1, B = 1, C = 1, obtained results for criteria (a), after the iteration number 80 are as follows:



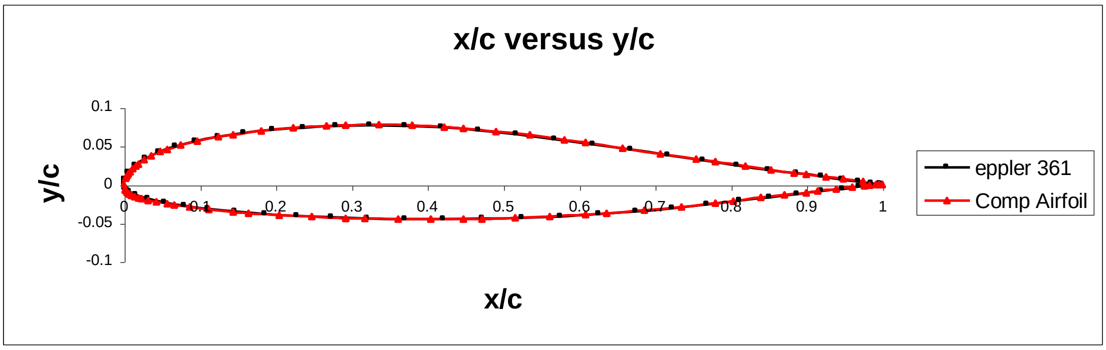
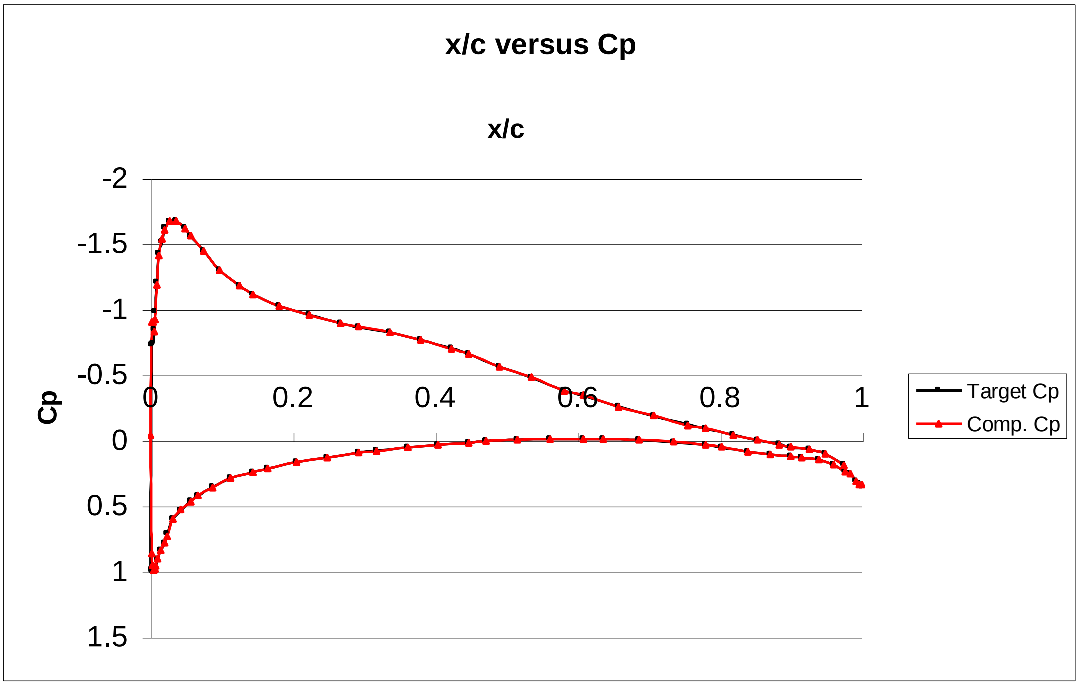


Figure 6.5 Results of residual correction algorithm after iteration 80 (A,B, C=1)

If we apply the second criteria with the same A, B and C values, after 977 iteration we come up with:



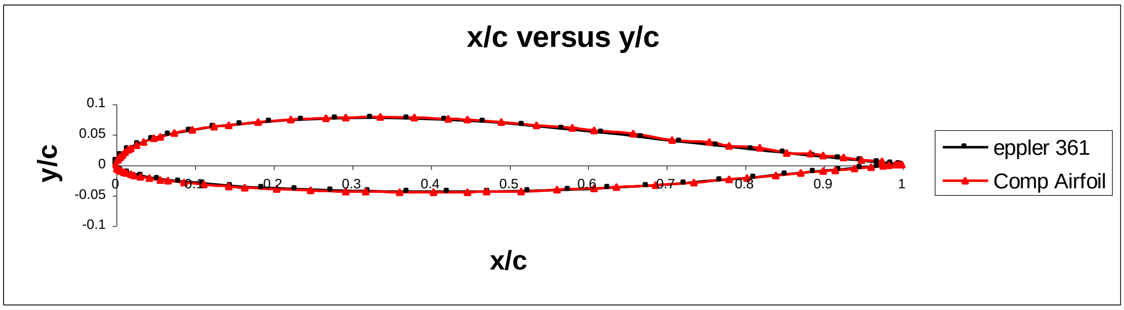
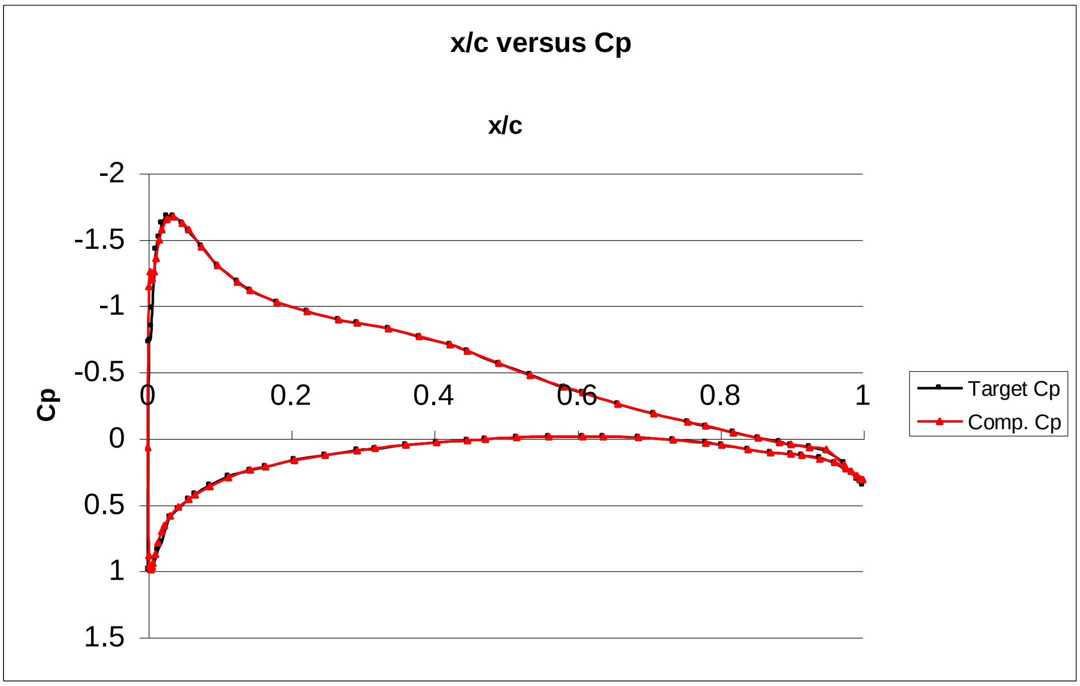


Figure 6.6 Results of residual correction algorithm after iteration 977 (A,B, C=1)

It is seen that, for this criteria, differences occurred about the leading edge in first criteria mostly disappeared. So we can reach nearly the same pressure distribution with target pressure distribution.

If we take the A = 3, B =3, C = 3 for the same two criteria, pressure distribution, airfoil geometry and iteration number at which the criteria is satisfied are presented below

For the first criteria, after 237 iterations, results are:



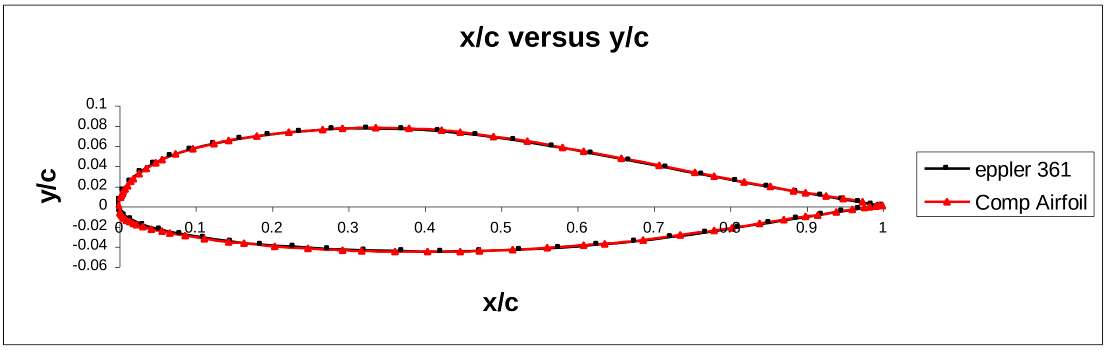
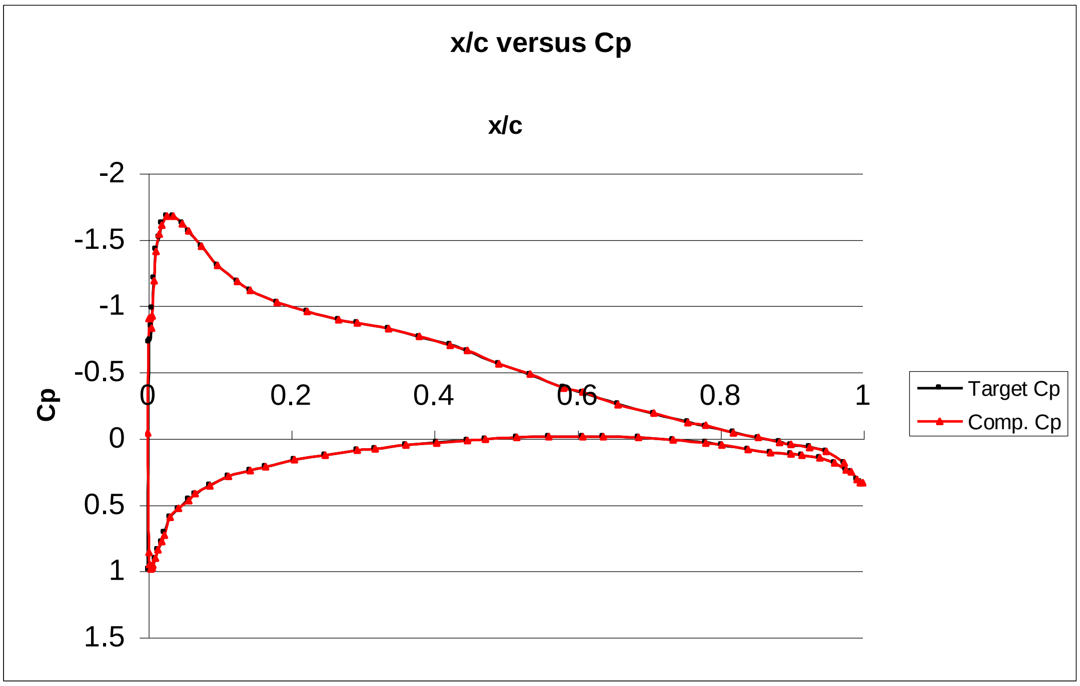


Figure 6.7 Results of residual correction algorithm after iteration 237 (A,B, C=3)

For the second criteria, after 2931 iterations, results are:



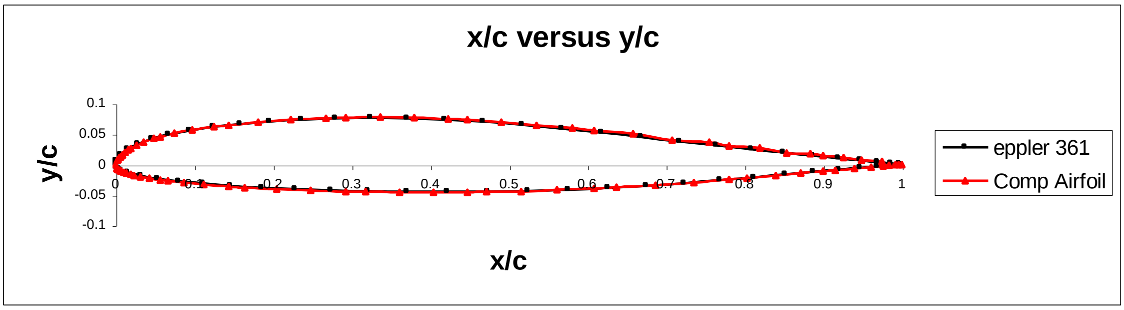
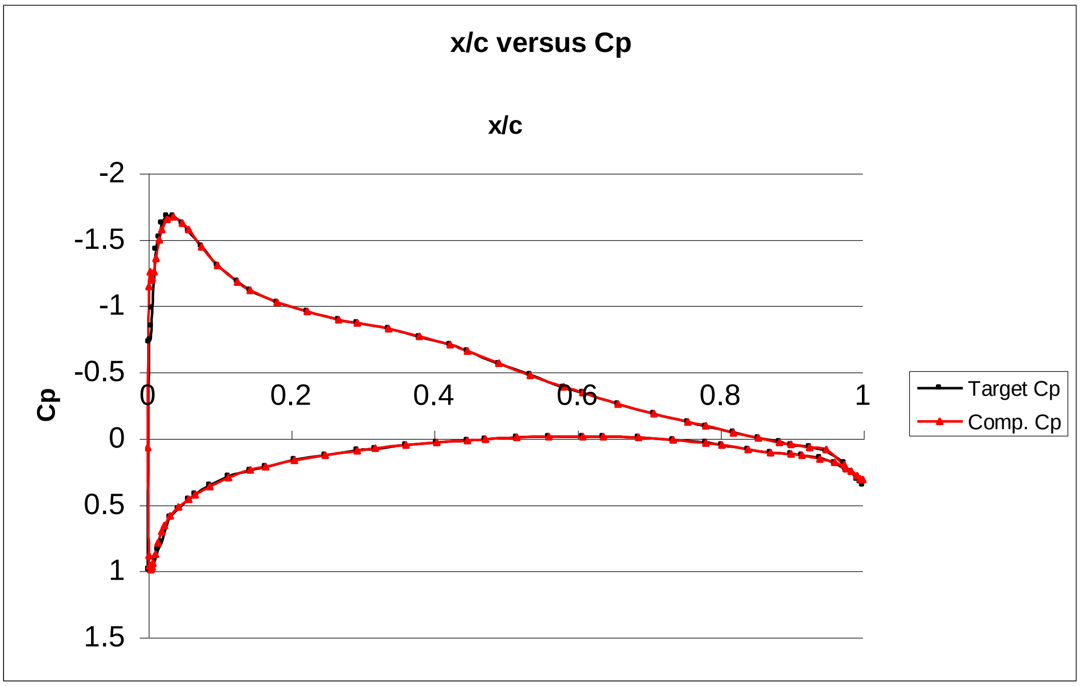


Figure 6.8 Results of residual correction algorithm after iteration 2931 (A, B, C=3)

As expected, iteration numbers gets larger, while there is an increase in A, B, C. To see this effect more evidently, it may be beneficial to see the results of the case A = 10, B = 10, C = 10. For the first criteria, after 789 iteration, obtained results are:



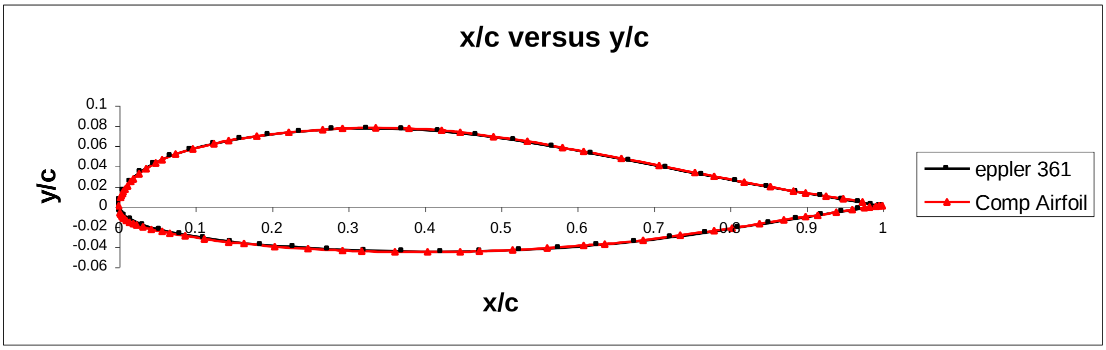


Figure 6.9 Results of residual correction algorithm after iteration 789 (A,B, C=10)

For the second criteria, after 9763 iteration, obtained results are:

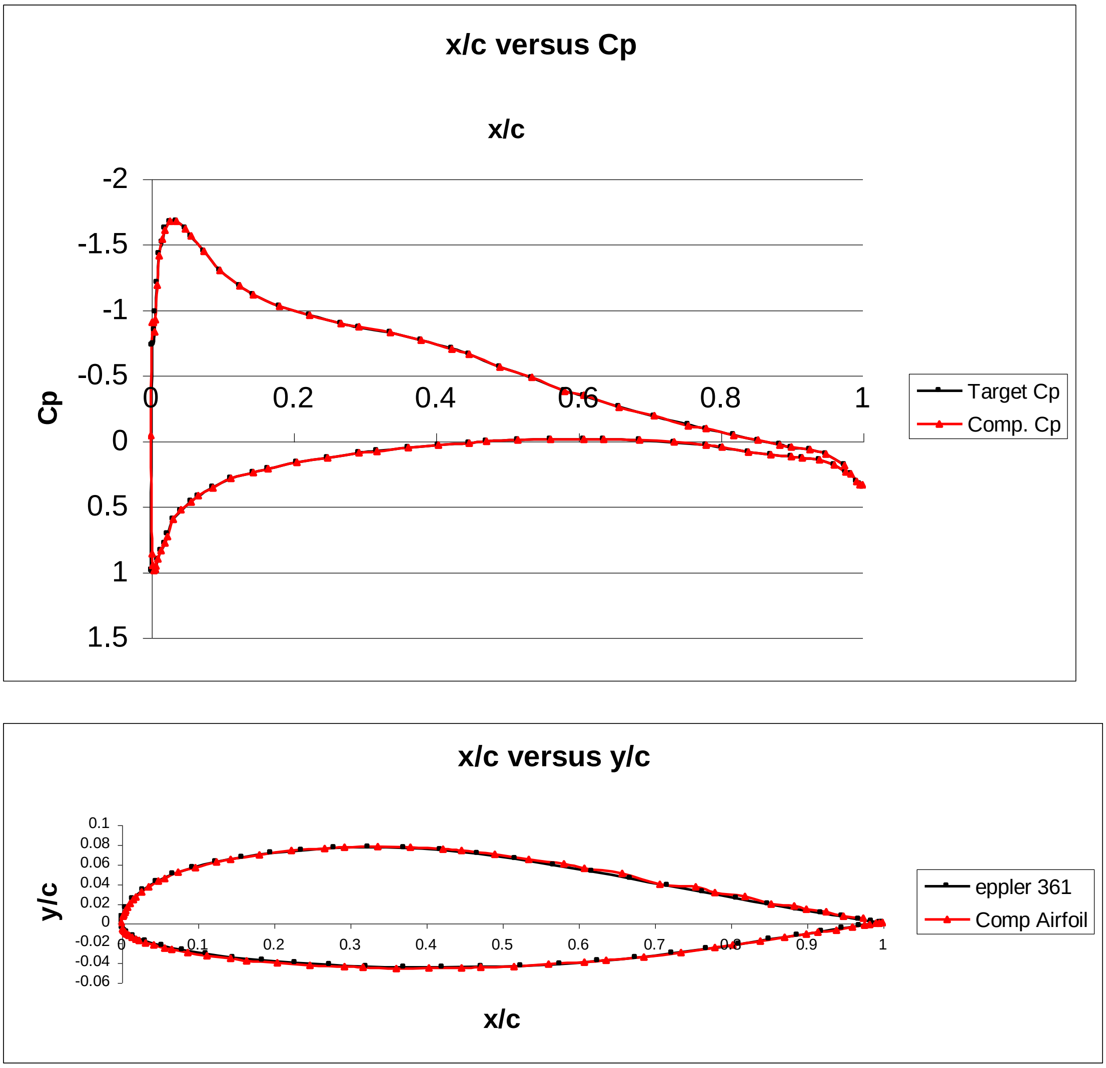


Figure 6.10 Results of residual correction algorithm after iteration 9763 (A, B, C=10)

By using these three cases, we can see the effects of A, B, C constants on the required iteration to satisfy the criteria. Following table shows the trend.

Table 6.1 Comparison of results according to constants and criteria

Values of A, B, C	Required iteration number for the first criteria	Required iteration number for the second criteria
1	80	977
3	237	2931
10	789	9763

6.3 Conclusion

Both inverse design methods give acceptable results. Method with genetic algorithm creates starting generations randomly. Therefore, although obtained results are in the acceptable limits, some differences from target values take attention. But on the other hand, randomness of the genetic algorithm also creates an advantage. That is, since genetic algorithm has randomness, accuracy of the algorithm does not change too much depending on position on the airfoil. And this feature of the algorithm may create an advantage in the case of more complex geometries.

The method with residual correction algorithm seems relatively better than the method with genetic algorithm. For this test case, because residual correction algorithm starts iteration with a certain airfoil (NACA 0012), it reaches better results. But since, in this algorithm, target values are reached by using ΔY's, sensitivity of ΔY is important for obtaining target values. So algorithm accuracy varies according to position on the airfoil.

For example, although obtained results for other part of the airfoil fit very well to the target values, algorithm accuracy does not show the same trend in the leading edge. In leading edge, remarkable difference is seen. To correct this situation, we must decrease value of criteria and this leads to an increase in iteration number.

In addition to these, in the case of more complex problems, first method is used to create an initial airfoil geometry for the second method.

REFERENCES

[1] **Anderson, J. D,** 1991. Fundamentals of Aerodynamics, McGraw-Hill Inc., New York.

[2] **Gerald, C. F. and Wheatley, P. O,** 1999. Applied numerical analysis. Reading, Addison-Wesley

[3] **Renner, G and Ekart, A,** 2003. Genetic algorithms in computer aided design, *Computer-Aided Design*, 3**5** 709-726

[4] **Goldberg, D. E.,** 1989. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Reading.

[5] **Obayashi, S. and Tsukahara, T.,** 1997. Comparison of optimization algorithms for aerodynamic shape design. *AIAA Journal* , **35(8)**: 1413-5.

[6] **Yamamoto, K and Inoue, O,** 1995. Applications of genetic algorithm to aerodynamic shape optimization. *Technical Report AIAA-95-1650-CP. AIAA*

[7] **Wang, J. F., Periaux, J. and Sefrioui, M,** 2002. Parallel evolutionary algorithms for optimization problems in aerospace engineering. *Journal of Computational and Applied Mathematics* **149** 155 -169

[8] **Quagliarella, D and Vicini, A,** 2001. Viscous single and multicomponent airfoil design with genetic algorithms. *Finite Elements in Analysis and Design* **37** 365-380

[9] **Vicini, A and Quagliarella, D,** 1997. Inverse and direct airfoil design using a multiobjective genetic algorithm, *AIAA J.*, **35 (9)** 1499-1505

[10] **Marco, N and Lanteri, S.,** 2000. A two-level parallelization strategy for Genetic Algorithms applied to optimum shape design. *Parallel Computing* **26** 377-397

[11] **Cappello, F. and Mancuso, A,** 2003. A genetic algorithm for combined topology and shape optimizations. *Computer-Aided Design* **35** 761-769

[12] **Oyama, A, Obayashi, S. and Nakamura, T.,** 2001. Real-coded adaptive range genetic algorithm applied to transonic wing optimization. *Applied Soft Computing* **1** 179-187

[13] **Cerrolaza, M, Annicchiarico, W and Martinez, M,** 2000. Optimization of 2D boundary element models using B-splines and genetic

algorithms. *Engineering Analysis with Boundary Elements* **24** 427-440

[14] **Annicchiarico, W. and Cerrolaza, M.,** 1999 Finite elements, genetic algorithms and B-splines: a combined technique for shape optimization. *Finite Elements in Analysis and Design* **33** (1999) 125-141

[15] **Fray, J. M. J. and Slooff, J. W,** 1980. A constrained inverse method for the aerodynamic design of thick wings with given pressure disribution in subsonic flow. *AGARD CP-No.285*, paper 16.

[16] **Iollo, A, Ferlauto, M and Zannetti, L.,** 2001. An Aerodynamic Optimization Method Based on the Inverse Problem Adjoint Equations. *Journal of Computational Physics* **173**, 87-115

[17] **Malone, J. B, Narramore, J. C and Sankar, L. N,** Airfoil Design Method Using the Navier-Stokes Equations. *Journal of Aircraft*, **28 (3)** 216-224

[18] **Huang, C. H and Hsiung, T. Y,** 1999. An inverse design problem of estimating optimal shape of cooling passages in turbine blades. *International Journal of Heat and Mass Transfer* **42** 4307-4319

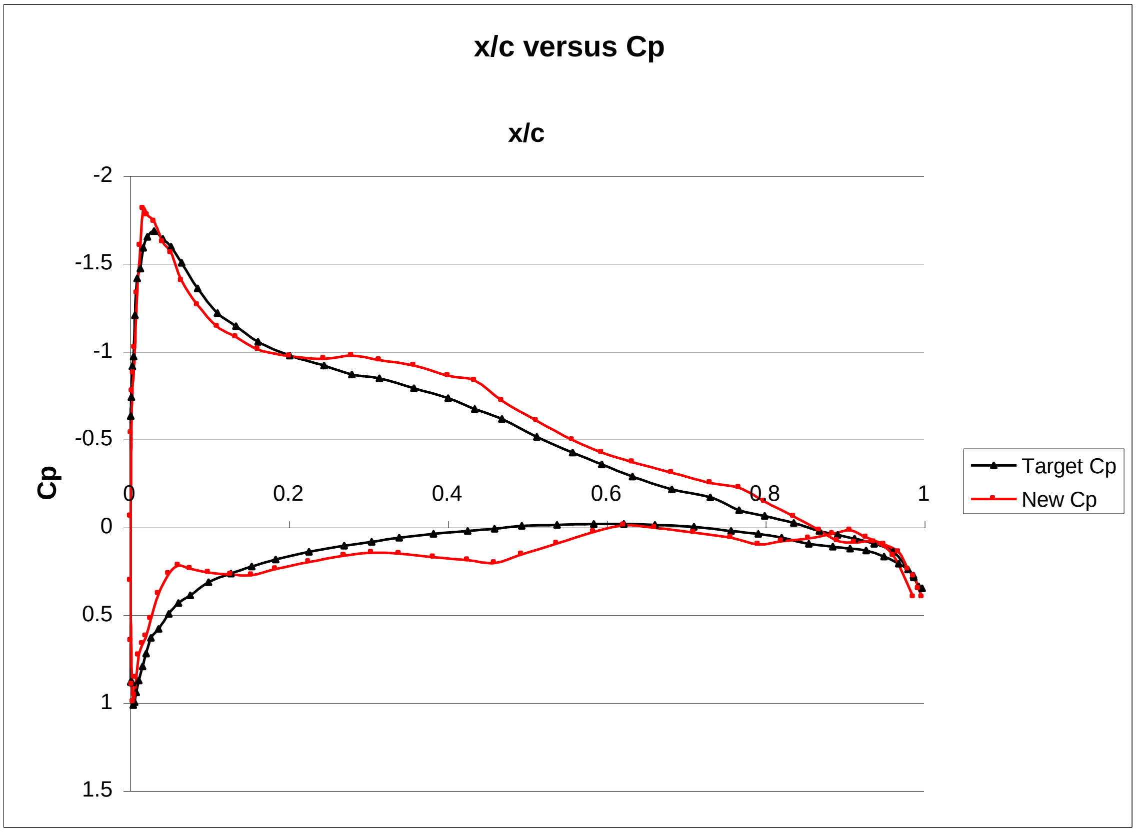
[19] Smith-Hess Panel Method, 15/01/2003 http://cromagnon.stanford.edu/aa200b/lect_notes/lect3.pdf

APPENDIX A

All results from both inverse design with genetic algorithm and inverse design with residual correction algorithm are presented below:

A 1 Results of Inverse Design with Genetic Algorithm

Generation 200:



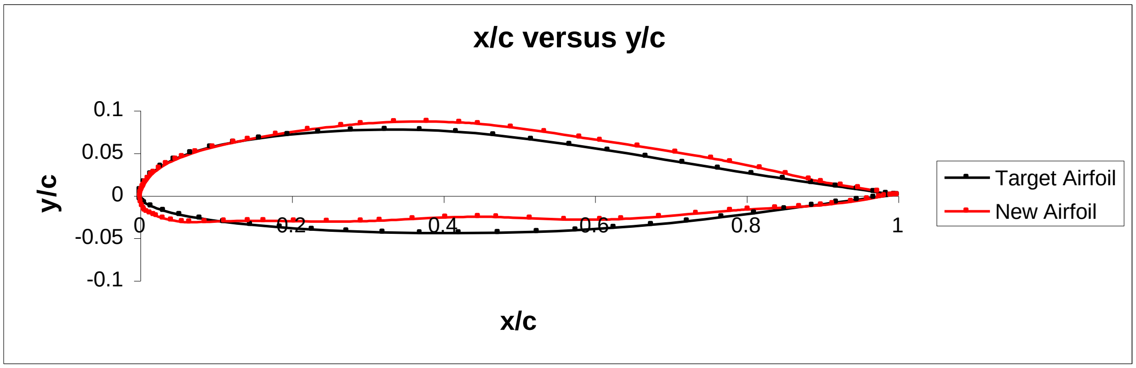
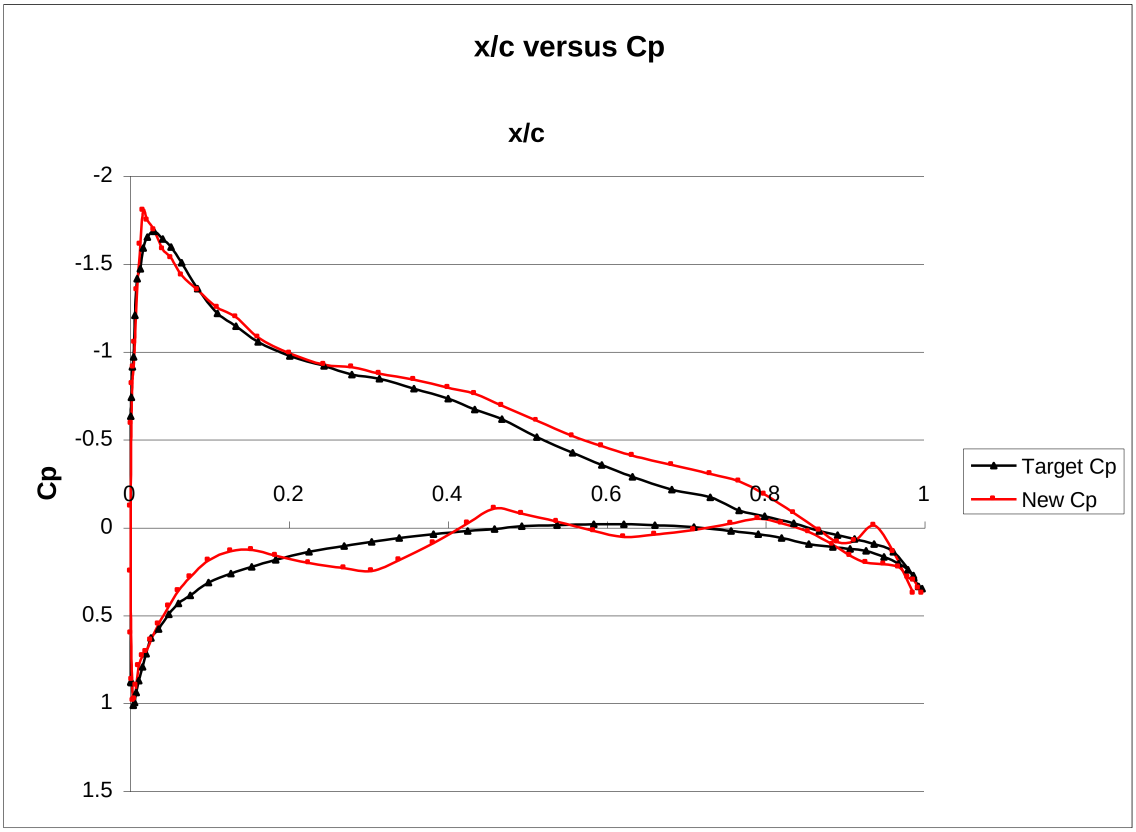


Figure A 1 Results of genetic algorithm after generation 200

Generation 400:



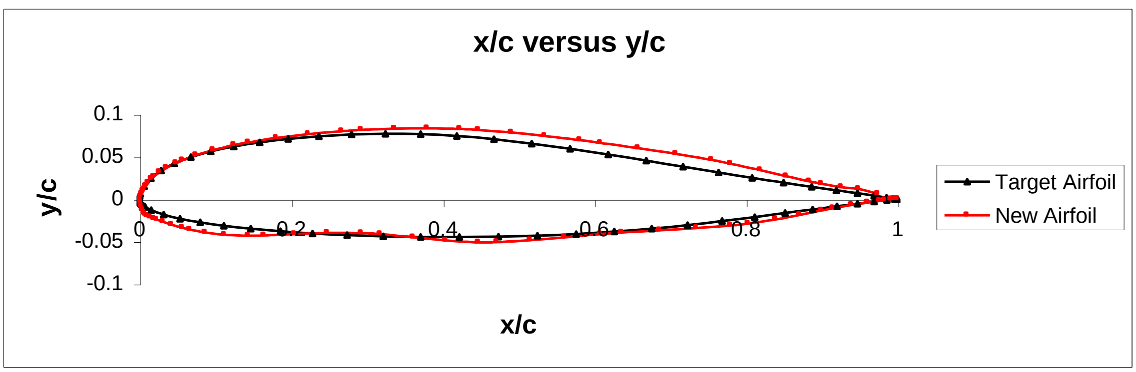
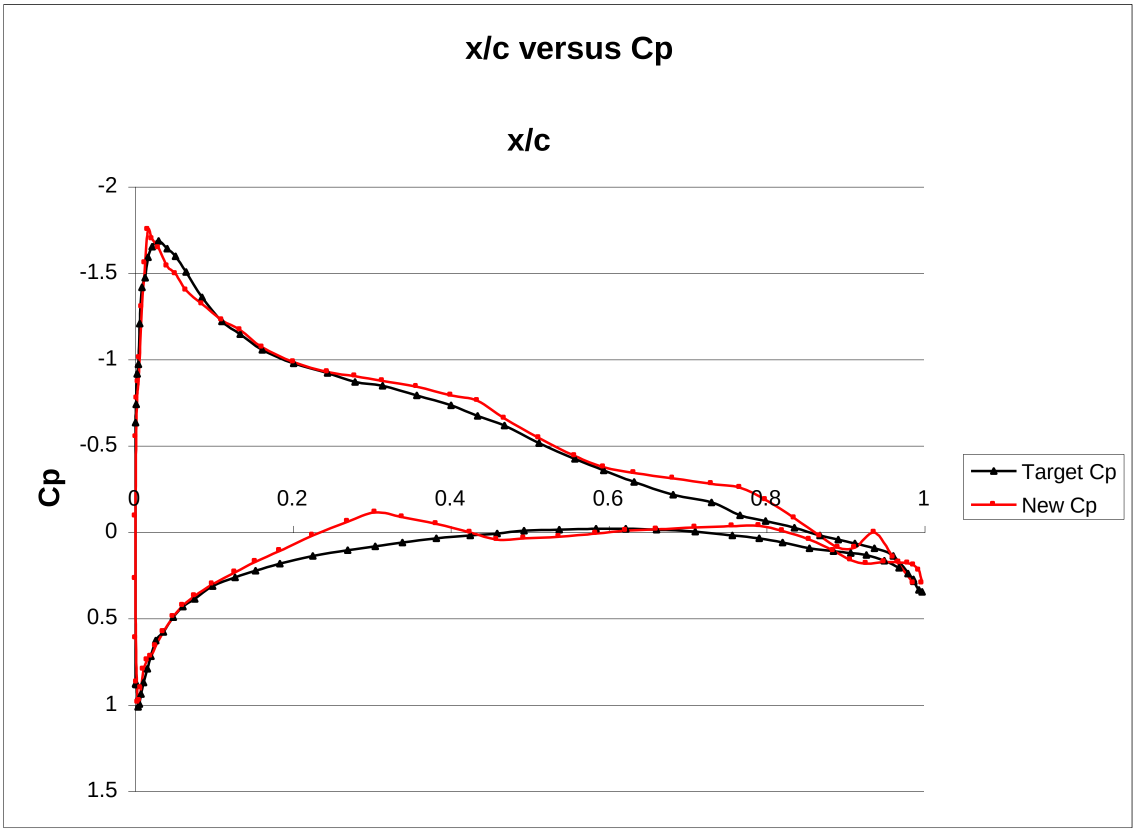


Figure A2 Results of genetic algorithm after generation 400

Generation 800:



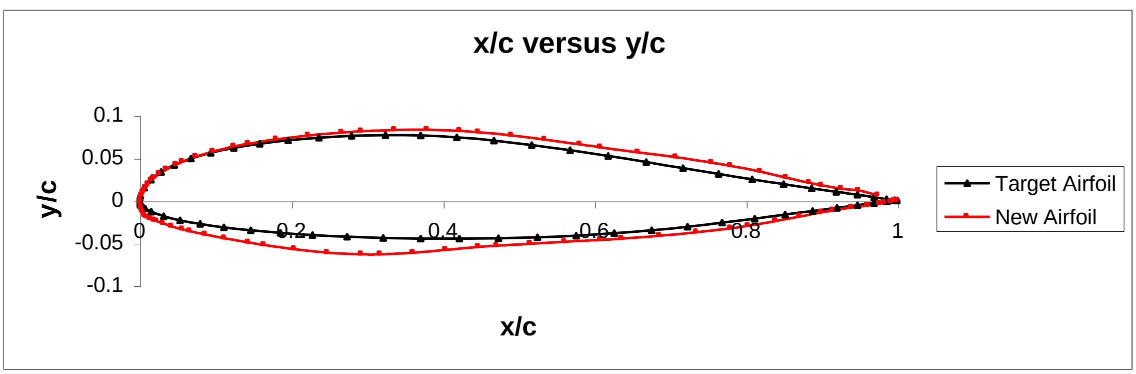
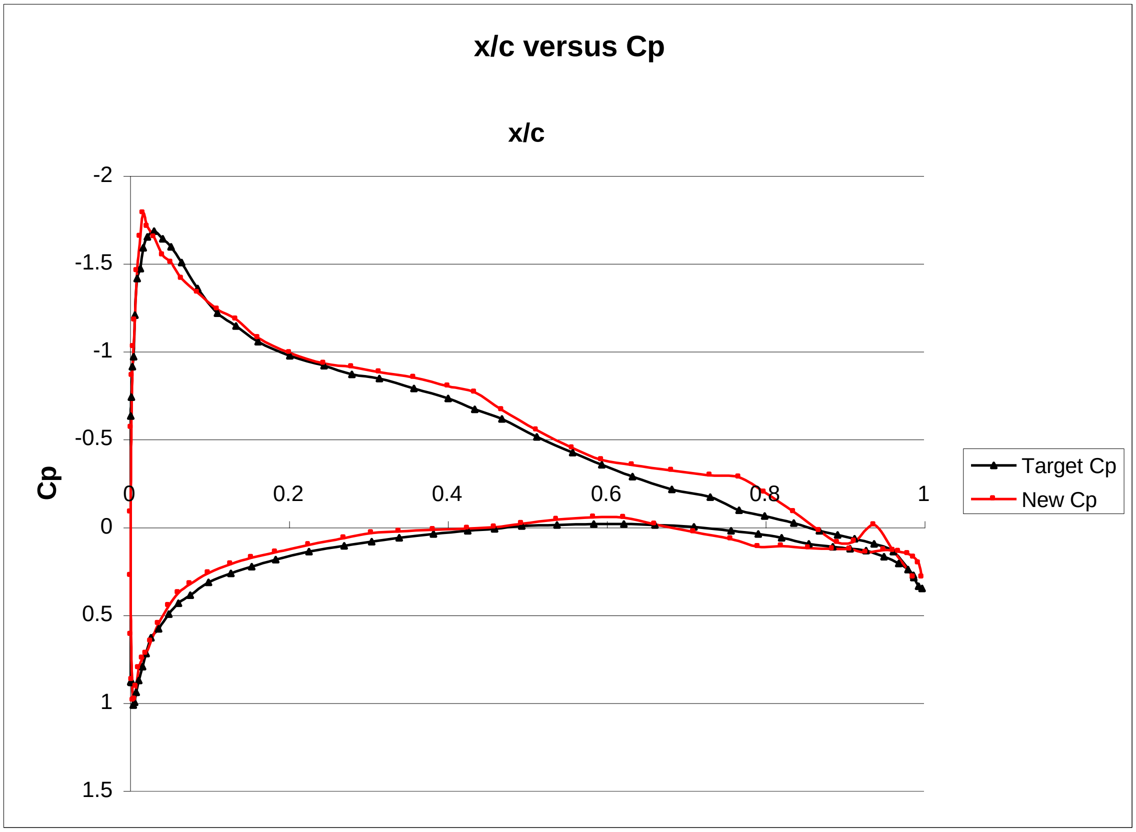


Figure A3 Results of genetic algorithm after generation 800

Generation 1200:



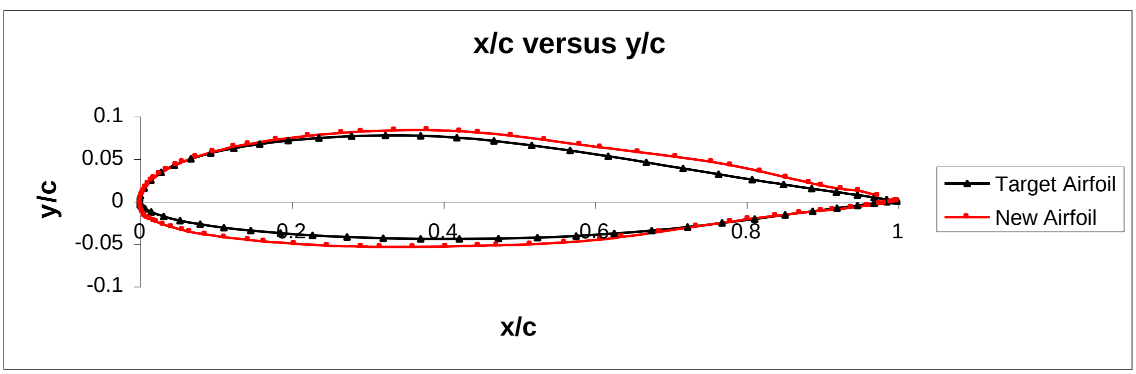
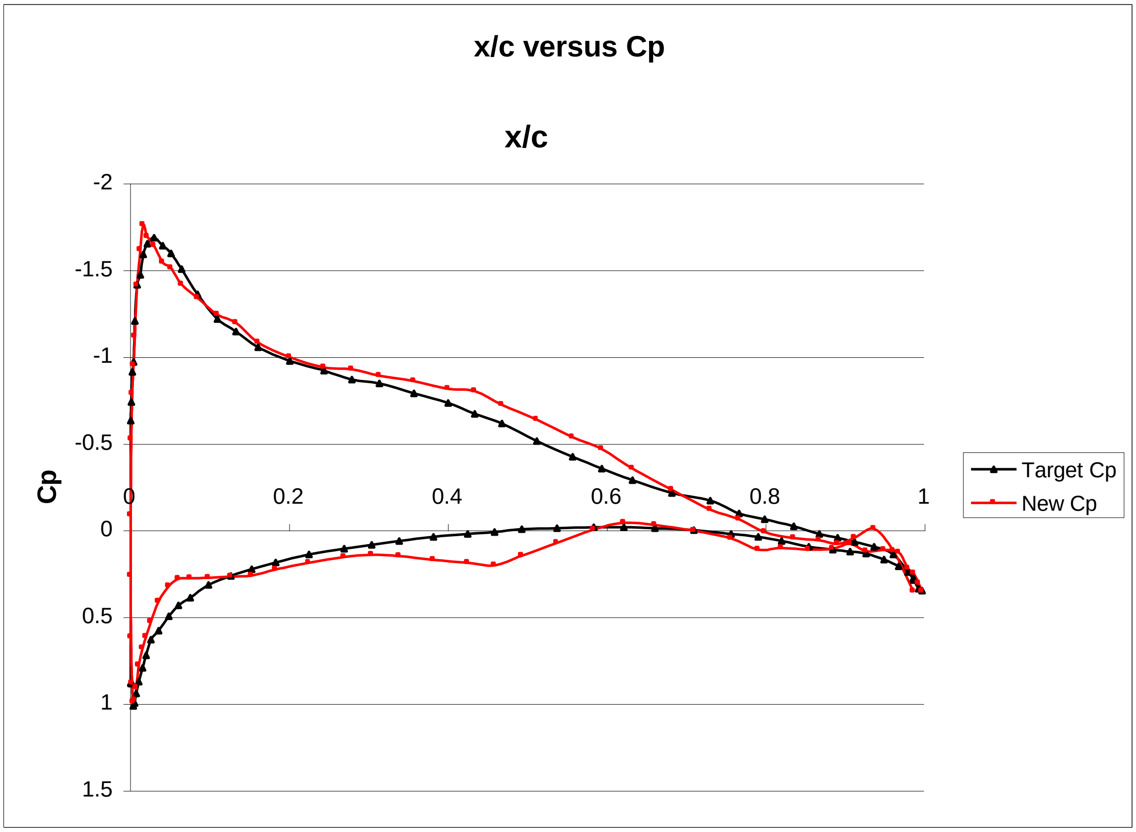


Figure A4 Results of genetic algorithm after generation 1200

Generation 1800:



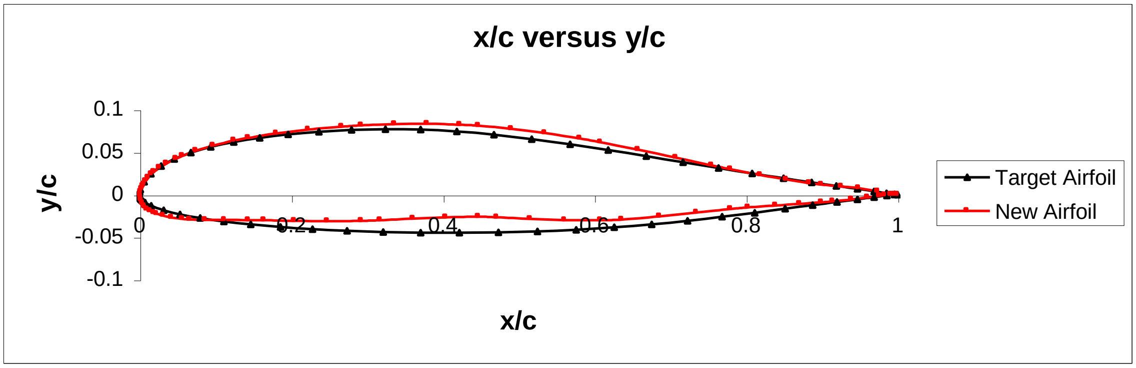
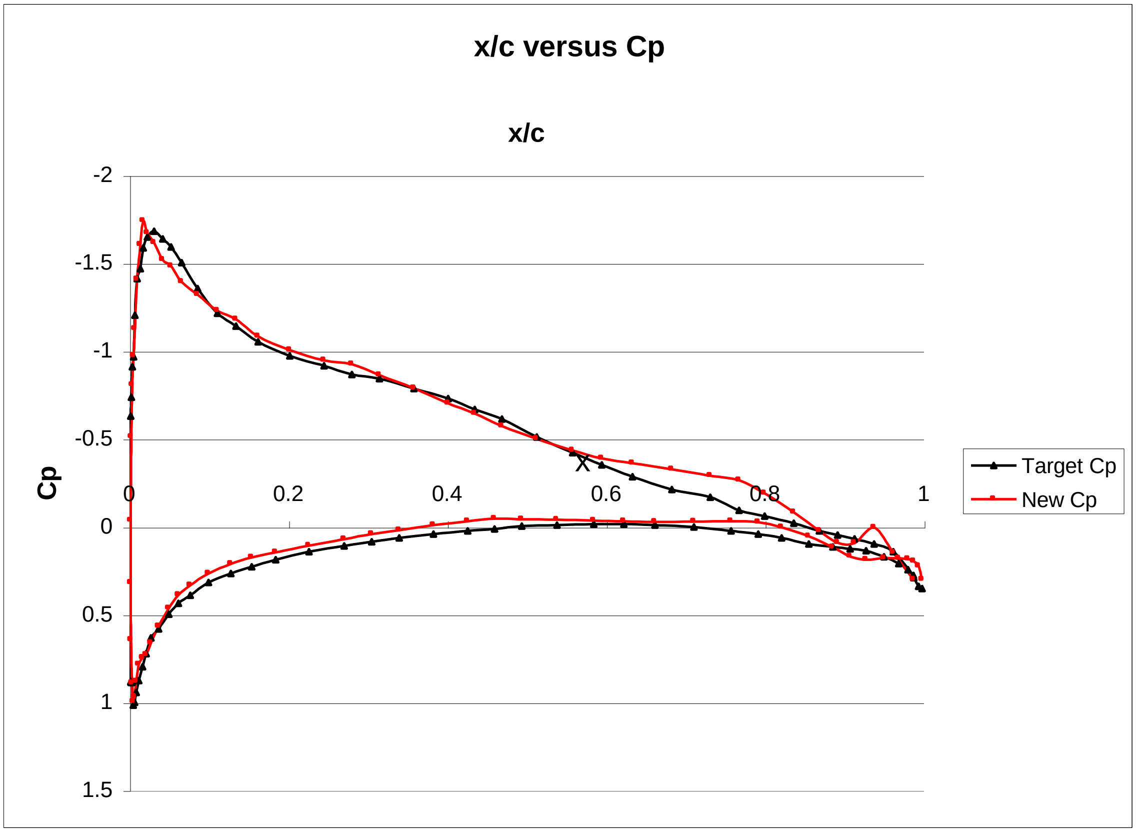


Figure A5 Results of genetic algorithm after generation 1800

Generation 2500:



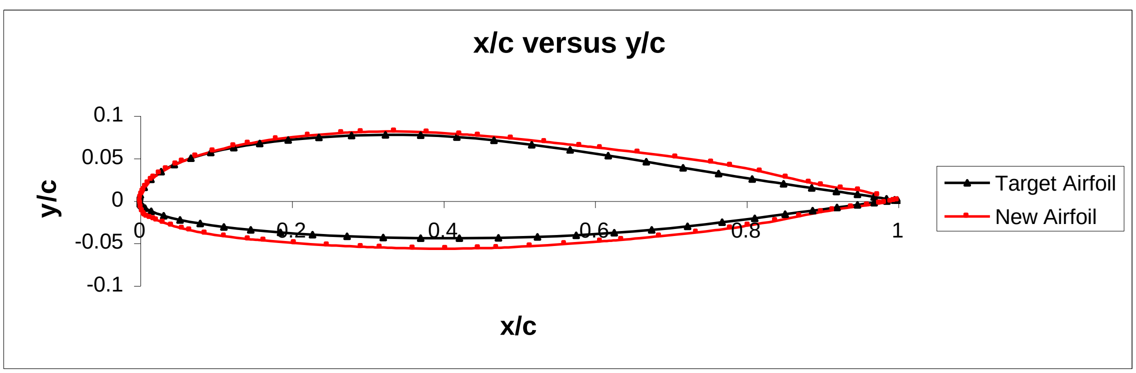
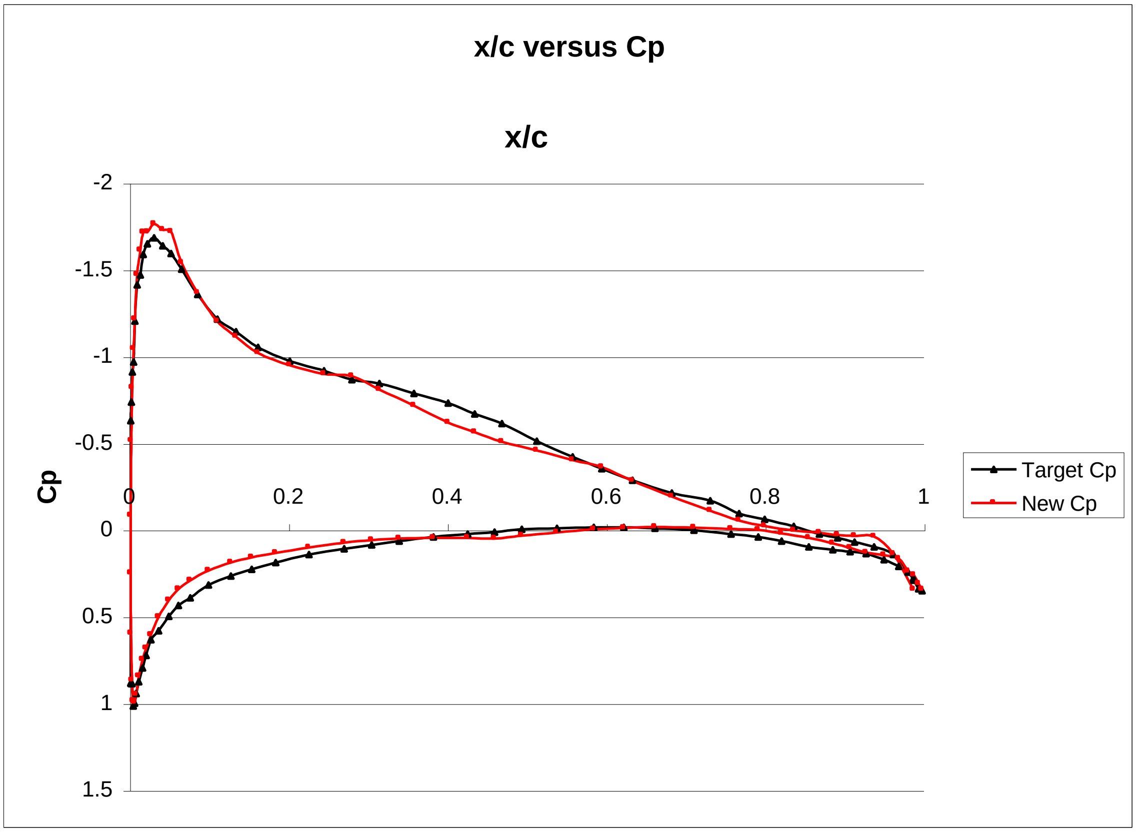


Figure A6 Results of genetic algorithm after generation 2500

Generation 3500:



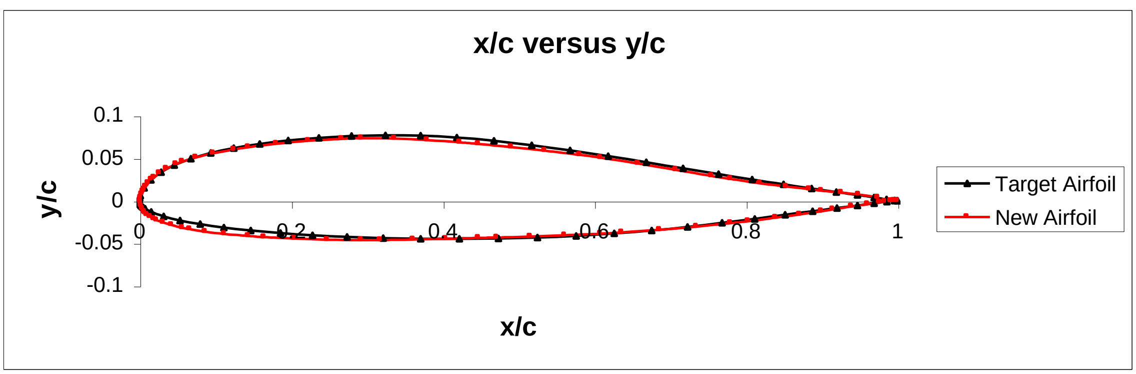
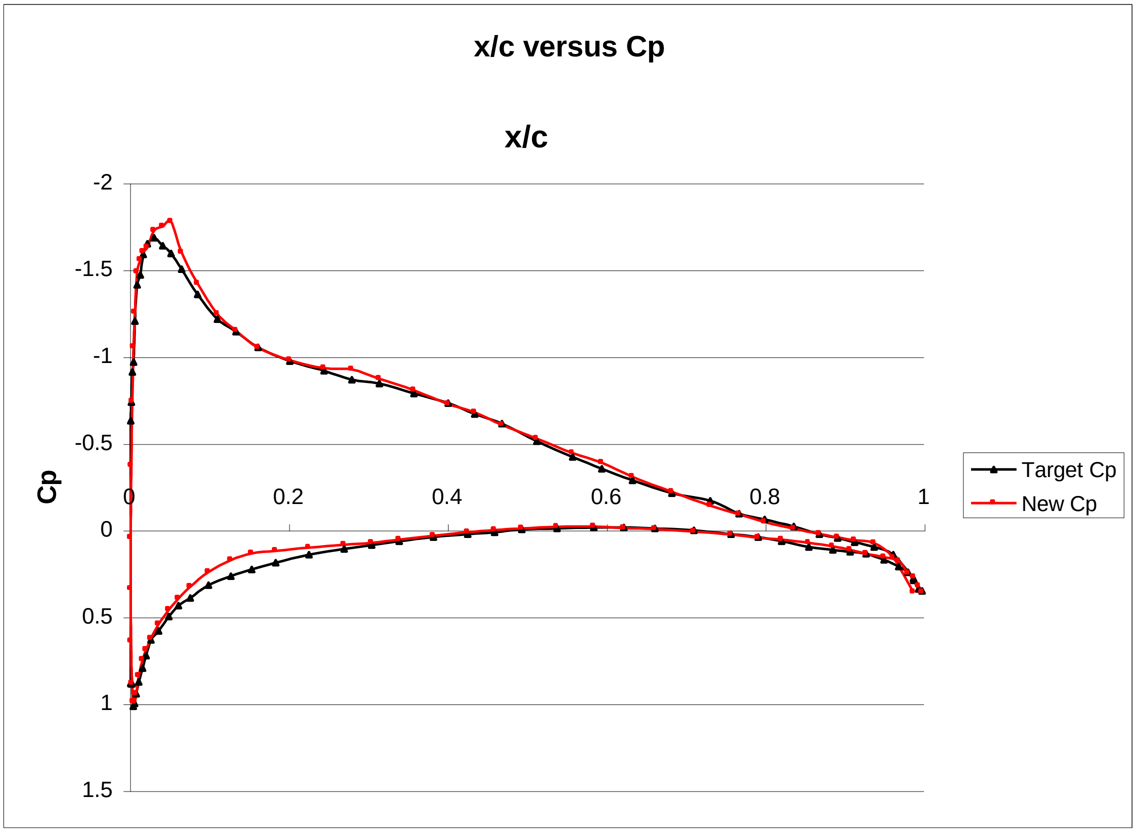


Figure A7 Results of genetic algorithm after generation 3500

Generation 4000:



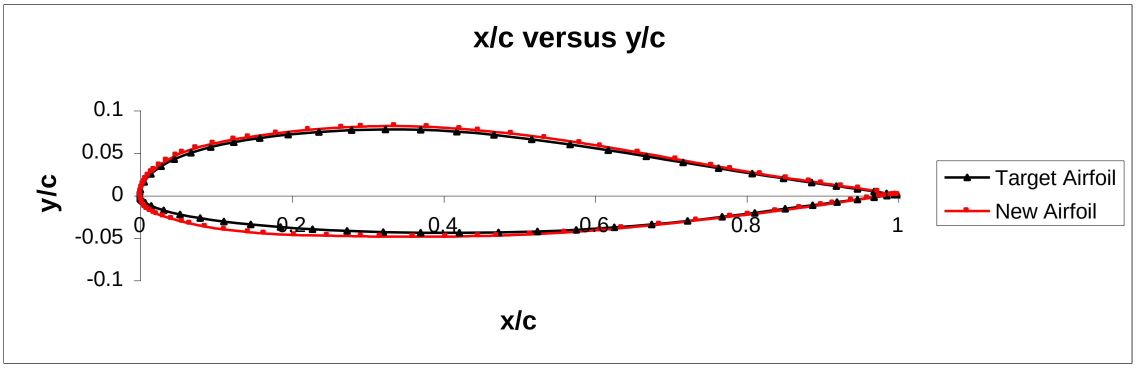
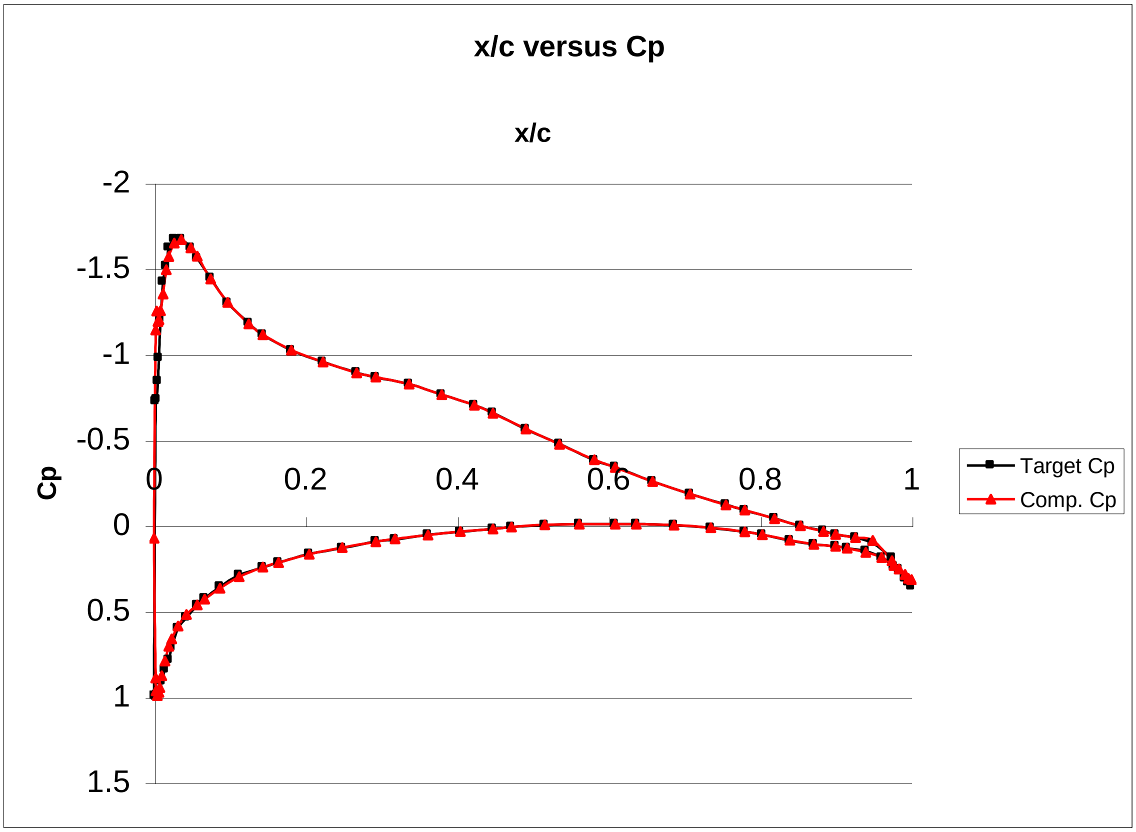


Figure A8 Results of genetic algorithm after generation 4000

A 2 **Results of Inverse Design with Residual Correction Algorithm**

A = B = C = 1

For the first criteria, iteration 80:



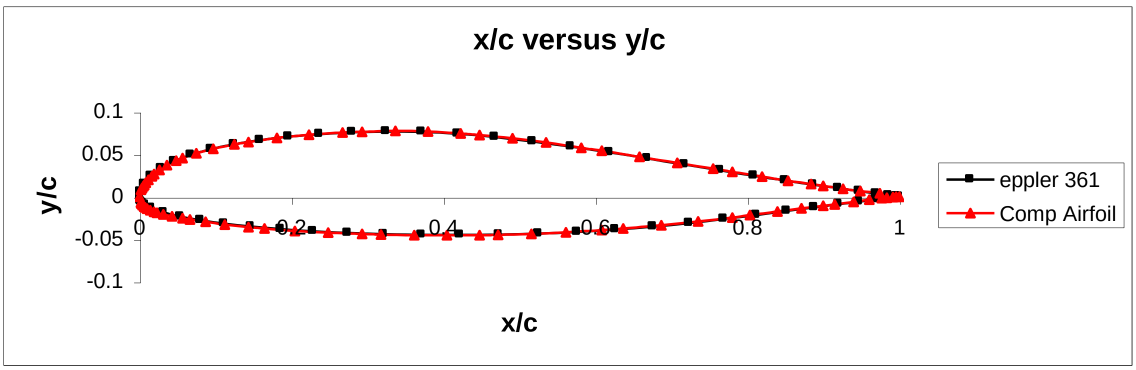
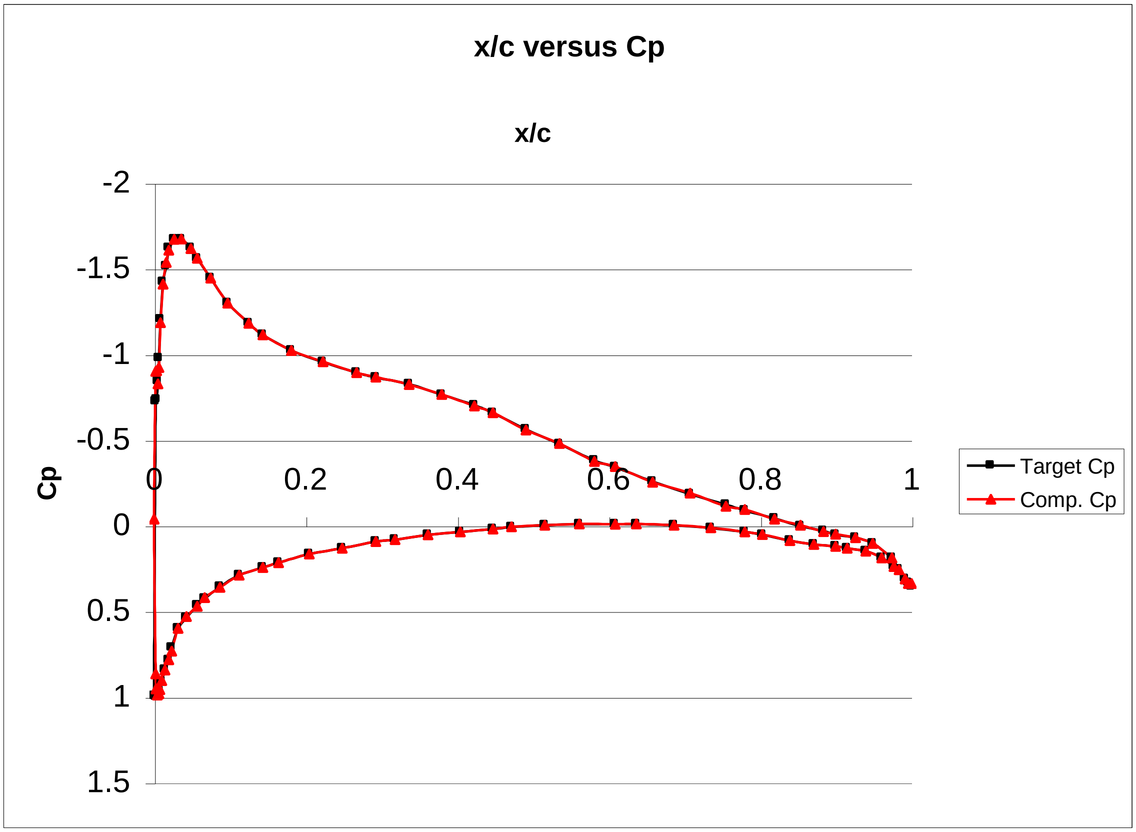


Figure A9 Results of residual correction algorithm after iteration 80 (A, B, C=1)

For the second criteria, iteration 977:



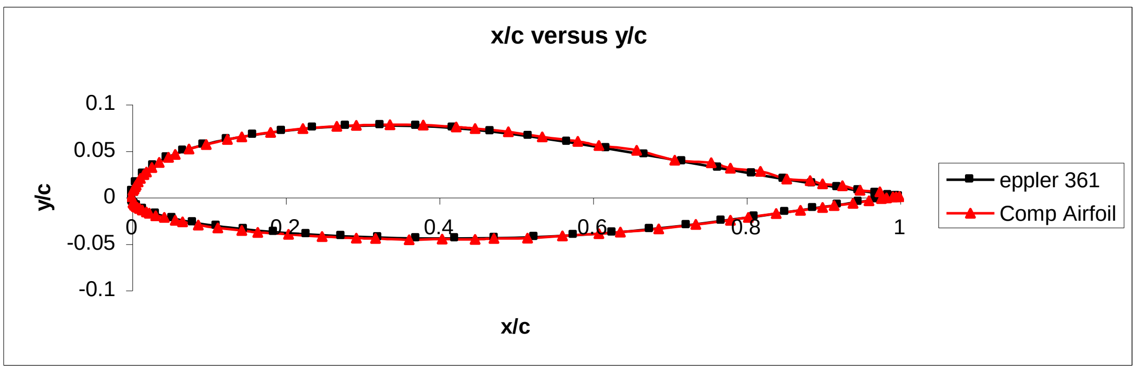
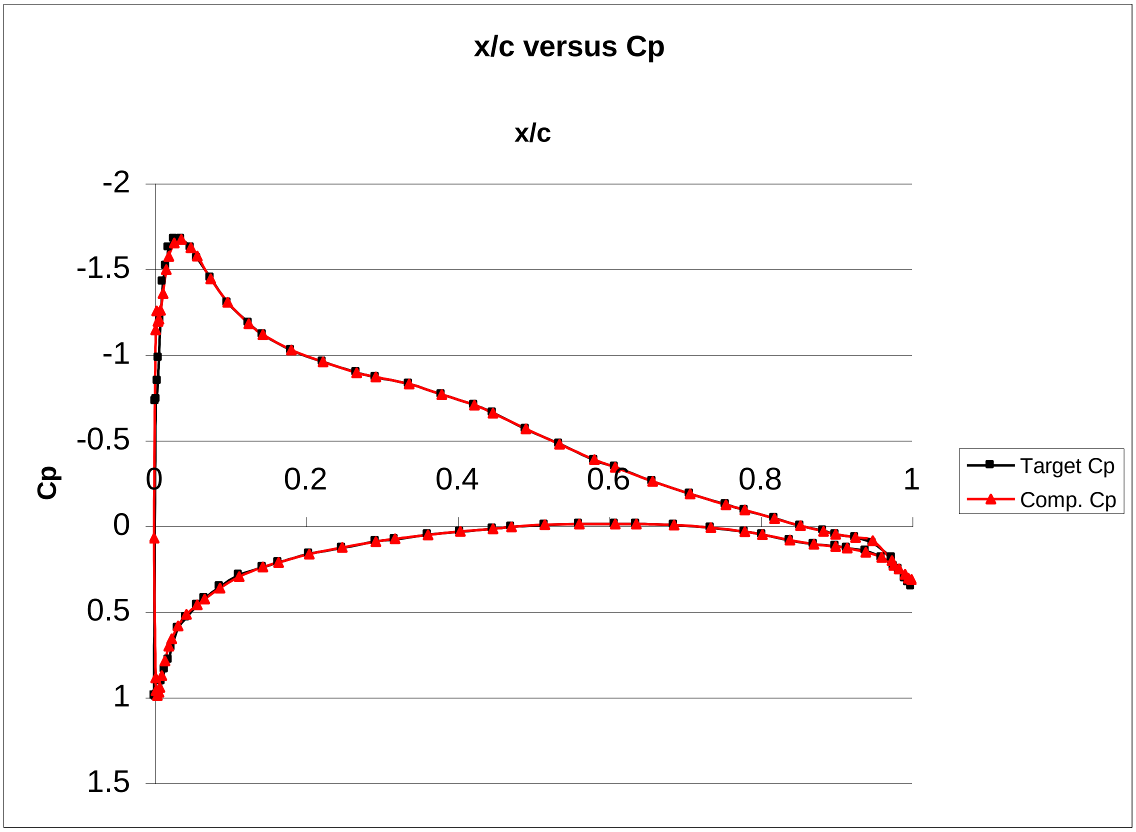


Figure A.10 Results of residual correction algorithm after iteration 977 (A, B, C = 1)

A = B = C = 4

For the first criteria, iteration 316:



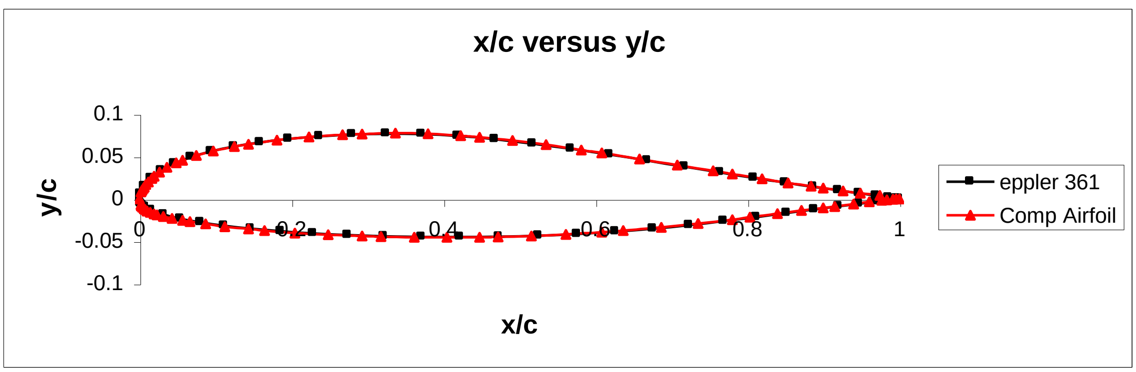
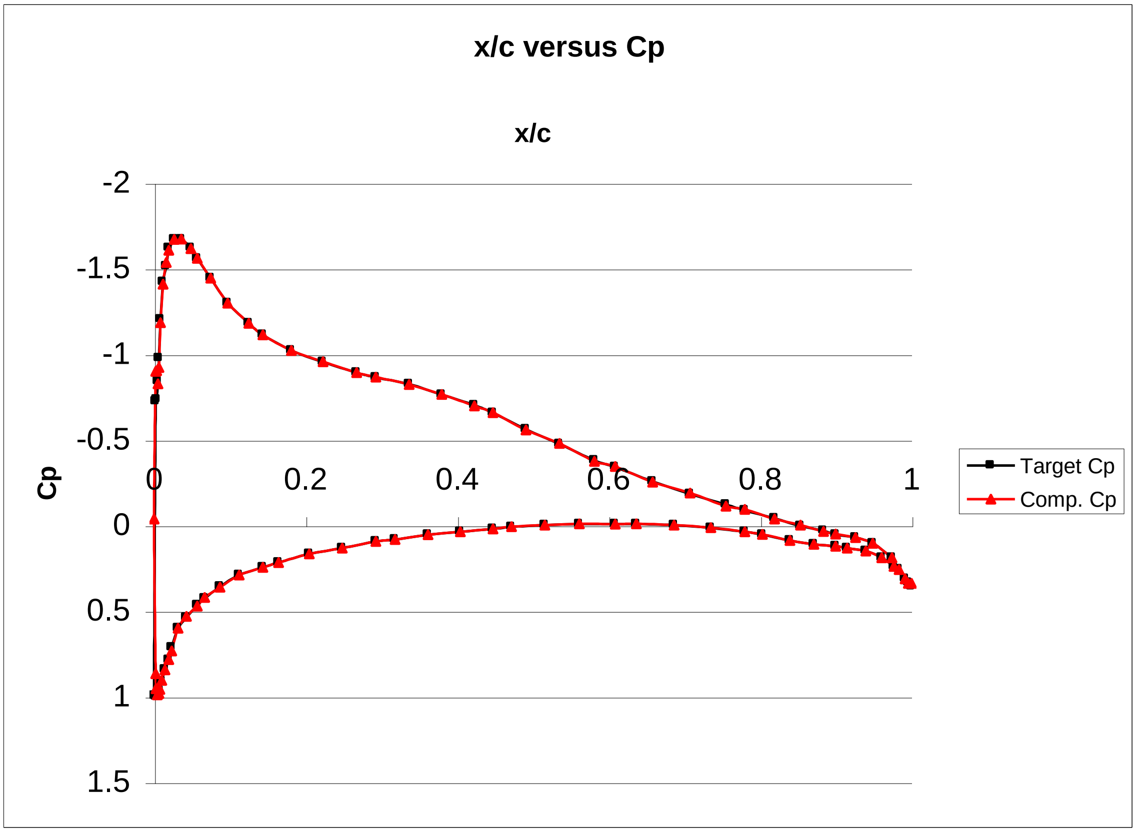


Figure A.11 Results of residual correction algorithm after iteration 316 (A, B, C = 4)

For the second criteria, iteration 3907:



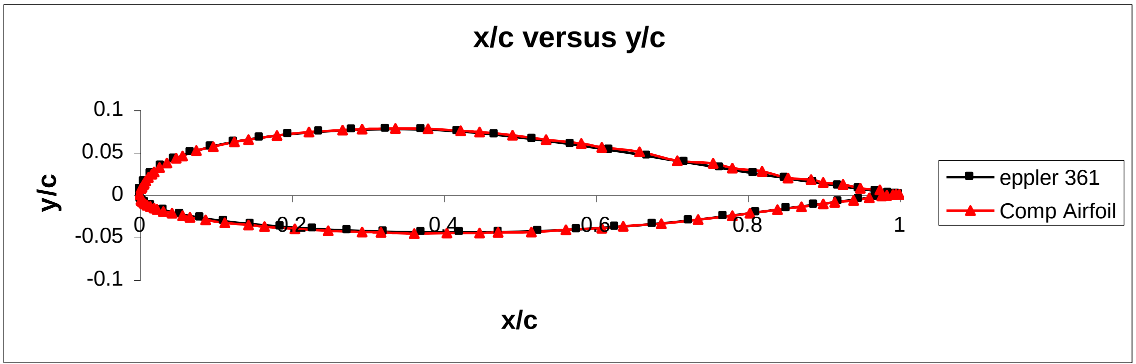
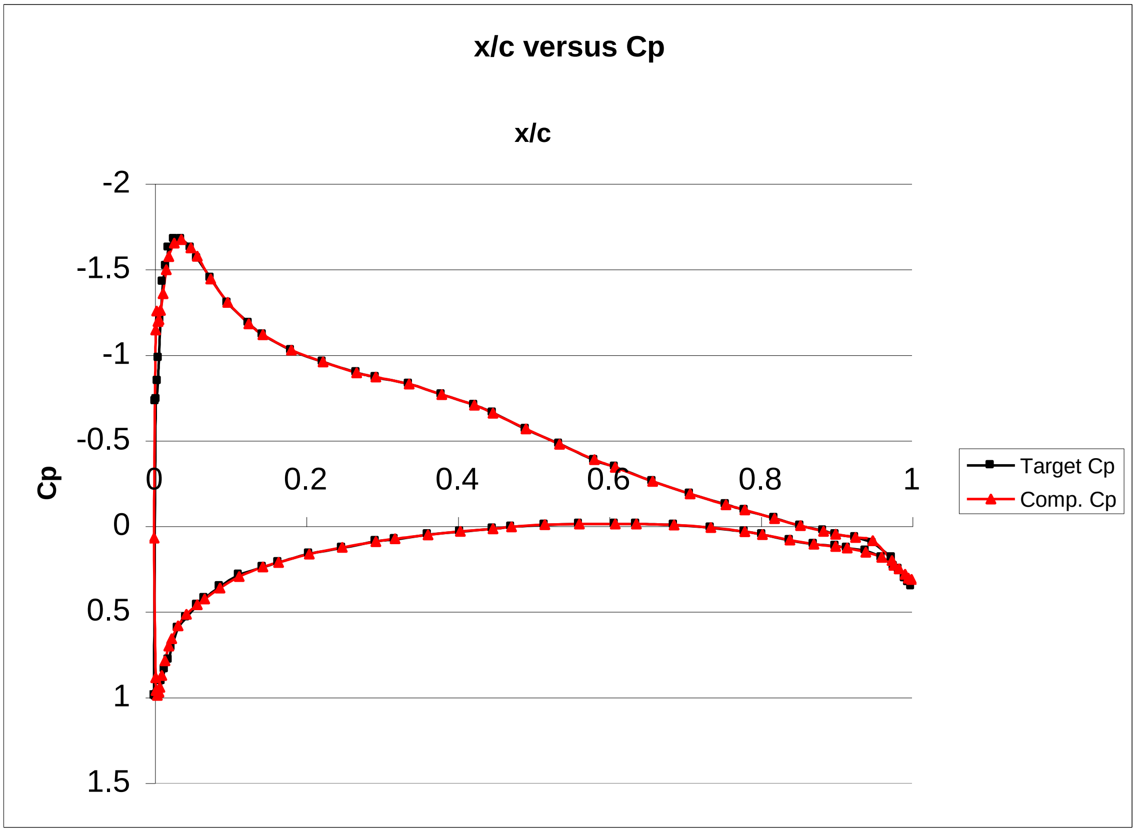


Figure A.12 Results of residual correction algorithm after iteration 3907 (A, B, C = 4)

A = B = C = 6

For the first criteria, iteration 474:



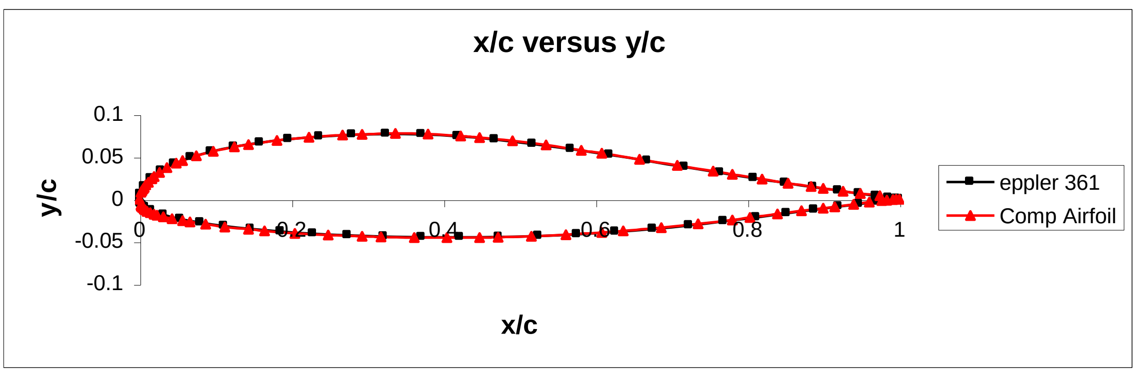


Figure A13 Results of residual correction algorithm after iteration 474 (A, B, C = 6)

For the second criteria, iteration 5860:

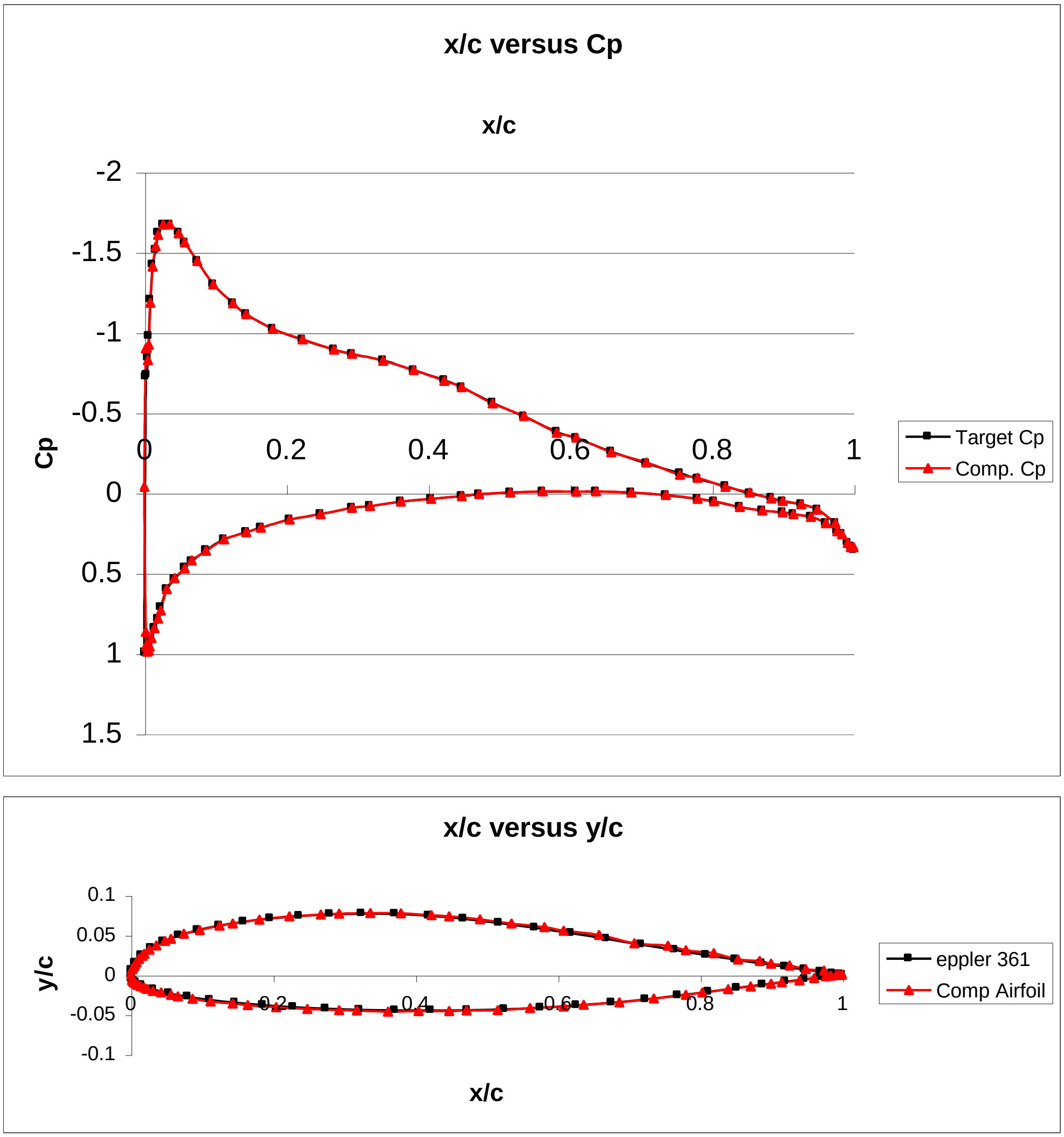
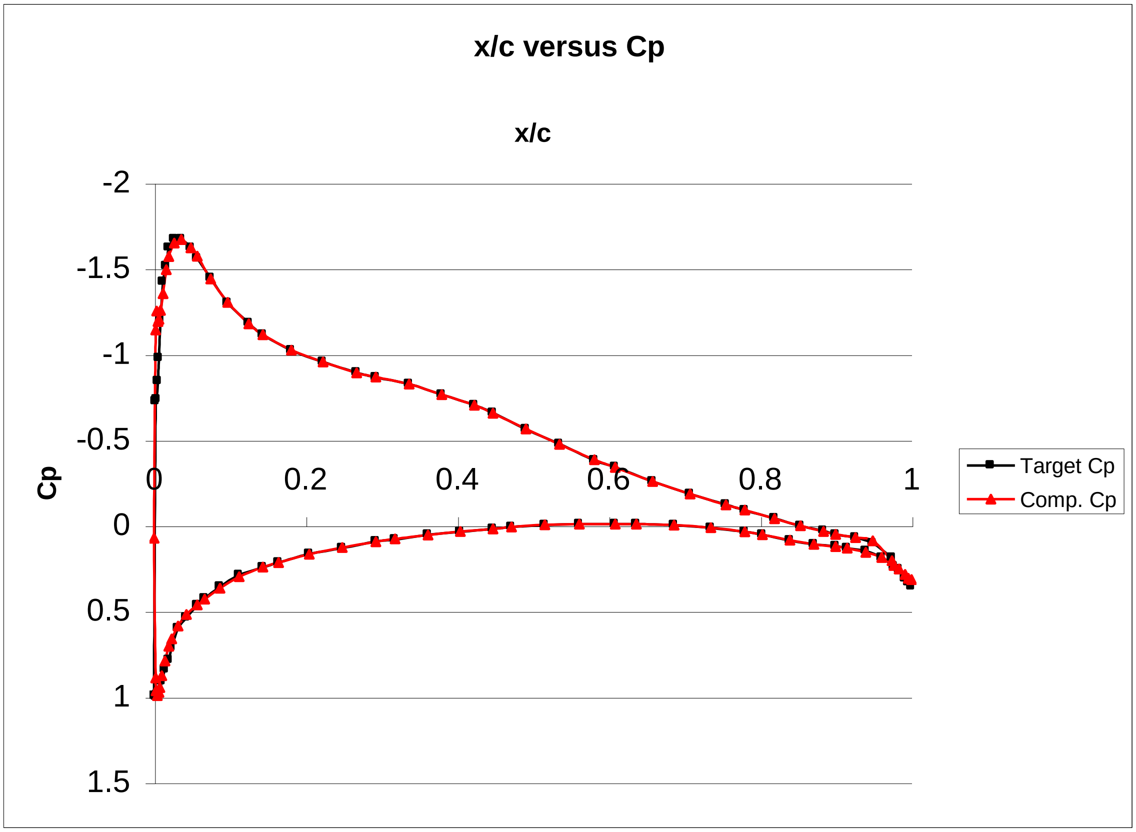


Figure A.14 Results of residual correction algorithm after iteration 5860 (A, B, C = 6)

A = B = C = 15

For the first criteria, iteration 1183:



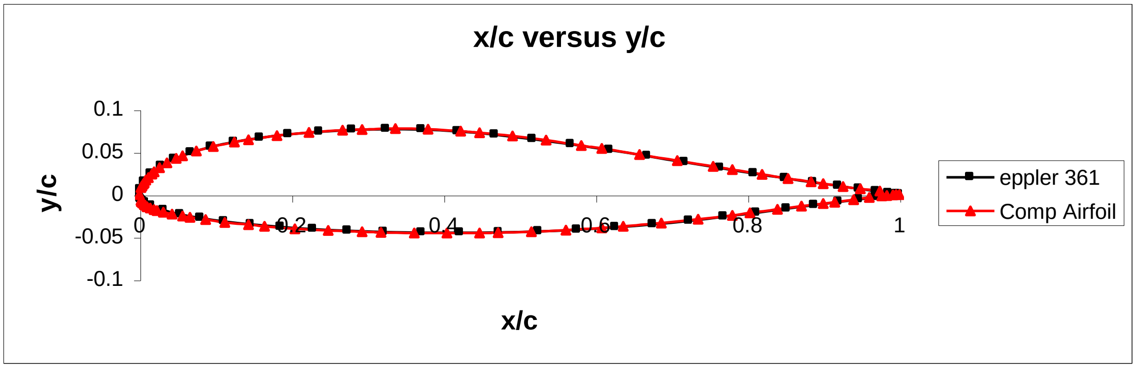
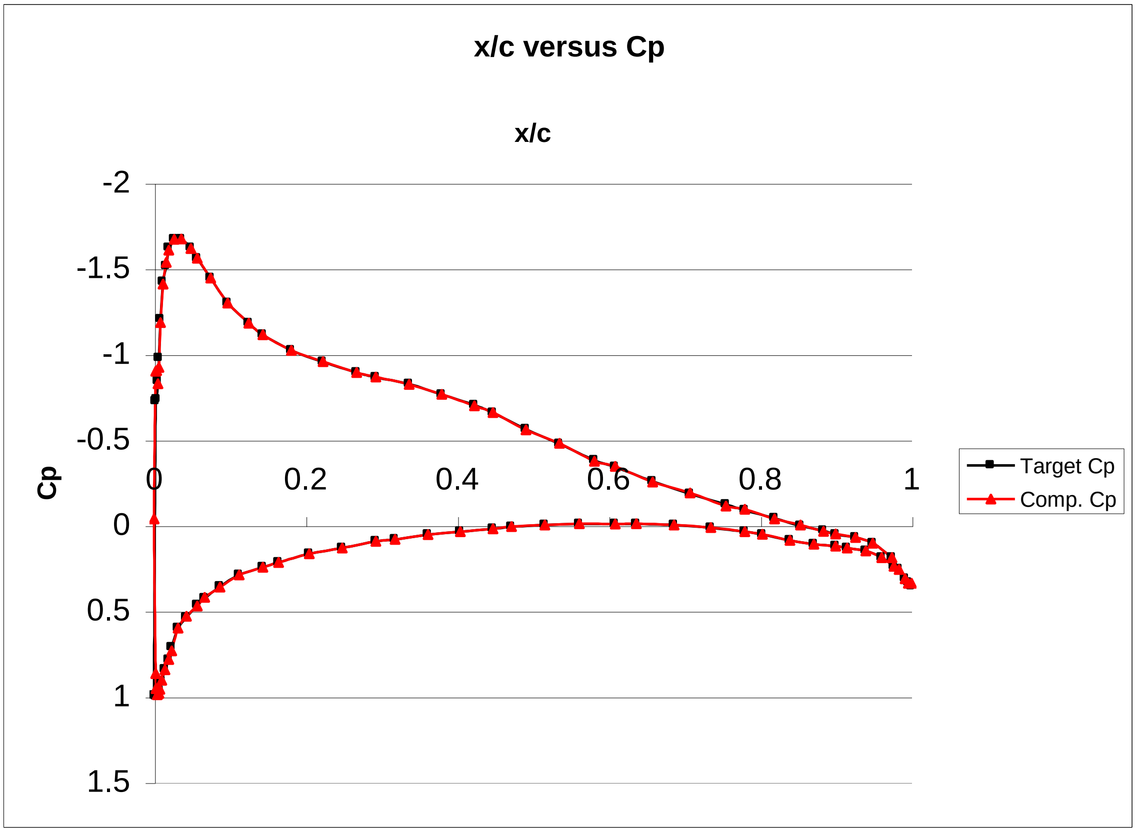


Figure A15 Results of residual correction algorithm after iteration 1183 (A, B, C = 15)

For the second criteria, iteration 14640:



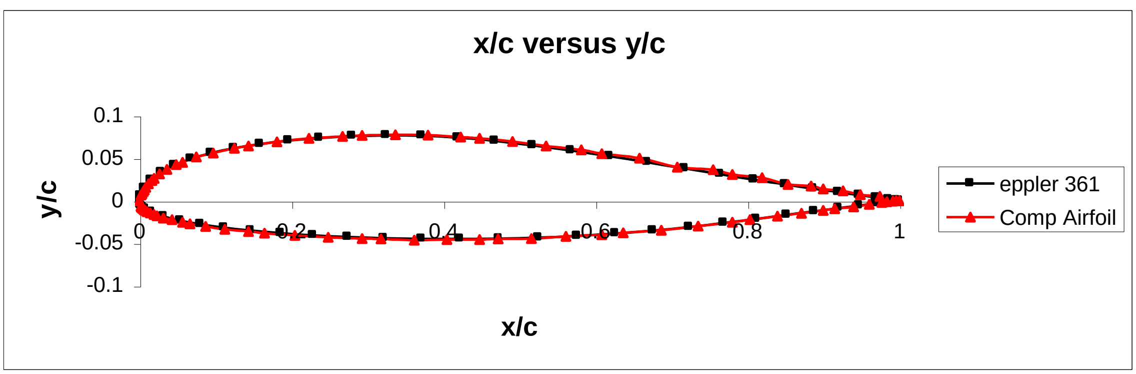


Figure A16 Results of residual correction algorithm after iteration 14640 (A, B, C = 15)

APPENDIX B

The fortran code of inverse design with residual correction algorithm is presented below. The fortran code of inverse design with genetic algorithm is also presented in a CD.

```
      PROGRAM RESIDUAL

      DIMENSION R(82), TVEL(84), DY(82), CVEL(84), BA(43,43), BU(39,39)
      DIMENSION BAA(43), B(82,82), XA(43), XU(39), DYTOT(82), PANLEN(84)
      DIMENSION BAU(43), BAD(43), BUA(39), BUU(39), BUD(39), BAR(43), BUR(39)
      DIMENSION CCP(84), TCP(84)
      REAL TOTCP
      INTEGER COUNT
      INCLUDE 'COMMONS'
      OPEN(5,FILE='NACA0012XY2.DAT')
      OPEN(7,FILE='TARGET_VEL2.DAT')
      OPEN(9,FILE='Y2.DAT')
      OPEN(75,FILE='D2.DAT')
      OPEN(90,FILE='KO2.DAT')
      OPEN(91,FILE='TCP2.DAT')
      OPEN(92,FILE='CCP2.DAT')

      COUNT=0
      LDA=82
      IPATH=1
       K1=15
       K2=15
       K3=15

       DO I=1,82
       DYTOT(I)=0.
      END DO

      DO I=1,84
      READ(5,*) X(I),Y(I)
      END DO

      DO I=1,84
      READ(7,*) TVEL(I)
      TCP(I)=1-(TVEL(I))**2
      END DO

 25   CALL PANEL(X,Y)

      DO I=1,84
      PANLEN(I)=XLENG(I)
      END DO

      CCP(1)=(CP(1)*XLENG(84)+CP(84)*XLENG(1))/(XLENG(1)+XLENG(84))

      DO I=2,84
      CCP(I)=(CP(I-1)*XLENG(I)+CP(I)*XLENG(I-1))/(XLENG(I)+XLENG(I-1))
      END DO
      CVEL(I)=SQRT(1-CCP(I))
```

```
END DO

DO I=1,82
IF(I.EQ.1) THEN
B(I,I+1)=(K2/(X(I+2)-X(I+1)))-2*K3/((X(I+2)-X(I+1))*
1 (X(I+2)-X(I)))
B(I,I)=K1-K2/(X(I+2)-X(I+1))+(2*K3/(X(I+2)-X(I)))*
1 ((1/(X(I+2)-X(I+1)))+(1/(X(I+1)-X(I))))
 R(I)=((TVEL(I+1))**2)-((CVEL(I+1))**2)

ELSE IF(I.GT.1.AND.I.LT.43) THEN
B(I,I+1)=(K2/(X(I+2)-X(I+1)))-2*K3/((X(I+2)-X(I+1))*
1 (X(I+2)-X(I)))
B(I,I)=K1-K2/(X(I+2)-X(I+1))+(2*K3/(X(I+2)-X(I)))*
1 ((1/(X(I+2)-X(I+1)))+(1/(X(I+1)-X(I))))
B(I,I-1)=-2*K3/((X(I+1)-X(I))*(X(I+2)-X(I)))
R(I)=((TVEL(I+1))**2)-((CVEL(I+1))**2)

ELSE IF(I.EQ.43) THEN
B(I,I)=K1-K2/(X(I+2)-X(I+1))+(2*K3/(X(I+2)-X(I)))*
1 ((1/(X(I+2)-X(I+1)))+(1/(X(I+1)-X(I))))
B(I,I-1)=-2*K3/((X(I+1)-X(I))*(X(I+2)-X(I)))
R(I)=((TVEL(I+1))**2)-((CVEL(I+1))**2)

ELSE IF(I.EQ.44) THEN
B(I,I+1)=(K2/(X(I+3)-X(I+2)))-2*K3/((X(I+3)-X(I+2))*
1 (X(I+3)-X(I+1)))
B(I,I)=K1-K2/(X(I+3)-X(I+2))+(2*K3/(X(I+3)-X(I+1)))*
1 ((1/(X(I+3)-X(I+2)))+(1/(X(I+2)-X(I+1))))
R(I)=((TVEL(I+2))**2)-((CVEL(I+2))**2)

ELSE IF(I.GT.44.AND.I.LT.82) THEN
B(I,I+1)=(K2/(X(I+3)-X(I+2)))-2*K3/((X(I+3)-X(I+2))*
1 (X(I+3)-X(I+1)))
B(I,I)=K1-K2/(X(I+3)-X(I+2))+(2*K3/(X(I+3)-X(I+1)))*
1 ((1/(X(I+3)-X(I+2)))+(1/(X(I+2)-X(I+1))))
B(I,I-1)=-2*K3/((X(I+2)-X(I+1))*(X(I+3)-X(I+1)))
R(I)=((TVEL(I+2))**2)-((CVEL(I+2))**2)

ELSE IF(I.EQ.82) THEN
B(I,I)=K1-K2/(X(I+3)-X(I+2))+(2*K3/(X(I+3)-X(I+1)))*
1 ((1/(X(I+3)-X(I+2)))+(1/(X(I+2)-X(I+1))))
B(I,I-1)=-2*K3/((X(I+2)-X(I+1))*(X(I+3)-X(I+1)))
R(I)=((TVEL(I+2))**2)-((CVEL(I+2))**2)

END IF
END DO

DO I=1,43
DO J=1,43
BA(I,J)=0.
END DO
END DO

DO I=1,43
DO J=1,43
```

```
BA(I,J)=B(I,J)
END DO
END DO

DO I=1,39
DO J=1,39
BU(I,J)=B(I+43,J+43)
END DO
END DO

 Z=1
 DO I=2,43
BAA(I)=BA(I,Z)
Z=Z+1
END DO

V=2
DO I=1,42
BAU(I)=BA(I,V)
V=V+1
END DO

DO I=1,43
BAD(I)=BA(I,I)
END DO

DO I=1,43
BAR(I)=R(I)
END DO

 Z=1
 DO I=2,39
BUA(I)=BU(I,Z)
Z=Z+1
END DO

V=2
DO I=1,38
BUU(I)=BU(I,V)
V=V+1
END DO

DO I=1,39
BUD(I)=BU(I,I)
END DO

DO I=1,39
BUR(I)=R(I+43)
END DO


CALL TDMA(BAA,BAD,BAU,BAR,XA,43)
CALL TDMA(BUA,BUD,BUU,BUR,XU,39)

DO I=1,43
DY(I)=XA(I)
```

```
      END DO

      DO I=1,39
      DY(I+43)=XU(I)
      END DO

      DO I=1,82
      DYTOT(I)=DYTOT(I)+DY(I)
      END DO

      DO I=1,82
      IF(I.LE.43) THEN
      Y(I+1)=Y(I+1)+(-DY(I))

      ELSE IF(I.GT.43) THEN
      Y(I+2)=Y(I+2)+DY(I)
      END IF
      END DO

      COUNT=COUNT+1

 500  CONTINUE

      TOTCP=0.
      DO I=1,84
      TOTCP=TOTCP+((ABS(TCP(I))-ABS(CCP(I)))**2)
      END DO

      IF (TOTCP.GT.1.) THEN
      GOTO 25
      ELSE
      GOTO 30
      END IF

 30   DO I=1,84
      WRITE(90,*)I,X(I),Y(I)
      WRITE(91,*)I,X(I),TCP(I),TOTCP,COUNT
      WRITE(92,*)I,X(I),CCP(I),TOTCP,COUNT
      END DO
      DO I=1,82
      WRITE(100,*)I,DYTOT(I)
      END DO

      END PROGRAM
**************************************************************************

      SUBROUTINE PANEL(X,Y)

       INCLUDE 'COMMONS'
C---
      OPEN (UNIT=6,FILE='PANELXY.OUT')
      OPEN (UNIT=7,FILE='AIRFOIL')
      OPEN (UNIT=8,FILE='XYVPXY.DAT')
      OPEN(UNIT=10,FILE='B.DAT')
      OPEN(UNIT=11,FILE='VELOCITY.OUT')
```

```
C--
      CALL AIRFOIL          ! CONSTRUCT THE AIRFOIL
C--
      CALL AMAT                                ! EVALUATE THE COEFFICIENT MATRIX A
      CALL GAUSS(1)                     ! SOLVE BY GAUSSIAN ELIMINATION
      CALL VELCP           ! VELOCITY & PRESSURE DIST. (CP)
      CALL DLMCAL           ! DRAG, LIFT & MOMENT COEFFICIENTS
C--
      CALL OUTPUT
C--
      DO I=5,8
       CLOSE(I)
      ENDDO
C--
      RETURN
      END

************************************************************************
      SUBROUTINE AIRFOIL
C--
C--  SETS UP COORDINATES
C--
      INCLUDE 'COMMONS'
      PI=4.0*ATAN(1.)
      PI2INV=0.5/PI
          ALPHA=5.0
      COSALF=COS(ALPHA*PI/180.0)
      SINALF=SIN(ALPHA*PI/180.0)

          NODTOT=84                            ! TOTAL NO. PANELS
      X(NODTOT+1)=X(1)
      Y(NODTOT+1)=Y(1)
C--
C--  EVALUATE SLOPES OF PANELS
C--
      DO 200 I=1,NODTOT
      DX=X(I+1)-X(I)
      DY=Y(I+1)-Y(I)
      DIST=SQRT(DX*DX+DY*DY)
C--
      XLENG(I)=DIST              ! PANEL LENGTHS
C--
      SINTHE(I)=DY/DIST              ! PANEL SLOPE INFO.
      COSTHE(I)=DX/DIST
C--
 200  CONTINUE
C--
C--  SET NODAL CONNECTIVITY OF PANELS
C--
      DO K=1,NODTOT
        LNOD1(K)=K              ! NODAL NOS OF EXTREME PTS
        LNOD2(K)=K+1
      ENDDO
      LNOD2(NODTOT)=1             ! CLOSE THE BOUNDARY
C--
C--  WRITE COORDINATES & LENGTHS ON DISC FILE
```

```
C---
      REWIND 7
      WRITE(7,*) NODTOT
C---
      DO I=1,NODTOT
        WRITE(7,*) I,X(I),Y(I)    ! EXTREME PT. COORDINATES
      ENDDO
C---
      DO K=1,NODTOT
        WRITE(7,*) K,LNOD1(K),LNOD2(K),XLENG(K)  ! CONNECT. & LENGTH
      ENDDO
C---
      DO K=1,NODTOT
       I1=LNOD1(K)              ! PANEL'S EXTREME POINTS
       I2=LNOD2(K)
        X1=X(I1)                ! COORDINATES
        Y1=Y(I1)
        X2=X(I2)
        Y2=Y(I2)
        XM(K)=0.5*(X1+X2)       ! MID-POINT COORDINATES
        YM(K)=0.5*(Y1+Y2)
      ENDDO
C---
      RETURN
      END
**************************************************************************
      SUBROUTINE AMAT
C---
C---  SET COEFFIENTS OF LINEAR SYSTEM
C---
      INCLUDE 'COMMONS'
C---
      KUTTA=NODTOT+1
C---
C---  INITIALIZE COEFFICIENTS
C---
      DO 90 J=1,KUTTA
 90   A(KUTTA,J)=0.0
C---
C---  SET VN=0 AT MIDPOINT OF ITH PANEL
C---
      DO 120 I=1,NODTOT
      XMID=XM(I)
      YMID=YM(I)
      A(I,KUTTA)=0.0
C---
C---  FIND CONTRIBUTION OF JTH PANEL
C---
      DO 110 J=1,NODTOT
      FLOG=0.0
      FTAN=PI
C---
      IF(J.EQ.I) GOTO 100
      DXJ=XMID-X(J)
      DXJP=XMID-X(J+1)
      DYJ=YMID-Y(J)
```

```
      DYJP=YMID-Y(J+1)
      FLOG=0.5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN=ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
C---
 100  CTIMTJ=COSTHE(I)*COSTHE(J)+SINTHE(I)*SINTHE(J)
      STIMTJ=SINTHE(I)*COSTHE(J)-SINTHE(J)*COSTHE(I)
      A(I,J)=PI2INV*(FTAN*CTIMTJ+FLOG*STIMTJ)
      B=PI2INV*(FLOG*CTIMTJ-FTAN*STIMTJ)
      A(I,KUTTA)=A(I,KUTTA)+B
      IF(I.GT.1.AND.I.LT.NODTOT) GOTO 110
C---
C--- IF ITH PANEL TOUCHES THE TRAILING EDGE
C---  ADD CONTRIBUTION TO KUTTA CONDITION
C---
      A(KUTTA,J)=A(KUTTA,J)-B
      A(KUTTA,KUTTA)=A(KUTTA,KUTTA)+A(I,J)
 110  CONTINUE
C---
C---  FILL IN KNOWN SIDES
C---
      A(I,KUTTA+1)=SINTHE(I)*COSALF-COSTHE(I)*SINALF
 120  CONTINUE
      A(KUTTA,KUTTA+1)=-(COSTHE(1)+COSTHE(NODTOT))*COSALF
     1          -(SINTHE(1)+SINTHE(NODTOT))*SINALF
C---
      RETURN
      END
**********************************************************************
      SUBROUTINE VELCP
C---
C---  EVALUATE VELOCITY & PRESSURE DISTRIBUTIONS
C---
      INCLUDE 'COMMONS'
C---
      DIMENSION Q(300)
C---
C---  RETRIEVE SOLUTION FROM A-MATRIX
C---
      DO 50 I=1,NODTOT
 50   Q(I)=A(I,KUTTA+1)
      GAMA=A(KUTTA,KUTTA+1)         ! VORTEX VALUE
C---
C---  FIND VTANG & CP AT MIDPOINT OF ITH PANEL
C---
      DO 130 I=1,NODTOT
      XMID=XM(I)
      YMID=YM(I)
      VTANG=COSALF*COSTHE(I)+SINALF*SINTHE(I)
C---
C---  ADD CONTRIBUTIONS OF JTH PANEL
C---
      DO 120 J=1,NODTOT
      FLOG=0.0
      FTAN=PI
C---
      IF(J.EQ.I) GOTO 100          ! SINGULARITY
```

```
      DXJ=XMID-X(J)
      DXJP=XMID-X(J+1)
      DYJ=YMID-Y(J)
      DYJP=YMID-Y(J+1)
      FLOG=0.5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN=ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
C---
 100  CTIMTJ=COSTHE(I)*COSTHE(J)+SINTHE(I)*SINTHE(J)
      STIMTJ=SINTHE(I)*COSTHE(J)-SINTHE(J)*COSTHE(I)
      AA=PI2INV*(FTAN*CTIMTJ+FLOG*STIMTJ)
      B=PI2INV*(FLOG*CTIMTJ-FTAN*STIMTJ)
      VTANG=VTANG-B*Q(J)+GAMA*AA
 120  CONTINUE
C---
      VELOC(I)=VTANG                   ! TANGENTIAL VELOCITY
      CP(I)=1.0-VTANG*VTANG            ! PRESSURE COEFFICIENT
C---
 130  CONTINUE
C---
      RETURN
      END
************************************************************************
      SUBROUTINE DLMCAL
C---
C---  EVALUATE DRAG, LIFT AND MOMENT COEFFICIENTS (CD,CL,CM)
C---
     INCLUDE 'COMMONS'
C---
      CFX=0.0
      CFY=0.0
      CM=0.0
C---
      DO 100 I=1,NODTOT
      XMID=XM(I)
      YMID=YM(I)
      DX=X(I+1)-X(I)
      DY=Y(I+1)-Y(I)
      CFX=CFX+CP(I)*DY
      CFY=CFY-CP(I)*DX
      CM=CM+CP(I)*(DX*XMID+DY*YMID)
 100  CONTINUE
C---
      CD=CFX*COSALF+CFY*SINALF
      CL=CFY*COSALF-CFX*SINALF
C---
      RETURN
      END
************************************************************************
      SUBROUTINE OUTPUT
C---
     INCLUDE 'COMMONS'
C---

      WRITE(6,'(2X,A,I3)')'NODTOT =',NODTOT
      WRITE(6,'(2X,A,I5)')'NACA =', NACA
      WRITE(6,'(2X,A,F6.2)')'ALPHA =', ALPHA
```

```
C--
      WRITE (6,*) ' NODAL CONNECTIVITY AND LENGTHS OF PANELS :'
      DO K=1, NODTOT
        WRITE (6,*) K,LNOD1(K), LNOD2(K), XLENG(K)
      ENDDO
C--
      WRITE (6,*) ' COORDINATES OF POINTS '
      WRITE (6,*) '      I     X(I)            Y(I) '
      DO I =1, NODTOT
        WRITE (6,*) I, X(I), Y(I)
      ENDDO
C--
      WRITE (6,*) ' MID-POINT COORDINATES :'
      DO K=1, NODTOT
        WRITE (6,*) K,XM(K),YM(K)
      ENDDO
C--
C--  FOR 2-DIM POTENTIAL FLOWS,  DRAG = 0
C--
      WRITE (6,*) ' DRAG, LIFT AND MOMENT COEFFICIENTS :'
      WRITE (6,1000) CD,CL,CM
 1000 FORMAT (' CD =',F8.5,2X,' CL =',F8.5,2X,' CM =',F8.5,/)
C--
      WRITE (6,*) ' VELOCITY AND PRESSURE DISTRIBUTIONS :'
      DO K=1, NODTOT
        WRITE (6,*) K,VELOC(K),CP(K)
      ENDDO
          DO I =1, NODTOT
          WRITE(11,*) VELOC(I)
          END DO
          CLOSE(11)
C--  THE CONSTANT VORTEX
C--
      WRITE (6,'(//,2X,A,F8.4)') ' GAMA =', GAMA
C--
C--  WRITE ON DISC FILE :
C--
      REWIND 8
      DO I =1, NODTOT
        CPM=-CP(I)
        WRITE (8,*) XM(I), YM(I), VELOC(I), CPM
      ENDDO
C--
      RETURN
      END
***********************************************************************
      SUBROUTINE GAUSS (NRHS)
C--
C--  SOLUTION OF LINEAR SYSTEM OF EQNS BY GAUUS ELIMINATION
C--  BY PARTIAL PIVOTING
C--
C--  A = COEFFICIENT MATRIX
C--  NEQNS = NO. OF EQUATIONS
C--  NRHS = NO. OF RIGHT-HAND SIDES
C--
C--  RIGHT-HAND SIDES AND SOLUTIONS STORED IN
```

```
C--  COLUMNS NEQNS+1 THRU NEQNS+NRHS OF "A"
C--
     INCLUDE 'COMMONS'


C--
      NEQNS=NODTOT+1              !neqns=190        np=191  ntot=191
      NP=NEQNS+1
      NTOT=NEQNS+NRHS


C--
C--  GAUSS REDUCTION
C--
      DO 150 I=2,NEQNS
C--
C--  SEARCH FOR LARGEST ENTRY IN (I-1)TH COLUMN
C--  ON OR BELOW MAIN DIAGONAL
C--
     IM=I-1

     IMAX=IM
      AMAX=ABS(A(IM,IM))

      DO 110 J=I,NEQNS


     IF(AMAX.GE.ABS(A(J,IM))) GOTO 110
     IMAX=J
      AMAX=ABS(A(J,IM))


 110  CONTINUE


C--  SWITCH (I-1)TH AND IMAX TH EQUATIONS
C--
     IF(IMAX.NE.IM) GOTO 140
!          write(10,*)' AMAX,amax

      DO 130 J=IM,NTOT
      TEMP=A(IM,J)
      A(IM,J)=A(IMAX,J)
      A(IMAX,J)=TEMP
 130  CONTINUE
C--
C--  ELIMINATE (I-1)TH UNKNOWN FROM
C--  ITH THRU (NEQNS)TH EQUATIONS
C--
 140  DO 150 J=I,NEQNS
      R=A(J,IM)/A(IM,IM)
      DO 150 K=I,NTOT
 150  A(J,K)=A(J,K)-R*A(IM,K)


C--
```

```
C---   BACK SUBSTITUTION
C---
      DO 220 K=NP, NTOT
      A(NEQNS, K)=A(NEQNS, K)/A(NEQNS, NEQNS)
      DO 210 L=2, NEQNS
     I=NEQNS+1-L
     IP=I+1
      DO 200 J=IP, NEQNS
200   A(I, K)=A(I, K)-A(I,J)*A(J, K)
210   A(I, K)=A(I, K)/A(I,I)
220   CONTINUE
C---
      RETURN
      END
*************************************************************************
      SUBROUTINE TDMA(L,D,U,C,X,N)
     INTEGER ND,NI, NM1
      PARAMETER (ND=900)
      REAL L(ND), D(ND), U(ND), C(ND), X(ND), P(0:ND), Q(0:ND), TEMP

              L(1) = 0.0
      U(N) = 0.0

C* FORWARD ELIMINATION
      DO 10 I = 1, N
        Temp = D(I) + L(I)*P(I-1)
        P(I) = -U(I) / TEMP
        Q(I) = ( C(I) - L(I)*Q(I-1)) / TEMP
10    CONTINUE
C*
C*    BACK SUBSTITUTION
     do 20 I = N,1,-1
        X(I) = P(I)*X(I+1) + Q(I)
20    continue

C* OUTPUTTING THE SOLUTION VECTOR
      WRITE(6,*) '              THE SOLUTION VECTOR IS:'
      WRITE(6,*)
      WRITE(6,21) (X(I),I = 1,N)
21  FORMAT('',25x,f15.9)
      RETURN
      END
```

CURRICULUM VITAE

Bülent TUTKUN was born in Istanbul on August 11, 1974. He graduated from Bosphorus University, Business Administration, in 1995. He obtained his B.Sc. degree from Istanbul Technical University, Department of Astronautical Engineering, in 2000. Same year he started M.Sc. studies in Astronautical Engineering at Institute of Science and Technology of ITU. He has been employed as a research assistant by the Faculty of Aeronautics and Astronautics in 2001.