Computational Aerodynamic Design Optimization Using Reduced Order Modeling for Large Deformation Problems

Ceren GÜRKAN

Supervisor: Dr Ben J. Evans

Submitted to the Swansea University in fulfillment of the requirements for the Master of Science in Computational Mechanics

Swansea, 2012-2013

DECLARATION

This thesis work has not previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ……

Date ……

STATEMENT 1

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ……

Date ……

STATEMENT 2

I hereby give my consent for my thesis, if relevant and accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organizations.

Signed ……

Date ……

Abstract

Being different from current geometric optimization algorithms, in this thesis work, a novel optimization algorithm was developed, which aims to take any user defined arbitrary geometry and optimizes it with respect to any design variable.

Studies presented starts with validation of previously obtained results using 'Modified Cuckoo Search' geometric optimization algorithm, to optimize Bloodhound SSC air intake duct with 'Radial Basis Function' mesh movement algorithm, as well as 'Proper Orthogonal Decomposition' reduction order method to reduce the computational costs.

Extending the borders of 'Modified Cuckoo Search' algorithm using 'Radial Basis Function' mesh movement method; new implementation has done on arbitrary, unit circle starting geometry. Design variable defined as lift drag ratio on the circle boundary and algorithm was implemented with and without 'Proper Orthogonal Decomposition' reduction order method. Quality of results and computational costs of these two different implementations were compared.

Anne ve babama,

sonsuz güven ve destekleri için...

To my mum and dad,

for their endless support and trust...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

Taking part in such an outstanding master program widened my vision and changed my life for ever. In here, I would like to mention my acknowledgements to special people that I was extremely lucky to meet with them.

First, I would like to thank millions to my supervisor, Dr Ben Evans, it was a great opportunity for me to work with him throughout one year, his enthusiasm inspired and motivated me all the times.

Next comes my friends, they were closest to me despite being kilometers away. Semih Gönen, Hasan Gökhan Güler and Ali Erdi Genç, thank you for being in my life, in good or in in bad, whenever the times.

Thanks to all my family, for supporting me to go after my dreams with always being strong.

And finally my beloved, David Villamizar Duque, thank you for changing my life and always being there for me.

DEFINITIONS & ABBREVIATIONS

CFD	Computational Fluid Dynamics
RBF	Radial Basis Fucntion
POD	Proper Orthogonal Decomposition
MCS	Modified Cuckoo Search
CAD	Computer Aided Design
FDGD	Fast Dynamic Grid Deformation
LHS	Latin Hypercube Sampling
MMCS	Multiple Modified Cuckoo Search
2D	2 Dimensional
3D	3 Dimensional
Ma	Mach Number

CHAPTER 1: INTRODUCTION

1. INTRODUCTION

1.1.MOTIVATION: BLOODHOUND SSC

Throughout the history, humankind has been working to reach to the biggest, the highest and the fastest. Because of this -from birth- passion, you are holding this thesis in your hands. Taking a part on design of the 'world's fastest car' was an extremely exciting challenge and was the biggest motivation for the creation of this thesis.

Even the history of reaching to fastest relies far back on history; in October 1970 Gary Gabelich's Blue Flame ushered in the rocket era with 622.407 mph. Thirteen years later, Richard Noble's Thrust2 regained the record for Britain with 633.468 mph on October 1983, and things stayed stable till September 1997; the day that Thrust SSC reached 714.144 mph. To inspire new generations, here we are in a new adventure named Bloodhound SSC (SuperSonic Car) which aims to improve land speed record to 1000 mph (just over 1600 kph). See Figure 1.1 below.

Figure 1.1. Bloodhound SSC

This thesis presents the validation of Modified Cuckoo Search optimization algorithm [1] using Proper Orthogonal Decomposition [2] on air intake duct of Bloodhound SSC examined in subsonic, transonic and supersonic regimes as well as the evolution of an arbitrary geometry to simple aerofoil using the same technique.

Reader now invited to examine this supersonic car more closely, especially the air intake duct and its geometric evolution history; since in this work the concern was air intake duct

geometry. Air intake duct of Bloodhound SSC is located in front of the jet engine EUROJET EJ200 See Figure 1.2. Even if it looks quite different than the one that you have in your car; Eurojet EJ200 is still an internal combustion engine. Air is sucked into the engine, it is compressed fuel is added which then burns, and this expands pushing the exhaust out of the nozzle producing thrust. The quality of the air sucked into the engine is addressed by intake duct.

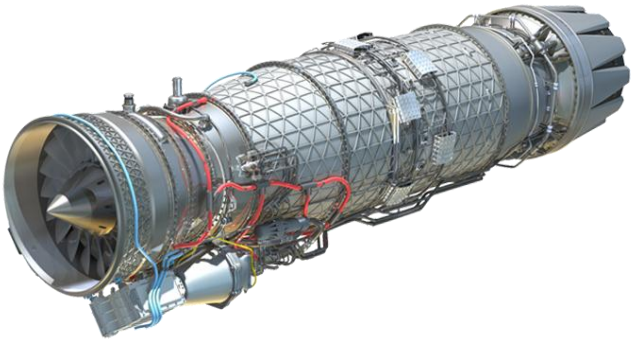


Figure 1.2. EJ200

The term 'quality' of the airflow here can be measured with different parameters such as total pressure distortion which is a measure of the magnitude of the variations in the total pressure (or total energy) across a plane slightly upstream of the engine compressor face, swirl, which is a measure of the mean deviation from a purely axial flow upstream from the compressor face, and flow stability, which is the degree to which the flow in the duct will fluctuate. Obviously it is a complex task, examining all this parameters with changing free stream conditions in all subsonic, transonic and supersonic regimes. This can be achieved in a short time scale with latest Computational Fluid Dynamics (CFD) techniques.

Here is the geometric evolution of the intake duct geometry to reach the best 'quality' air.

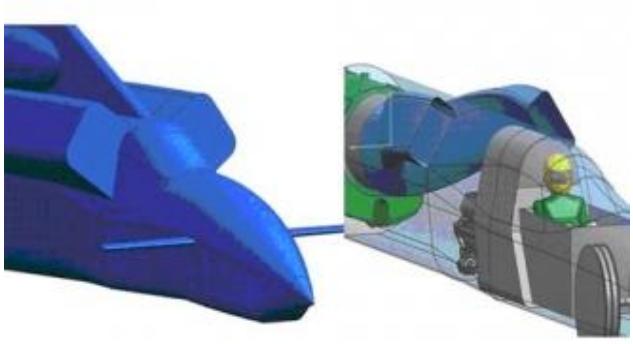


Figure 1.3. Initial Duct Geometry

The very initial duct geometry designed by Ron Ayers and Glynne Bowsher as a twin intake See Figure 1.3.

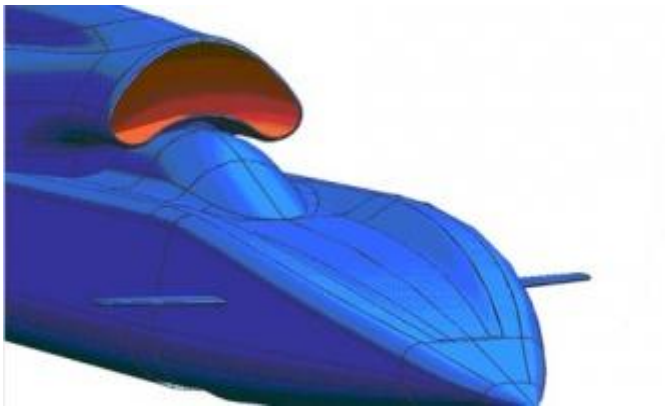


Figure 1.4. Latest version of duct geometry

Shortly it become apparent that this was not the clever way of designing the intake duct and the geometry is changed with a single intake duct geometry which follows the intake canopy. The latest version of the air intake duct can be seen in Figure 1.4 above. The duct geometry above was obtained only with the experience of Bloodhound engineers. Although, last year some optimization work using reduced order modelling conducted on it [3]; the results were not validated. This work includes the confirmation of reliability of the method used previously as well as extension of the same algorithm to optimization of arbitrary geometry, large deformation problems.

1.2. OBJECTIVES AND PROJECT AIM

This thesis work; which was supervised by Dr. Ben J. Evans -CFD engineer of Bloodhound SSC and lecturer in Swansea University- tested effectiveness of Modified Cuckoo Search (MCS) optimization algorithm together with Radial Basis Function (RBF) mesh movement method on different problems with and without Proper Orthogonal Decomposition (POD) reduced order method which was used to accelerate the convergence rate of MCS optimization algorithm.

Obviously in the literature, it is not new to use RBF's for mesh movement, or using reduction order methods to decrease the computational costs [4][5][6]. The uniqueness of this work presented; comes from the extension of current methodology in such a way that geometric optimization starts with an arbitrary geometry. Scientists tend to improve current geometries

with respect to any design parameter with geometric optimization techniques; however, when we have no idea about where to start, a long, time and money consuming trial and error process comes to the stage. At the end of this thesis work, a rough idea about the starting geometry was obtained, for the cases reader does not know where to start to the geometric optimization.

As a summary then; the first objective of this thesis was to validate MCS with/out POD algorithm on Bloodhound SSC air intake duct for different fluid regimes and secondly implementing the RBF mesh movement and MCS optimization algorithms on an arbitrary geometry with and without POD to test the robustness of this novel implementation.

The overall aim of this thesis was to create a general purpose optimization algorithm for arbitrary geometry large deformation aerodynamic design optimisation problems.

1.3. THE PROBLEM STATEMENT

As it is mentioned above; the first objective of this thesis was validating MCS with/out POD algorithm on Bloodhound SSC air intake duct. The target was minimizing the distortion function -at the engine inlet- which was defined in Equation 1.1 where $\boldsymbol{\Gamma}$ being the domain which distortion was computed (in 2D $\boldsymbol{\Gamma}$ is the line at the engine inlet), $\boldsymbol{L}$ is the length of the domain, $\mathbf{P}_t$ and $\overline{\mathbf{P}}_t$ being the total pressure and average pressure on $\boldsymbol{\Gamma}$ respectively.

Distortion was chosen to be the objective function for optimization simply because; among other variables that can define the air quality such as the swirl or the flow stability; distortion is considered as the most important parameter by aerodynamicists.

So the first problem statement was; find the optimum duct geometry that minimizes the distortion at the engine inlet. To achieve this, present code from last years' work [3] was modified for different fluid regimes and regenerated for without POD case.

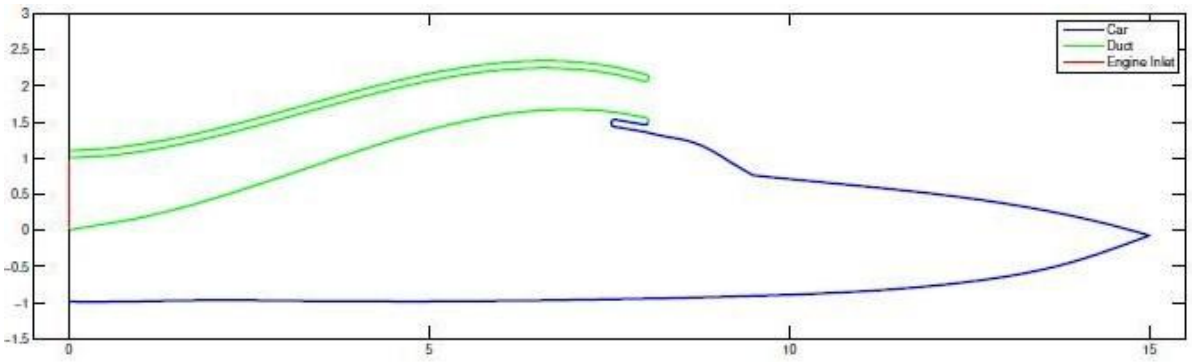


Figure 1.5. Bloodhound SSC air intake duct geometry

$$Distortion = D = \frac{\left(\int_{\Gamma} \mathrm{P}_t l \, dl\right)}{\bar{\mathrm{P}}_t \, L} \qquad \textbf{Equation 1.1}$$

After achieving the first aim, the new target became extending the methodology that worked for intake duct to a general purpose optimization algorithm. This general purpose geometric algorithm was created in such a way; that it can work for any starting geometry. In our case, the representative starting geometry was chosen as a unit circle.

To realize the second aim; overall MCS with/out POD algorithm coded on MATLAB environment and tested using university cluster. The design parameter selected to optimize was 'Lift-Drag Ratio' and fluid was viscous air, whereas the starting arbitrary geometry as mentioned above was a simple 1 unit radius circle; since it is known that circle has to evolve to a simple aerofoil shape to optimize the Lift-Drag ratio in viscous air.

So the second problem statement was; find the optimum geometry with a unit circle used as an arbitrary starting point, which maximize the Lift-Drag ratio on the boundary where the total lift and drag forces are defined with Equation 1.2 and Equation 1.3.

$$Li = \int pn * kdL \qquad \textbf{Equation 1.2}$$

$$D = \int pn * idL \qquad \textbf{Equation 1.3}$$

Where:

Li is the lift,

D is the drag,

L is the length of boundary,

p is the pressure value,

n is the unit normal vector pointing into the wing,

i is the unit vector parallel to the free stream direction and,

k is the vertical unit vector, normal to the free stream direction.

The procedure that was followed for the solution of above problems was presented in Figure 1.6.

Here in this chapter; presented the general overview of the procedure which was followed. In the next section, the theoretical aspects of the mesh movement and optimization algorithms were explained in detail.

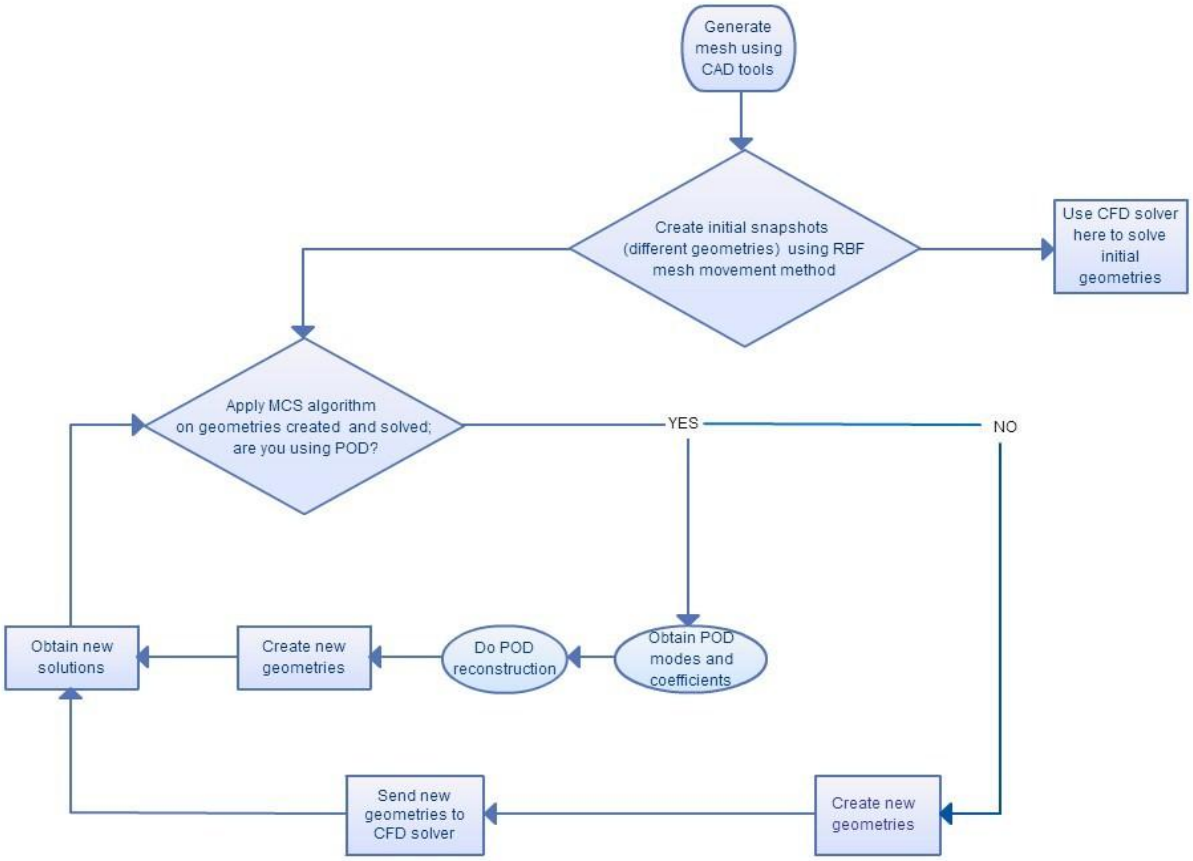


Figure 1.6. MCS with/out POD flowchart

CHAPTER 2: METHODOLOGY

2. METHODOLOGY

2.1 MESH MOVEMENT METHOD

The classical mesh movement approach in fluid-structure interaction problems would be generating the desired geometry in CAD environment, passing geometrical information to CFD system and evolving aerodynamic geometry with transferring geometrical data between CAD and CFD systems constantly. With RBF mesh movement method; however, no need of continuous transfer of data between different systems but rather; after the creation of initial computational mesh, aerodynamic geometry is controlled by displacement of "control nodes" on the boundary. The location and number of control nodes are crucial in that sense since they are going to be the ones deciding the boundary movement and accordingly the movement of whole mesh. Figure 2.1 below represents unit circle boundary geometry (blue) with 6 control nodes located on it (red).

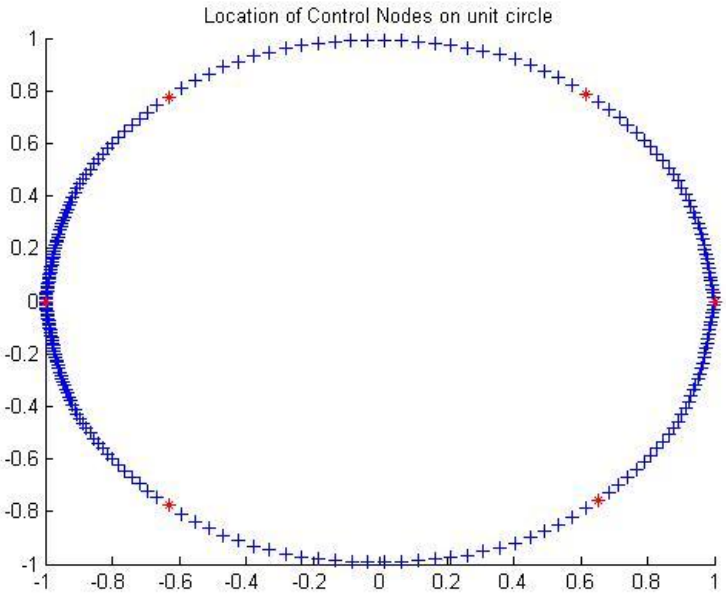
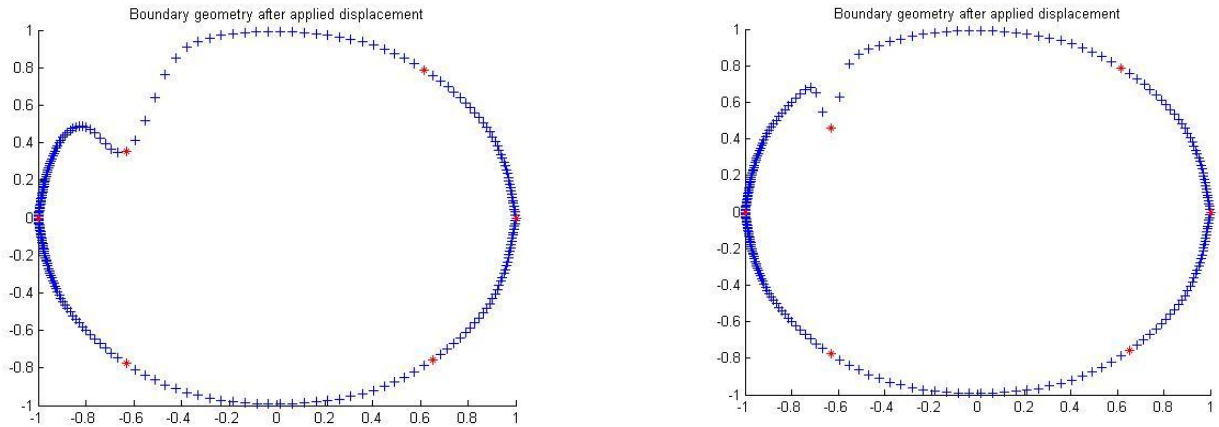


Figure 2.1. Unit circles with control points located on it

So called “control nodes” here are the specific nodes, located in the boundary which leads the movement of the other nodes on the boundary or so called “boundary nodes”. Displacement of boundary nodes are decided by RBF’s which can be defined in several ways. The feature desired for smooth boundary movement was lower displacement as drifted away from control nodes. Among many other options Gaussian RBF is chosen for mesh movement[1] which is defined by Equation 2.1 below where r is the distance between control node and boundary node and c is the RBF coefficient. Value of RBF is called as weight, so the idea was decreasing the weight with increasing distance. The RBF coefficient is a control measure for the distribution of the weight on surface. This means if c is a large number the weight distribution among the surface will not vary a lot and smooth surface will be guaranteed and in opposite case, sharp peaks would appear on the surface.

$$\Phi(\zeta) = e^{-r^2/c^2} \qquad \textbf{Equation 2.1}$$

A boundary node can be controlled by one or more than one control node; by this way, discontinuities on the boundary after movement are prevented. Figure 2.2 below shows movement of a control node and respective movement of boundary nodes in its’ range. Decaying pattern of boundary node displacement as drifting away from control node as well as the effect of RBF coefficient is represented.



(a) (b)

Figure 2.2. Boundary movement (a) with high RBF coefficient (b) with low RBF coefficient

[1] Reader referred to [7] for the reason of Gaussian RBF selection.

Mesh nodes or inner nodes (the ones which are not on the boundary); however, displaced with using Fast Dynamic Grid Deformation or so called FDGD method. This method is presented by Liu et al [8] and further examined by Kara C. [7]. Here the method is explained shortly, for further information reader referred to [7] and [8].

FDGD method starts with "Delaunay Mapping" in the whole domain which looks like Figure 2.3 and Figure 2.4 below for a unit circle.

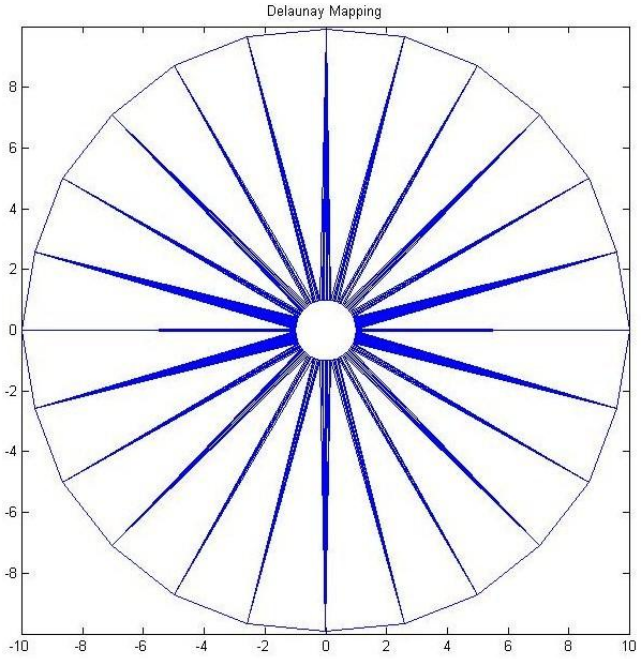


Figure 2.3. Delaunay Mapping

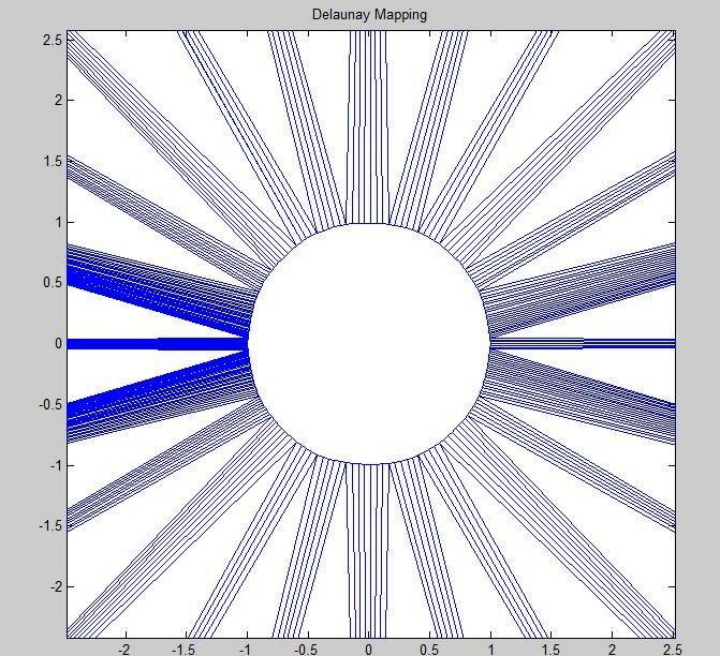


Figure 2.4. Delaunay Mapping (zoom view)

It should be kept in mind that with Delaunay Mapping there were two kinds of triangles on the domain; the coarse ones (Delaunay triangles) and the fine ones (computational mesh triangles). Idea here was moving the course triangles with the movement of mesh triangles i.e fine triangles. This means indirectly, moving the inner nodes with the movement of boundary nodes. To achieve the aim; first fine triangles were flagged according to which coarse triangle they were located in. After that, so called "area ratios" were calculated. For better understanding see Figure 2.5 below. In Figure 2.5; representative course triangle ABC has three fine triangles inside namely, ABP, APC and CPB. After calculating the areas of these four triangles with Equation 2.2 area ratios were calculated with Equation 2.3.

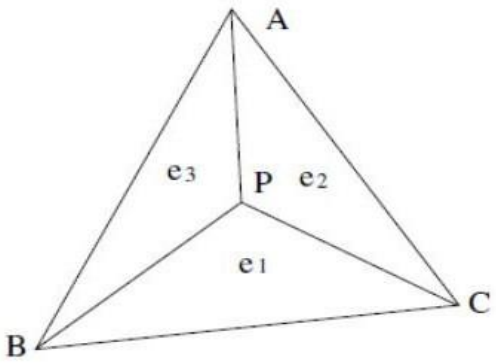


Figure 2.5. Representative triangle

$$S_1 = \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_P & y_P & 1 \end{vmatrix}$$

$$S_2 = \begin{vmatrix} x_A & y_A & 1 \\ x_P & y_P & 1 \\ x_C & y_C & 1 \end{vmatrix}$$

$$S_3 = \begin{vmatrix} x_C & y_C & 1 \\ x_P & y_P & 1 \\ x_B & y_B & 1 \end{vmatrix}$$

$$S = \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix}$$

Equation 2.2

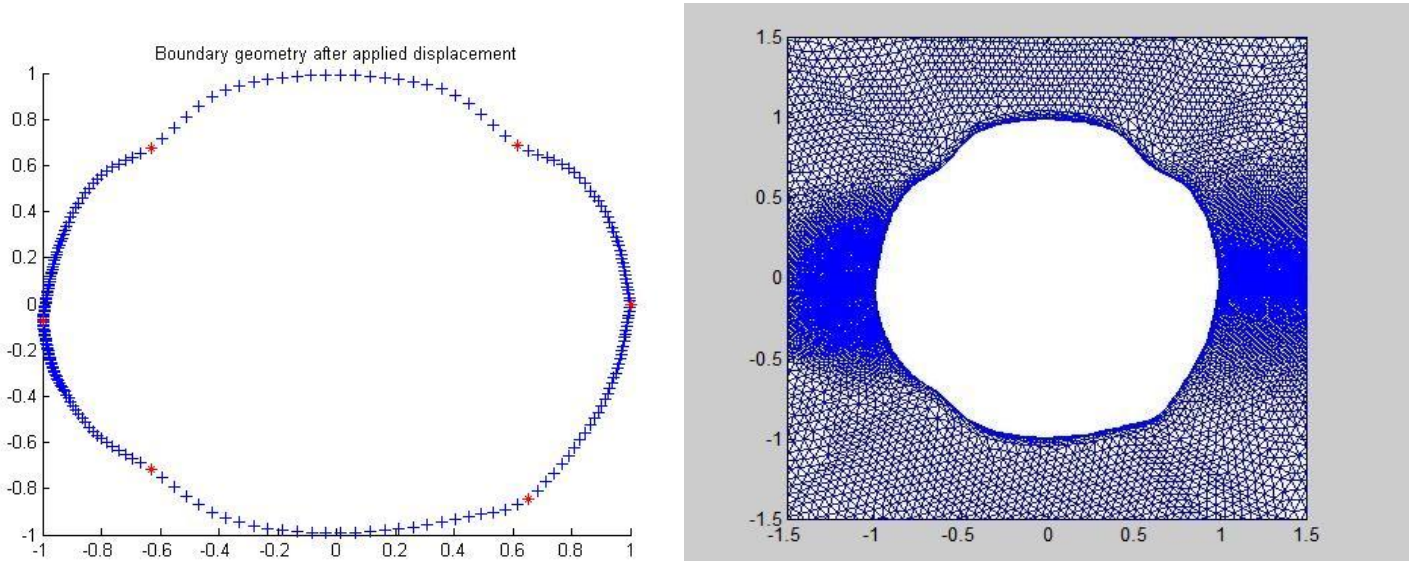
$$e_i = \frac{S_i}{S}$$ **Equation 2.3**

After a boundary movement, boundary node coordinates and accordingly area of fine triangles were obviously changing and the new coordinates for inner nodes were calculated by keeping the area ratios the same. For example on representative triangle ABC, new coordinates for point P (inner node coordinate) is calculated as described in Equation 2.4 and Equation 2.5.

$$x_P = \sum_i e_i x_i$$ **Equation 2.4**

$$y_P = \sum_i e_i y_i$$ **Equation 2.5**

See Figure 2.6. (a) Defromed boundary (b) Deformed mesh below for a finely moved whole mesh with RBF technique for boundary nodes and FDGD technique for inner nodes.



(a) (b)

Figure 2.6. (a) Defromed boundary (b) Deformed mesh

2.1. GEOMETRIC OPTIMIZATION ALGORITHM

To optimize the boundary geometry; in this thesis, "Modified Cuckoo Search" optimization algorithm was used. Modified Cuckoo Search or so called MCS algorithm was derived from the Cuckoo Search algorithm which was first presented by X.-S. Yang and S. Deb. at 2010 [9]. Being a genetic algorithm MCS is more advantageous than gradient based methods besides, shows better performance than other genetic algorithms such as Particle Swarm, Differential Evolution or Cuckoo Search [1].

Like any other genetic optimization algorithm MCS starts with an initial population which aims to optimize a function on a restricted domain by following some rules at each iteration. Initial sampling for the MCS algorithm was obtained by using Latin Hypercube Sampling[2] (LHS) and followed by the methodology explained in *Algorithm 1*.

First of all, reader should remember that, as explained in detail at Section 2.1; geometry movements were done in this thesis work with RBF mesh movement algorithm and reader may also remember the core of the mesh movement algorithm was displacing the control nodes first and then displacing the boundary accordingly. So the basic idea behind optimization algorithm was finding proper displacement for control nodes to reach an optimum geometry.

Initial sampling of the optimization algorithm created in a design space which the boundary of this design space was defined by the maximum allowed displacement of control nodes. Initial samples were scattered on this domain using LHS which each sample was offering a different displacement for control node and offering different boundary geometry accordingly. So as a summary, initial samples or so called nests was created with LHS in the design space defined by maximum allowed displacement of control nodes and each of these nests (geometries) was optimized with following *Algorithm 1* below.

Number of nests that mentioned above can be as much as the user defines, however it should be kept in mind more number of nests requires more computer power and time to solve. Lets now discover the terms used at *Algorithm 1* closely for better understanding of MCS. The

[2] For more information about Latin Hypercube Sampling reader referred to [3]

term "fitness"; which mentioned a lot at *Algorithm 1*, defines the objective function value for each of the initial geometries (nests). Another new concept mentioned at *Algorithm 1* was Lévy Flight[3]. This is a special way of moving nests in the design space to search for better fitness values. At each cuckoo iteration nests were moved to different positions according to their status (being one of the best fitness nests or not) to search for better fitness values. While top nests were not moved far from their current position; discarded nests were moved further from their current positions. Idea here was searching the optimum position in the design space more around the best fitness nests and moving less fitness nests all around design space to explore existing of better spots in the design space. Finally at the *Algorithm 1*; G was defined as current iteration number and A was a constant generally fixed to 1.

Algorithm 1: Modified Cuckoo Search

---

1. Do initial sampling of n number of nests $\boldsymbol{x_i}(i = 1,2 \dots, n)$
2. For all $\boldsymbol{x_i}$ calculate fitness $F_i = f(x_i)$
3. Sort nests by fitness considering best 25% as top nests and remaining 75% as abandoned nests
4. For all abandoned nests current position being $\boldsymbol{x_i}$ ***do***

Calculate a Lévy flight step size $\alpha = A/\sqrt{G}$ and perform a Lévy flight from $\boldsymbol{x_i}$ to create a new nest $\boldsymbol{x_k}$

5. $\boldsymbol{x_i} = \boldsymbol{x_k}$ and $F_i = f(x_k)$ ***end do***

6. For all top nests current position being $\boldsymbol{x_i}$ ***do***

Pick another nest from the top nests randomly, selected random top nest as $\boldsymbol{x_j}$

- If $\boldsymbol{x_i} = \boldsymbol{x_j}$ calculate a Levy flight step size $\alpha = A/G^2$ and perform a Lévy flight from $\boldsymbol{x_i}$ to generate a new nest $\boldsymbol{x_k}$. And set $F_k = f(x_k)$.
- If $\boldsymbol{x_i} \neq \boldsymbol{x_j}$ and $F_i \neq F_j$ perform a Lévy flight from low fitness nest to high fitness nest with step size being $\frac{|\boldsymbol{x_i} - \boldsymbol{x_j}|}{\Phi}$ where $\Phi = \frac{1+\sqrt{5}}{2}$ (golden ratio). Call the new nest created as $\boldsymbol{x_k}$.
- If $\boldsymbol{x_i} \neq \boldsymbol{x_j}$ and $F_i = F_j$ perform a Lévy flight and move to half way. Call the new created nest as $\boldsymbol{x_k}$.
- Choose a random nest among all the nests called $\boldsymbol{x_l}$.

If $F_l > F_k$ $\quad x_k = x_l$
If $F_l < F_k$ $\quad x_k = x_k$

Important point which needs attention about *Algorithm 1* is the fact that the fitness (or the desired function value) for each nest in each iteration has to be calculated. Depending on the

[3] For detailed information about Levy Flight reader referred to [3].

objective function, this can be a costly procedure. To reduce the computational cost; in this thesis work Proper Orthogonal Decomposition (POD) reduction order method was implemented. Moreover, the results obtained with POD and without POD compared and effectiveness of this reduction order implementation tested. Details of POD method presented in the following section.

2.2. REDUCED ORDER MODELLING AS A CONVERGENCE ACCELERATOR[4]

To avoid fitness calculation via CFD solutions at each cuckoo iteration, POD reduction order method was implemented in this thesis work. For any function desired to optimize, here summarized how POD works.

Suppose the function desired to optimize is $z(x,t)$ which can be defined in variables-separated form as follows:

$$z(x,t) \approx \sum_{K=1}^{M} a_k(t)\Phi_k(x) \qquad \textbf{Equation 2.6}$$

The representation of Equation 2.6 is not unique; for example the choice of $\Phi_k(x)$ can be made as Fourier series, Legendre polynomials or Chebyshev polynomials and so on. Any choice of $\Phi_k(x)$ creates a basis for the objective function. POD aims to choose the most suitable basis functions i.e set of $\Phi_k(x)$ for any specific objective function.
If we have chosen orthonormal basis functions such as:

$$\int_x \Phi_{k1}(x)\,\Phi_{k2}(x)\,dx = \begin{cases} 1 \; if \; k1 = k2 \\ 0 \; otherwise \end{cases} \qquad \textbf{Equation 2.7}$$

Then

$$a_k(t) = \int_x z(x,t)\,\Phi_k(x)\,dx \qquad \textbf{Equation 2.8}$$

Equation 2.8 means for orthonormal basis functions, the determination of the coefficient function $a_k(t)$ depends only on $\Phi_k(x)$ but not the other Φ's.
The real question is how should be the choice of $\Phi_k(x)$. Orthonormality is useful. Other than that; for any desired accuracy (any value of M in Equation 2.6) $\Phi_k(x)$ can be chosen in such a

[4] For this section [10] was taken as a reference.

way that the approximation as good as possible in Least Squares sense[5]. That means the first two of function $\Phi_k(x)$ gives the best two term approximation; in a similar manner lets say the first seven gives the best possible seven term approximation of objective function and so on. These special, ordered, orthonormal functions are called the proper orthogonal modes for the objective function $z(x,t)$; with these functions, the expression in Equation 2.6 is called the POD of $z(x,t)$.

2.2.1. FINITE DIMENSIONAL CASE

POD was applied on this paper to decompose pressure field which was the only necessary field to achieve first -distortion calculation See Equation 1.1- and second -lift/drag ratio calculation See Equation 1.2 and 1.3- objectives of this thesis work.

Since finite dimensional pressure field was the concern; this section included to explain POD application on finite dimensional case. Here is how it works:

The objective field first has to be defined in a matrix form. In our case, pressure field defined in such a way that each column includes the data of each snapshot (different geometry) and each row includes the data of all the nodes in the geometry. In other words, column one corresponds to snapshot one with each row element being the nodal pressure value on that snapshot geometry. In this case number of columns (say m) was highly bigger than number of rows (say N) See Equation 2.9 in our case.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{Nm} \end{bmatrix} \qquad \textbf{Equation 2.9}$$

After defining the objective function in matrix from next thing to do is computing the singular value decomposition (SVD) of this matrix A which is of the form:

$$A = U\Sigma V^T \qquad \textbf{Equation 2.10}$$

Where U is an NxN orthogonal matrix, V is an mxm orthogonal matrix with T indicating matrix transpose, and Σ is an Nxm matrix with all elements zero except along the diagonal.

[5] For details about Least Squares [10]

The diagonal elements of Σ called the singular values of A and also A^T and are unique. In Equation 2.10, let $U\Sigma = Q$. Then the matrix Q is Nxm, and $A = QV^T$. Letting q_k being the k[th] column of Q and v_k being the k[th] column of V; matrix A can be written as in Equation 2.11

$$A = QV^T = \sum_{k=1}^{m} q_k v_k^T \qquad \textbf{Equation 2.11}$$

Equation 2.11 is the discrete form of Equation 2.6. Matrix A represents function $z(x,t)$; column matrix q_k represents function $a_k(t)$ and row matrix v_k^T represents$\Phi_k(x)$. So the new aim is to calculate matrix V to obtain proper orthogonal modes of matrix A. Matrix V can be obtained simply by using commercial software like MATLAB however if $m \gg N$, it is more efficient to first compute the matrix U as the matrix of eigenvectors of AA^T. This method is called the "method of snapshots" in the POD literature.

2.2.2. RECONSTRUCTION

All the procedure explained on Section 2.3 till this point was to avoid fitness calculation using CFD solutions at each cuckoo iteration. To achieve this, POD decomposition implemented on the pressure field which obtained from CFD solver while constructing the initial snapshots (Figure 1.5). Once the pressure field obtained (first cuckoo iteration), at subsequent iteration, a new pressure field had to be 'guessed' for new fitness calculation. In this section; how to reconstruct a new pressure field (or any field interested) after decomposing it was explained.

With keeping the Equation 2.11 the known pressure field could be written as follows:

$$p(x) = \sum_{k=1}^{m} q_k(x) v_k^T \qquad \textbf{Equation 2.12}$$

With x representing the known spatial coordinate and $p(x)$ representing the known pressure value at known spatial coordinates. When moved to subsequent cuckoo iteration; spatial coordinates had changed and new pressure value needed to be 'guessed'. The unknown new pressure field could be defined as:

$$p(\hat{x}) = \sum_{k=1}^{m} q_k(\hat{x}) v_k^T \qquad \textbf{Equation 2.13}$$

Since in Equation 2.13 the term v_k^T or in other words the proper orthogonal modes were already known; to 'guess' new pressure field only $q_k(\hat{x})$ term or so called coefficients term has to be calculated. In this thesis work coefficients term was approximated using Radial Basis Function interpolation. According to Buhmann[11] the general form of RBF interpolation can be written as in Equation 2.14

$$f(x) = \sum_{i=1}^{N} \lambda_i \zeta(\|x - x_i\|) \qquad \textbf{Equation 2.14}$$

Where λ_i called as interpolation coefficients and $\zeta(r)$ is any radial basis function desired. So finally from known $q_k(x)$ coefficients term λ_i interpolation coefficients calculated solving $\boldsymbol{F=B\Lambda}$ system for $\boldsymbol{\Lambda}$ See Equation 2.15 and then with using those interpolation coefficients obtained unknown $q_k(\hat{x})$ finally calculated See Equation 2.16.

$$q_k(x) = \sum_{i=1}^{N} \lambda_i \zeta(\|x - x_i\|) \qquad \textbf{Equation 2.15}$$

i.e $\boldsymbol{F=B\Lambda}$

Where $\boldsymbol{F} = q_k(x)$, $\boldsymbol{B} = \zeta(\|x - x_i\|)$ and $\boldsymbol{\Lambda} = \lambda_i$

$$q_k(\hat{x}) = \sum_{i=1}^{N} \lambda_i \zeta(\|\hat{x} - x_i\|) \qquad \textbf{Equation 2.16}$$

The reconstruction of pressure field (See Equation 2.13) using proper orthogonal modes v_k^T obtained from POD decomposition together with coefficients term $q_k(\hat{x})$ obtained using RBF interpolation concludes Chapter 2. The coming chapter devoted to implementation of the methodologies explained in this chapter; as well as the results obtained after those implementations.

CHAPTER 3: IMPLEMENTATION, RESULTS AND ANALYSIS

3. IMPLEMENTATION

This chapter devoted to implementation of methodologies explained in Chapter 2 to the problems of concern.

3.1.MCS WITH/OUT POD ON BLOODHOUND SSC AIR INTAKE DUCT

3.1.1. MESH MOVEMENT

Reader may remember the optimization was initiated with mesh movement algorithm. At the end of mesh movement, initial snapshots or nests were aimed to be obtained; and after that finally, initial nests were sent to CFD solver for initial solutions. This section devoted to better understanding of how mesh movement algorithm works specifically for Bloodhound SSC air intake duct geometry. Performance analysis of mesh movement algorithm was presented as well as the initial snapshots obtained.

To understand the mesh movement algorithm implementation on Bloodhound SSC air intake duct; 2D duct boundary geometry is created via MATLAB; specifying the locations of 7 control nodes on it with huge far field definition to catch free stream behaviour as well. See Figure 3.1.

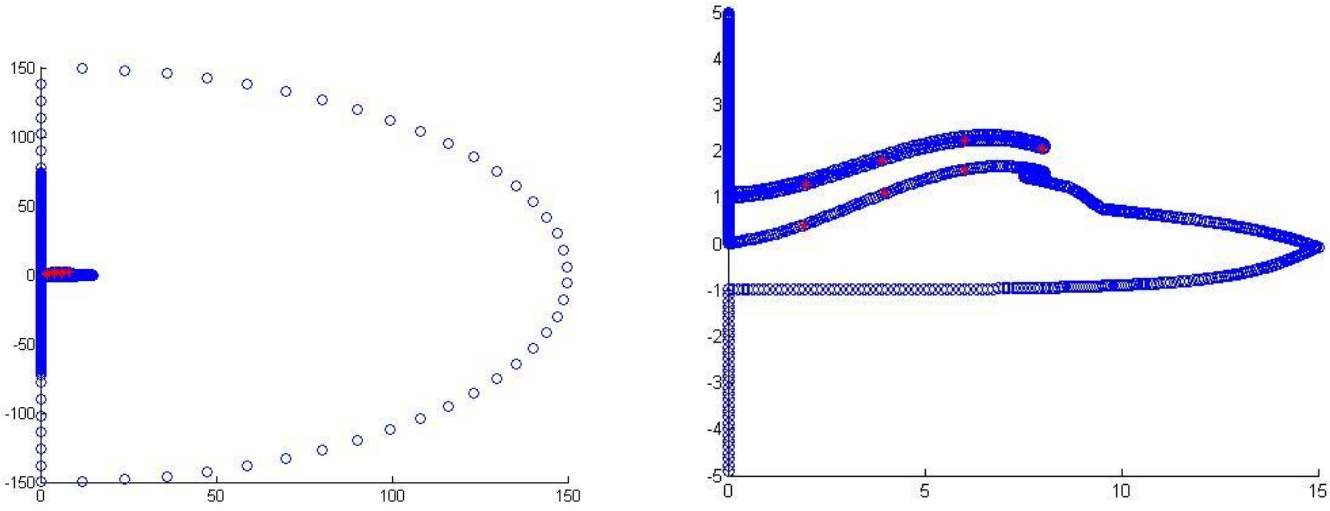


Figure 3.1. Bloodhound SSC far field and boundary geometry

Viscous computational mesh with viscous boundary layer was created with a CAD tool and mesh information passed to MATLAB. Computational mesh includes 20973 nodes in total, 260 of them being on the boundary and defined as viscous. The final form of 2D computational mesh of Bloodhound SSC is shown in Figure 3.2.

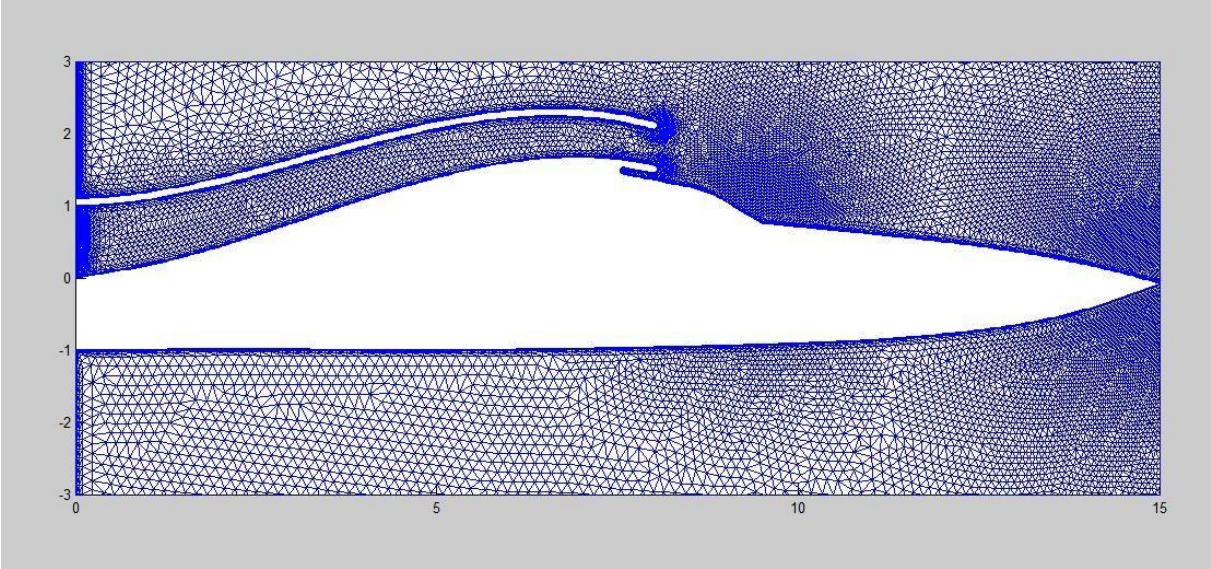


Figure 3.2. Bloodhound SSC computational mesh

If the whole code is thought in two parts with first part being the mesh movement algorithm and second part being optimization (MCS) algorithm; after the creation of boundary and mesh; RBF mesh movement algorithm or so called first part of the code started to apply. Other than the location of control nodes; Mach number, maximum displacement of control nodes, number of nests and number of solver iterations (iteration number that solver is going to use at the end of mesh movement to solve initial snapshots) had to be defined by the user to run the first part. By choosing Mach number as 0.8, maximum x and y displacements as 0.1 and 0.3 respectively and number of nests as 30; following initial snapshots (nests) were obtained. Out of 30 nests, only 6 nests were shown for representative purposes See Figure 3.3. It should be mentioned here that MCS is a totally heuristic algorithm. This is why, mesh movement part of the algorithm was seeded randomly in the very beginning; means anytime when first part of the code (mesh movement part) was run, different initial geometries were obtained. Running the same code to obtain initial snapshots again with the same input values, user is going to end up with different snapshots than the ones shown in Figure 3.3.

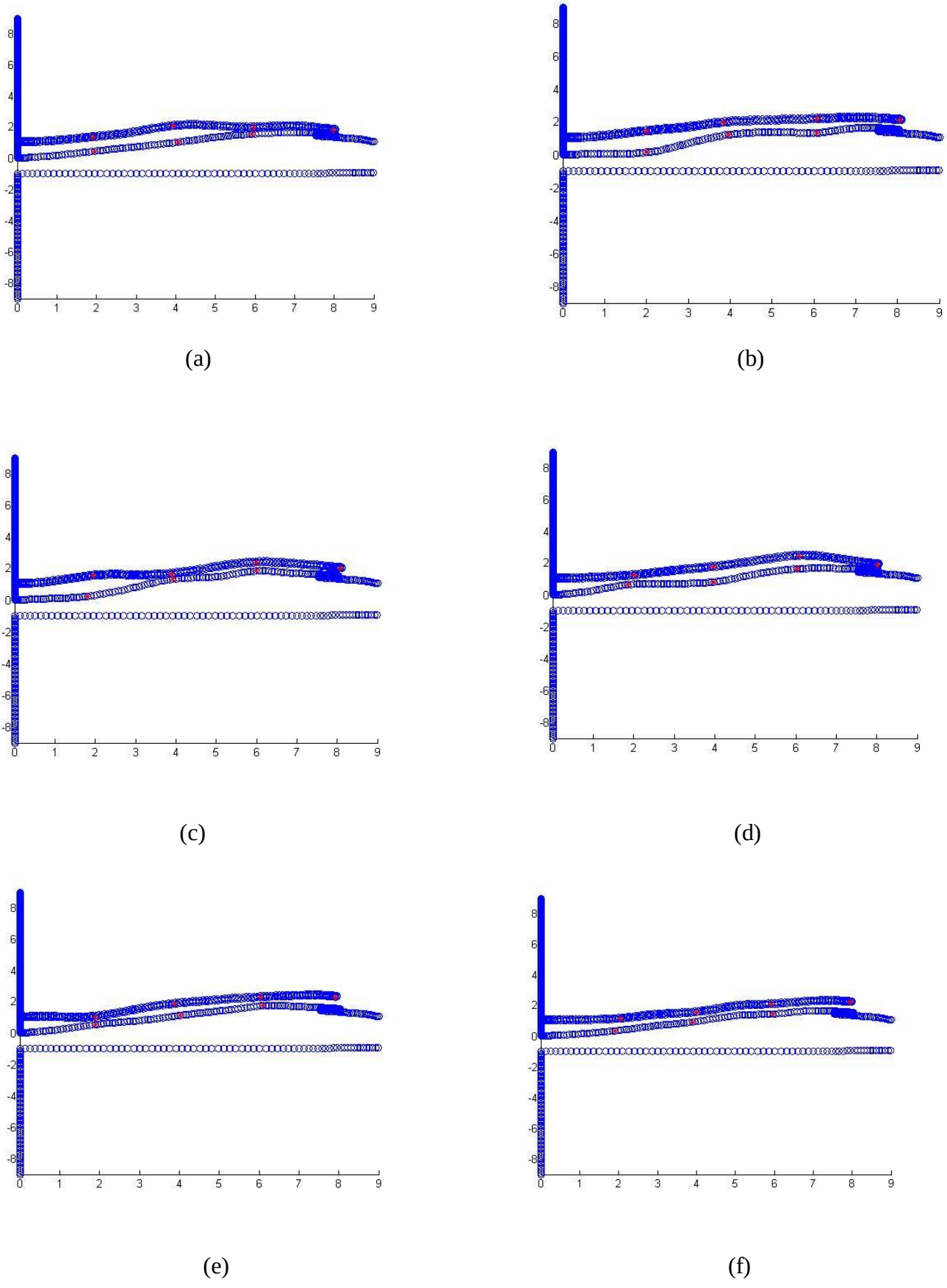
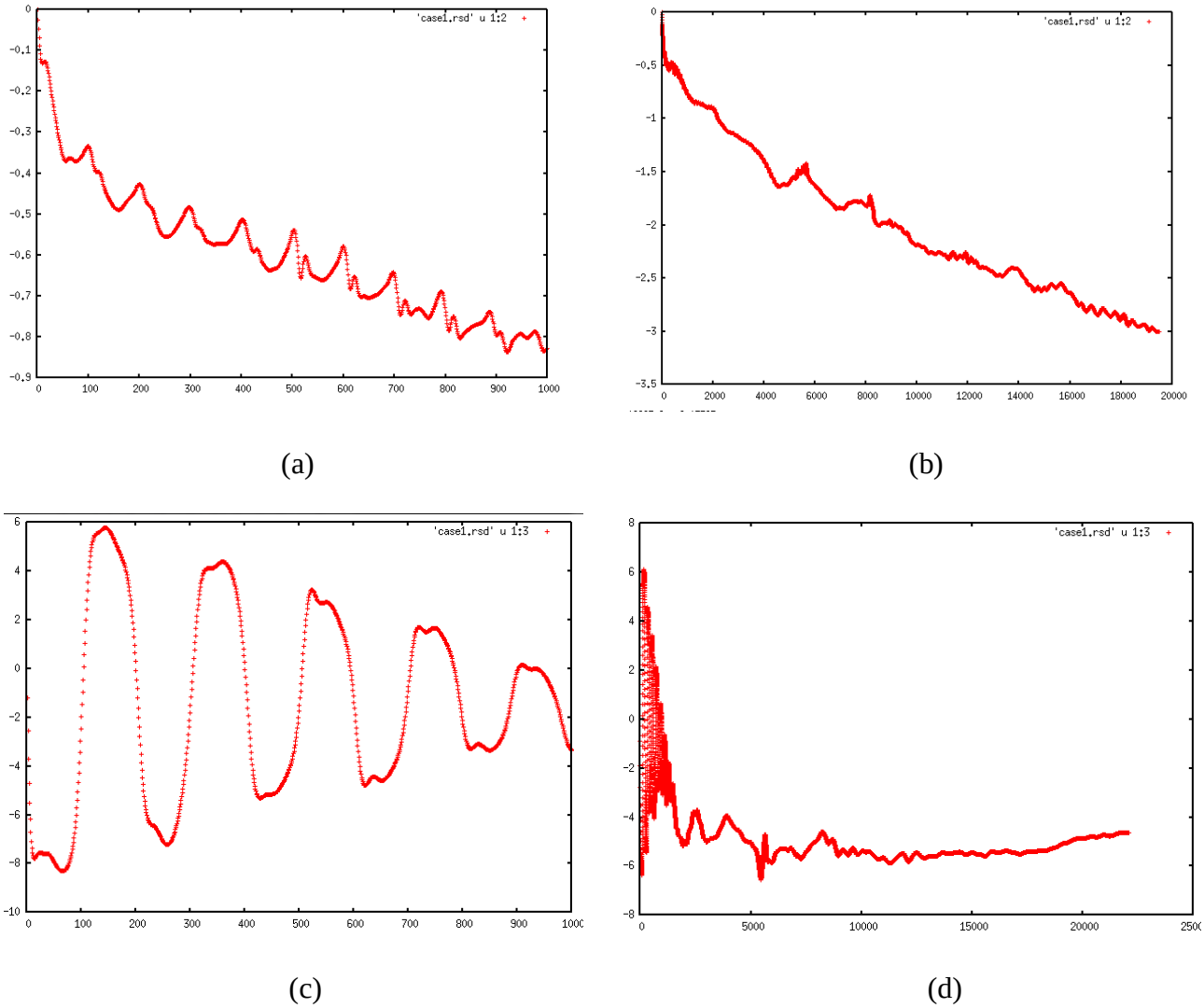


Figure 3.3. Initial nests (a) first (b) second (c)third (d)fourth (e)fifth (f)sixth

After obtaining initial nests like the ones shown in Figure 3.3, next step was sending those nests to a CFD solver, for obtaining initial solutions. Whenever preceded to optimization part; using POD, fitness values for new nests were going to be estimated using modes and coefficients acquired from these initial solutions. Before sending the initial nests to CFD solver; number of maximum iterations that CFD solver is going to use, was an important input variable to decide. To understand its effect, and decide on a reasonable value; reader now invited to examine the behaviour of CFD solver residual with respect to number of solver iterations. Figure 3.4 below shows the trend of residual and lift for different number of solver iterations.



(a) (b)

(c) (d)

Figure 3.4. CFD residual Vs number of iterations (a)residual with 100 iterations (b)residual with 20000 iterations (c) lift with 1000 iterations (d) lift with 20000 iterations

It could be seen, as the number of solver iterations increases residual (the diffrence between two iterations calculated by solver) decreases more and more without any oscillations, as well as the lift calculated by CFD solver stabilizes around 20000 iterations (or -3 in the y axis of graphs shown on Figure 3.4); suggests, as a stopping criteria for number of solver iterations, user can safely choose -3 (ratio between current and initial residual) or 20,000 which is surely a good approximation for solution. After choosing the stopping criteria for CFD solver and obtaining initial nests; it was time to proceed to second part of the code which MCS with/out POD implemented on Bloodhound SSC air intake duct geometry.

3.1.2. OPTIMIZATION

The optimization algorithm MCS was presented in detail in Section 2.2; however, what reader is going to find in this section is the implementation of MCS on Bloodhound SSC air intake duct, as well as the performance analysis of the method with respect to changing design variables. From the first part (mesh movement part) of the code initial nests were already obtained and the stopping criteria for the CFD solver already decided. Optimization part, or so called second part of the algorithm starts with solving initial nests using CFD solver. At the end of CFD solution, output files with '.resp' extension, including nodal pressure values for each nest was obtained. A Swansea University undergraduate student implemented the MATLAB algorithm to obtain distortion at the engine inlet from nodal pressure values obtained after CFD solution .

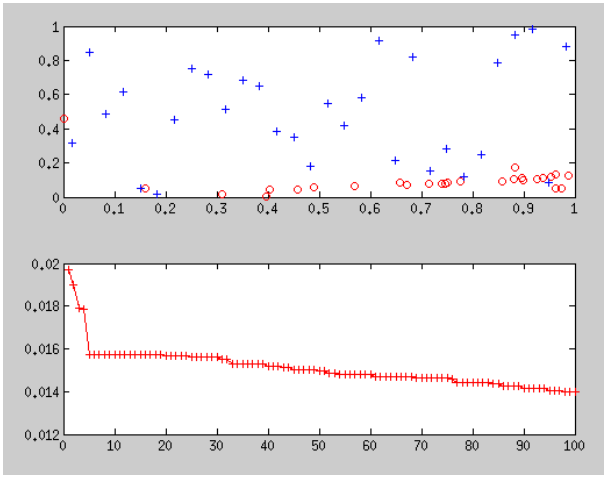
Important thing here to mention is that the initial nests were solved in parallel with using as much number of processors as the number of nests (running each nest in each processor) to save enormous time. Table 3-1 below shows time required to solve initial nests in the CFD solver to give an idea to the reader about time sense.

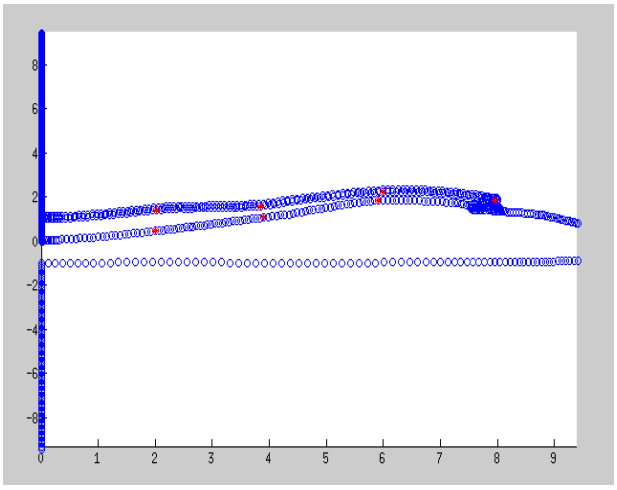
Table 3-1. Wall time with increasing CFD solver iterations

Number of Control Nodes	Mach Number	Number of Nests	Number of CFD Iterations	Wall Time
7	0,8	30	100	~5 min.
7	0,8	30	1000	~20 min
7	0,8	30	20000	~2 hours

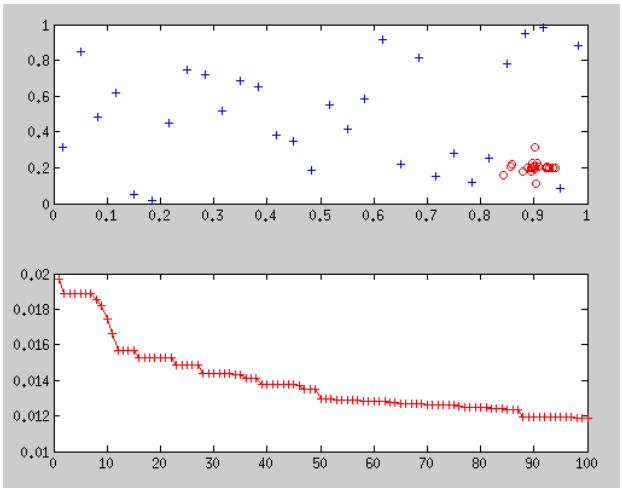
After setting stopping criteria for CFD solver to -3 and getting initial solutions; MCS algorithm explained in detail in Section 2.2, implemented. Being Ma 0.8, number of nests 30, number of control nodes 7, maximum allowed displacements in x and y directions 0.1 and 0.3

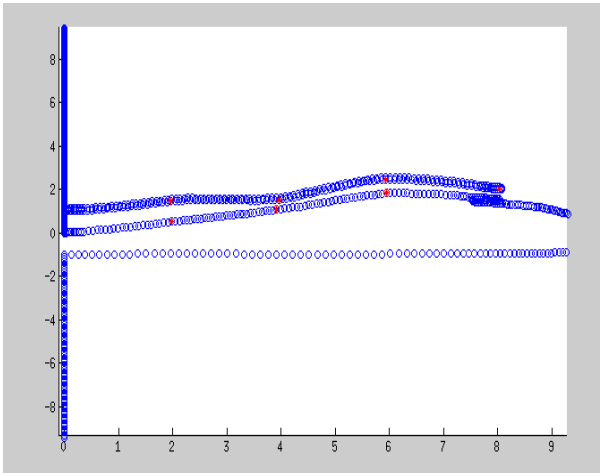
respectively, and number of cuckoo iterations 100, following results (See Figure 3.5) were obtained. Reader should understand with 7 control nodes and with two degrees of freedom (x and y displacements) allowed for each, a 14 dimensional space was analysed. In the following Figure 3.5 ; the top graph shows the initial positions of nests, in the design space normalized between zero and one, with blue +'s, and the final positions of the nests after MCS algorithm implemented, with red o's. All the graphs were drawn represents only 2 of the dimensions out of 14 for representative purposes; and those two dimensions presented were the ones belong to control node number one (the leftmost down control node in Figure 3.1) however, nests shows more or less similar behaviour also in other control nodes. In Figure 3.5 below, the bottom graph, on the other hand; shows the convergence history of the best nest through 100 cuckoo iterations. Since MCS is a totally heuristic algorithm, second part of the code run for five consecutive times to get a better understanding of the algorithm over averaged values.



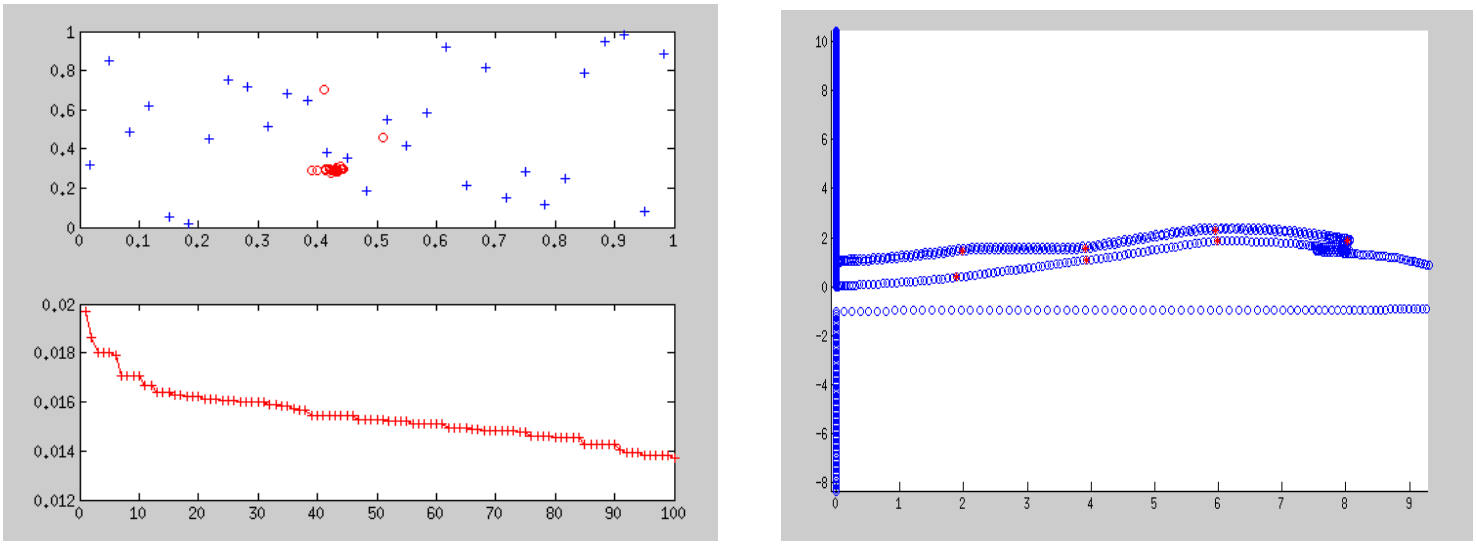


(a)

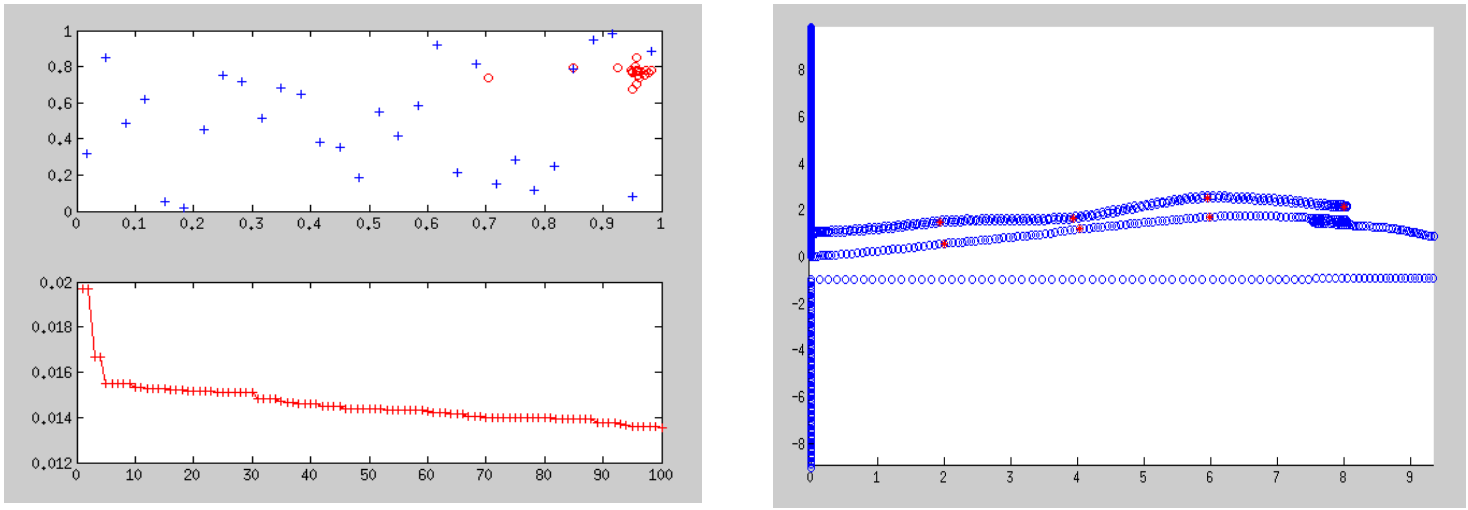




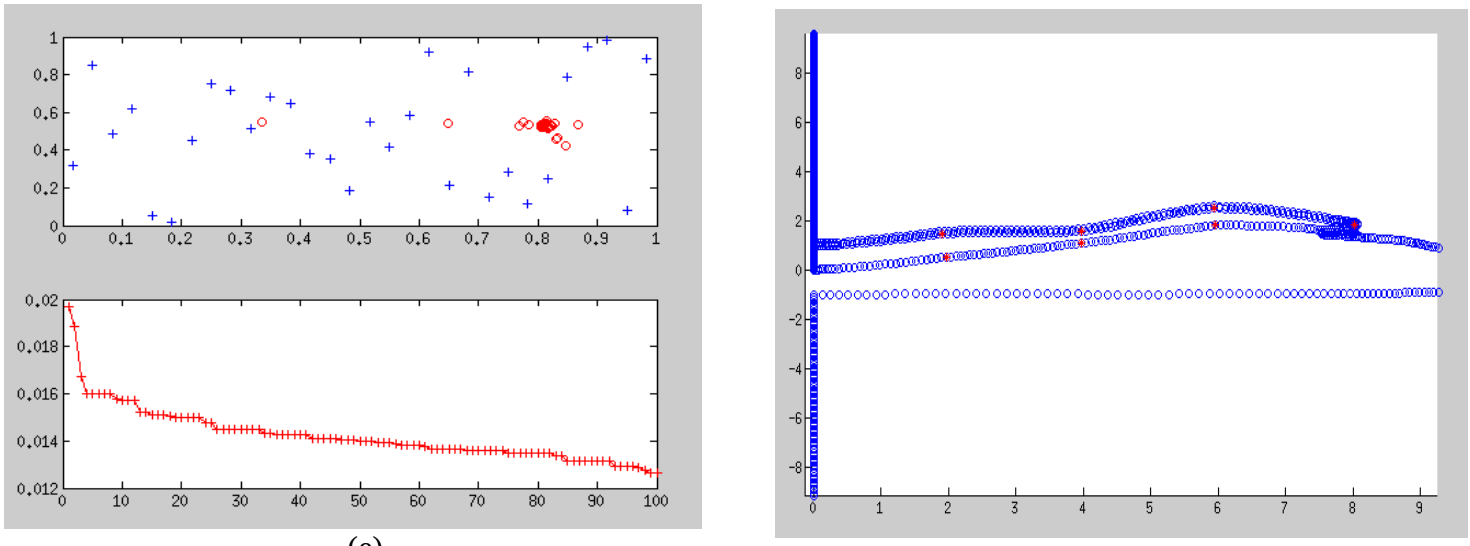
(b)



(c)



(d)



(e)

Figure 3.5. MCS algorithm, on the left nest positions and convergence history, on the right optimum geometry (a) first run (b) second run (c) third run (d) fourth run (e) fifth run

Results presented in Figure 3.5 were impressive with being quite different from each other at the end of each run. When running a completely heuristic algorithm some kind of discrepancy between consecutive runs is acceptable but looking at above results one can easily sees this is more than little discrepancy. Above results makes its mandatory to validate results obtained with MCS and think on the probable problem.

Reader may remember the only 'guess' had done in MCS with POD algorithm was 'guessing' the new fitness values for new geometries using POD. In each of the runs POD decomposition worked in the same way however when it comes to POD reconstruction RBF interpolation used and this brings the difference between the runs. Here it should be clarified that RBF interpolation used in this thesis work both for mesh movement and POD reconstruction; the one which was mentioned here was the RBF used in POD reconstruction. It had mentioned in Section 2.3.2, different RBF types were available and the general form of RBF's was shown.

RBF type used in POD reconstruction for obtaining above results was Gaussian RBF which is defined as in Equation 3.1 where d is the mean distance between the points in the new nests (known) used for interpolation and c is a coefficient should be varied.

$$\Phi(r) = {}^{1}/_{c\,d^2} * e^{-{r^2}/{c\,d^2}} \qquad \textbf{Equation 3.1}$$

Results obtained MCS with POD using RBF type defined in Equation 3.1 were validated in coming section 3.1.3, however, before that, reader now invited to explore performance of MCS with POD algorithm with respect to changing design parameters.

Table 3-2 below shows the effect of number of cuckoo iterations, of the same problem analysed above, on the optimized solution.

Table 3-2. Effect of number of cuckoo iterations on solution

				Distortion Value on Different Runs					
Number of Control Nodes	Mach	Number of Nests	Number of Cuckoo Iterations	run1	run2	run3	run4	run5	Average Distortion
7	0,8	30	20	0.015	0.015	0.015	0.0162	0.0146	0.0153
7	0,8	30	50	0.0137	0.0148	0.0142	0.014	0.0152	0.0144
7	0,8	30	100	0.0128	0.0122	0.0131	0.0132	0.0132	0.0129

Clearly seen from Table 3-2, increasing number of cuckoo iterations leads to lower average distortion values at the engine inlet; which means better results. To give a time sense to the reader Table 3-3 below shows the wall time to complete different number of cuckoo iterations.

Table 3-3. Wall time to complete different number of cuckoo iterations

Number of Control Nodes	Mach	Number of Nests	Number of Cuckoo Iterations	Wall Time
7	0,8	30	20	~1 min.
7	0,8	30	50	~2 mins.
7	0,8	30	100	~5 mins.

Figure 3.6 below is again important to understand the effect of number of cuckoo iterations on optimized solution. In Figure 3.6, positions of top 7 nests (25% of total nests; 0.25*30=7.5) were shown in the problem domain with increasing number of cuckoo iterations. By looking at Figure 3.6, it can be seen clearly how nests agree on one specific solution point and gathered together with increasing number of cuckoo iterations.

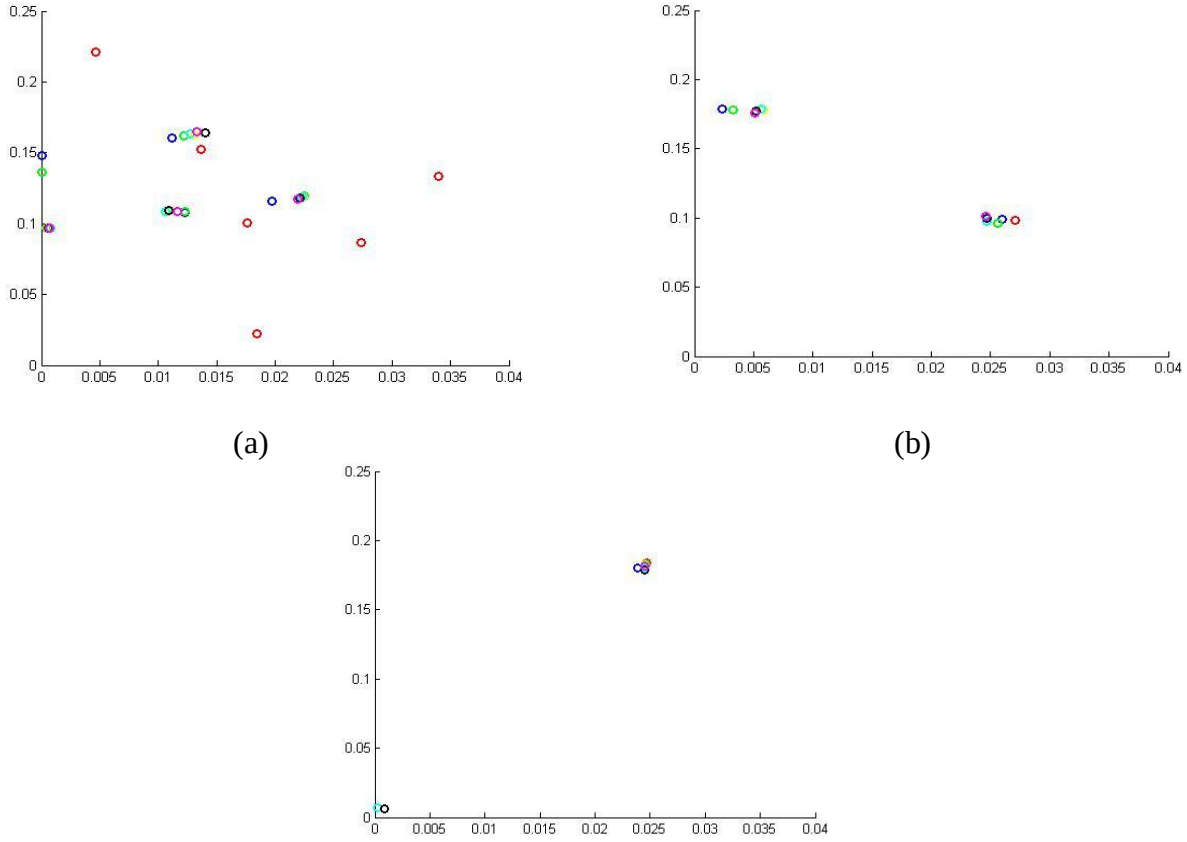


Figure 3.6. Position of top nests with increasing number of cuckoo iterations (a) 20 iterations (b) 50 iterations (c) 100 iterations

To understand the effect of number of nests on the optimized solution Table 3-4 below should be examined together with Table 3-2. The problem definition is still the same, with only changing number of nests from 30 to 20.

Table 3-4. Effect of number of nests on optimized solution

				Distortion Value on Different Runs					
Number of Control Nodes	Mach	Number of Nests	Number of Cuckoo Iterations	run1	run2	run3	run4	run5	Average Distortion
7	0,8	20	20	0.021	0.0218	0.0218	0.0218	0.0219	0.0217
7	0,8	20	50	0.0209	0.0217	0.0195	0.0198	0.0206	0.0205
7	0,8	20	100	0.0207	0.0179	0.0208	0.0209	0.0177	0.0196

Examining Table 3-4 with Table 3-2, it can be understood decreasing number of nests leads to higher distortion values at the engine inlet which means worst solutions. Increasing number of nests is definitely costly yet the results show it is worthed.

Ultimately Figure 3.7 below includes some important information about MCS algorithm. In Figure 3.7 distance between top nests, for different number of total nests, was shown with respect to increasing number of cuckoo iterations.

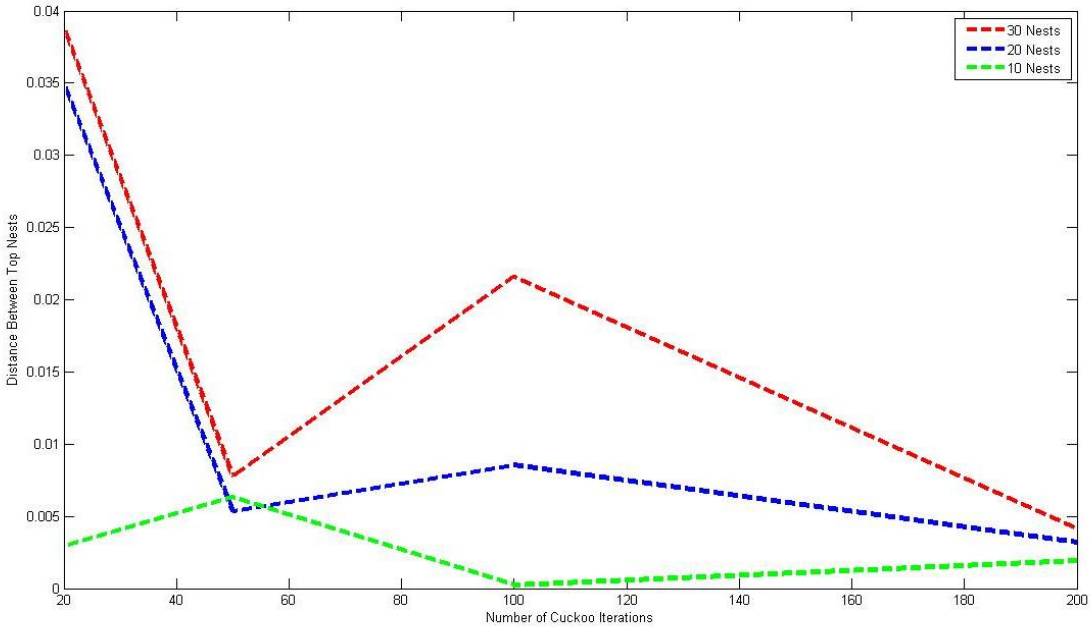


Figure 3.7. Distance between top nests with increasing number of cuckoo iterations

Before the iterations start, top 2 nests of total 10 nests case were closer to each other than top nests of total 20 and 30 nests cases. This was an expected result since number of top nests increase with increasing number of total nests, and having more top nests increase the probability of divergence. With increasing number of cuckoo iterations top nests were not always getting closer to each other, however; when number of iterations reaches to 200, does not matter the total number of nests, all the top nests agree on solution point and get closer. Top nests of total 30 nests case were always more diverted from each other than total 20 nests case and in the same way; top nests of total 20 nests case were always more diverted than total 10 nests case. As a conclusion for this discussion it can be said that does not matter with how many nests you have, as number of cuckoo iterations increase, they always gets

closer to optimum solution. The quality of the solution obtained, however; depends on the number of total nests, with many other parameters as well.

3.1.3. VALIDATION

In Section 3.1.2 MCS with POD algorithm applied on Bloodhound SSC air intake duct geometry with 7 control nodes, Ma being equal to 0.8, stopping criteria for CFD solver being -3, number of cuckoo iterations being 100, maximum allowed displacements being 0.1 and 0.3 in x and y directions respectively and number of nests being 30. In Figure 3.5 results were presented and possible source of the problem detected as RBF selection on POD reconstruction. In this section, reader is going to find the validation of the results obtained using MCS with POD by comparing results with the ones obtained without POD.

Problem definition for validation studies has changed a bit to get healthier results. Number of control nodes reduced to 1 and maximum 0.3 displacement allowed only in y direction. By this way, problem dimension reduced to 1 rather than 14. To understand this 1 dimensional space, 20 nests were used, -3 was chosen as a stopping criteria for the solver and number of cuckoo iterations was decided as 100. New problem was solved in three different fluid regimes by setting the Ma number to 0.5, 0.8 and 1.3 respectively. See Figure 3.8 for the location of control node on Bloodhound SSC geometry.

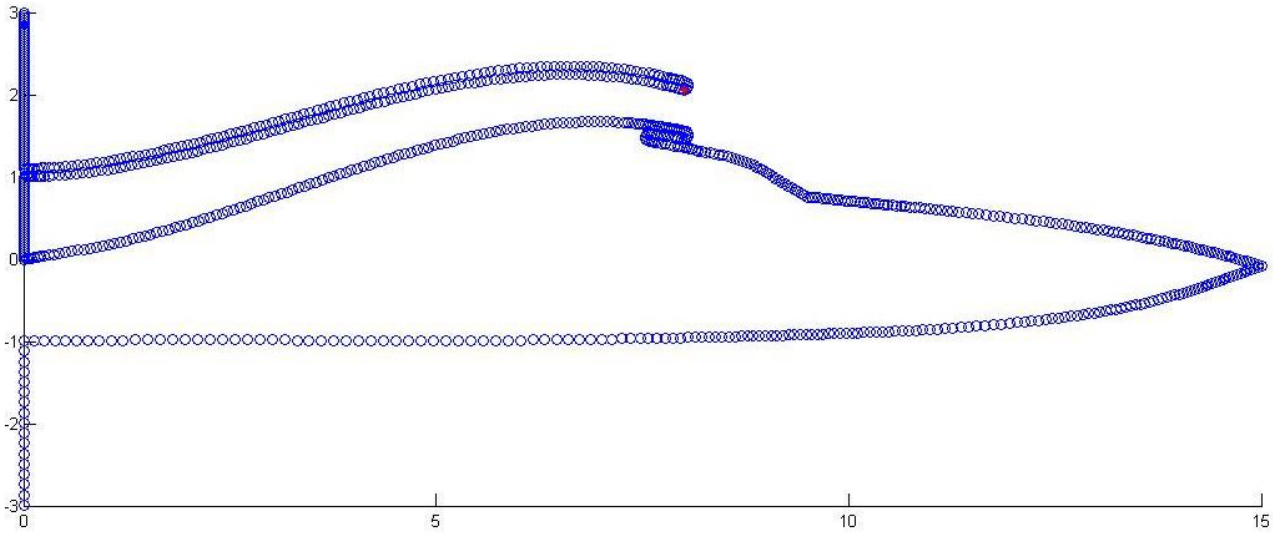


Figure 3.8. Bloodhound SSC

Moreover and most importantly, different RBF types used for POD reconstruction and the behaviour of the algorithm with different RBF types analysed. To decide on the best RBF type, problem was solved without POD as well. Lets now examine what has been done.

Other than Gaussian RBF or so called RBF type 1 (Equation 3.1), two more RBF's introduced to test on POD reconstruction. Equation 3.2 and Equation 3.3 defines RBF type 2 and RBF type 3, respectively.

$$\Phi\,(r) = |r| \qquad \textbf{Equation 3.2}$$

$$\Phi\,(r) = \sqrt{r^2 + cd^2} \qquad \textbf{Equation 3.3}$$

where d is the mean distance between the points in the new nests (known) used for interpolation and c is a coefficient should be varied. Figure 3.9 is graphical representation for different RBF types.

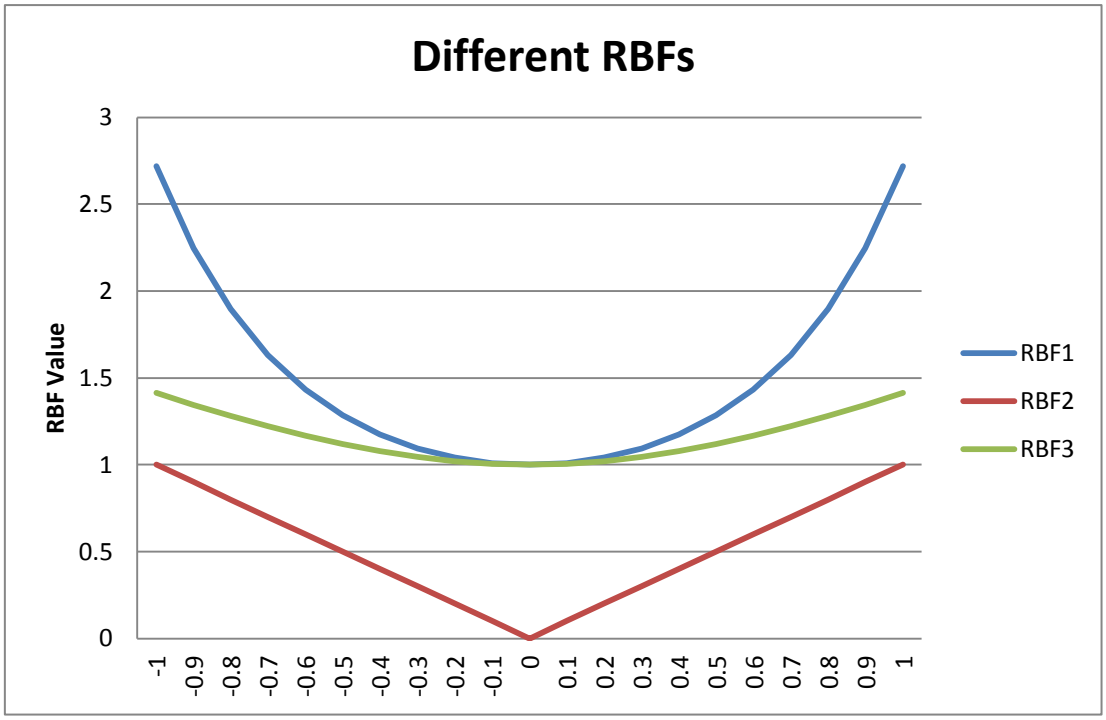


Figure 3.9. Different RBF types

Other than introducing new RBF types, it was an important step to apply MCS without POD. MCS without POD surely gives healthy results since no estimation was done. At each cuckoo iteration, new nests were solved by CFD solver and distortion calculation at the engine inlet was done with 'real' pressure values obtained from solver. Without POD each cuckoo iteration took considerable amount of time; this is why for without POD case, number of cuckoo iterations set to 10. Table 3-5 below shows the time requirements to complete MCS with/out POD.

Table 3-5. Time requirements MCS with/out POD

	Number of Control Nodes	Number of Nests	Number of Cuckoo Iterations	Wall Time (solving initial nests)	Wall Time (optimization)	Total Wall Time
With POD	1	20	100	~45 min.	~ 10 min/100 iter.	~55 min
Without POD	1	20	10	~45 min.	~45min/1 iter.	~495 min=8.25 h.

Table 3-5 clearly presents the enormous time saving that using POD suggests so using POD is a practical choice however now lets use POD in a clever way by improving our RBF selection for POD reconstruction.

Three different RBF types for three different fluid regimes were tested using MCS with POD algorithm as in the problem definition above. For each fluid regime five consecutive runs were made and results were averaged at the end. Table 3-6 below shows the ultimate results from MCS with POD with three different RBF's and MCS without POD. If the length of the car accepted as 15 "units", the y displacments shown in Table 3-6 are also in "units" or they are basically unitless.

Table 3-6. Results MCS with/out POD

Ma	y Displacement MCS With RBF1	y Displacement MCS With RBF2	y Displacement MCS With RBF3	y Displacement MCS Without POD
0.5	0.2921	0.1650	0.1491	0.1616
0.8	-0.4450	-0.1650	-0.1661	-0.1638
1.3	0.1721	0.2682	0.3000	0.2435

Figure 3.10 is the graphical representation of results shown at Table 3-6.

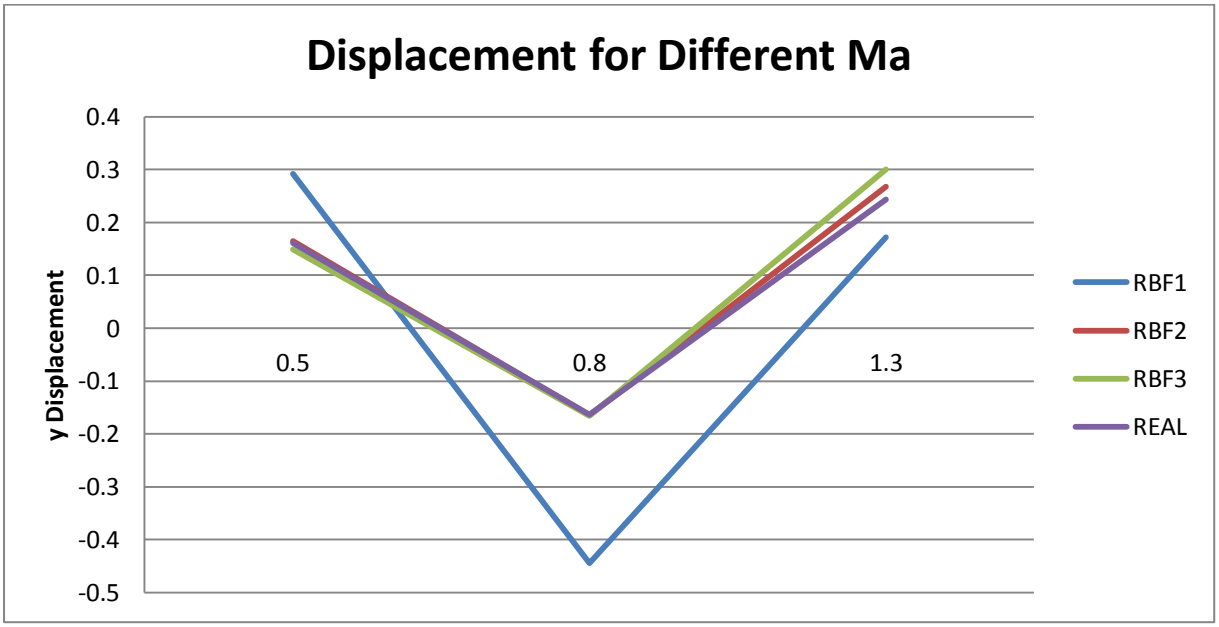


Figure 3.10. Results MCS with/out POD

As a physical fact, it was known with increasing Ma number, gap at the duct inlet should be lesser. With any of RBF's moving from Ma 0.5 to Ma 0.8 leads negative displacements from positive displacements to control node; means closing the gap at the duct inlet. However moving to supersonic regions with Ma 1.3, results have changed to positive displacement values as well. This shows does not matter MCS algorithm applied with POD or without

POD, its results are not trustable in supersonic regions. This may be because, on supersonic regime the particle physics change in such a way that ramjets do not produce net thrust and algorithm was not clever enough to catch this phenomenon.On the other hand, in subsonic and supersonic regions physical facts overlap with analysis results so MCS can safely be applied. By looking at Figure 3.10 another important deduction can be made. RBF type 1, tailing off rapidly (See Figure 3.9) so being different than RBF type 2 and RBF type 3; leads to quite different results with MCS algorithm. RBF type 2 and RBF type 3 on the other hand, being similar to each other; leads really close results with MCS algorithm in all fluid regions. Furthermore RBF type 2 and RBF type 3 results are fairly close to 'real' or MCS without POD results which means rather than working with RBF type 1, working with RBF type 2 or RBF type 3 is the clever thing (See Figure 3.10).

As a conclusion of this section it can be said MCS with or without POD algorithm is trustable at subsonic and transonic regimes but not at supersonic regimes. Moreover it was shown how RBF selection affects the results and understood RBF type 2 and RBF type 3 were better choices than RBF type 1 for MCS with POD algorithm.

3.2.MCS WITH/OUT POD ON ARBITRARY GEOMETRY

Validating MCS with/out POD algorithm on subsonic and transonic regions brought the passion to extend the algorithm to a general purpose geometric optimization algorithm that can work with any arbitrary geometry and optimize it with respect to any design parameter. In this section, reader is going to find MCS with/out POD algorithm applied on arbitrary geometry large deformation problem as well as the analysis of the results obtained.

3.2.1. MESH MOVEMENT

Arbitrary geometry chosen for studies presented in this section was unit circle geometry as mentioned in Chapter 1 before. Test case problem to implement MCS with/out POD can be defined as; unit circle with 6 control nodes located on it (See Figure 3.11) with Mach number being 0.8, and in a viscous flow with zero angle of attack. All control nodes allowed to displace only in y direction so the problem was set as 6 dimensional. Number of nests set to 20 and maximum number of iterations for CFD solver decided as 10,000. The objective function desired to optimize was lift drag ratio on the boundary; and the final optimized geometry supposed to be an aerofoil-like geometry or in other words unit circle has to be 'squeezed' to increase the lift drag ratio on the boundary. With known (after mesh movement

and CFD solution) or guessed (with POD decomposition or reconstruction) pressure values lift and drag was calculated at each node as indicated in Equation 1.2 and Equation 1.3 and integrated over the boundary.

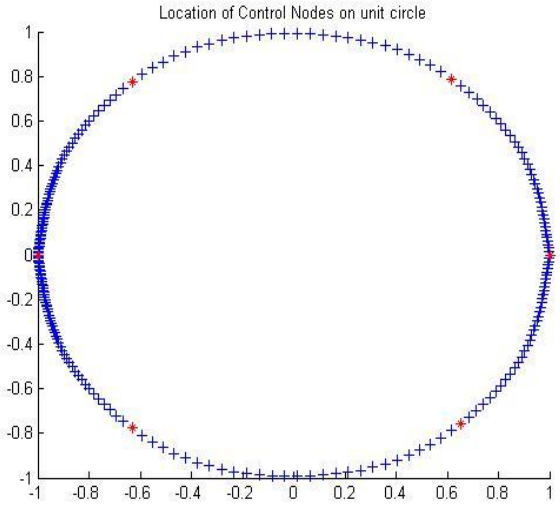
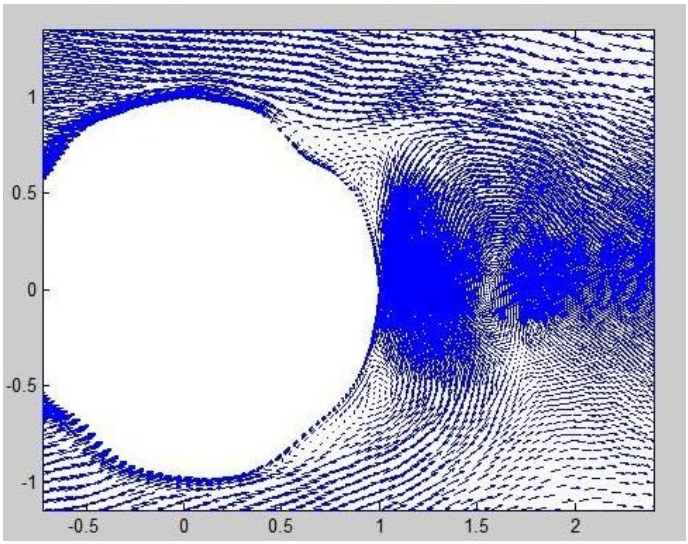


Figure 3.11. Unit circle and control nodes

About the problem definition first question comes to minds were number of maximum CFD solver iterations. This number was set to 10,000 but not to -3 (CFD stopping criteria can be a number or a ratio) because of one simple reason. When CFD residual graphed against solver iterations; oscillations after iteration 10,000 detected. Because of these oscillations it was meaningless to set higher stopping criteria since that would be a time consuming choice. For better understanding See Figure 3.12 and Figure 3.13; shows fluid flow around circle and vortex shedding at the downstream which is the phenomena cause oscillations in CFD residual and makes convergence harder.



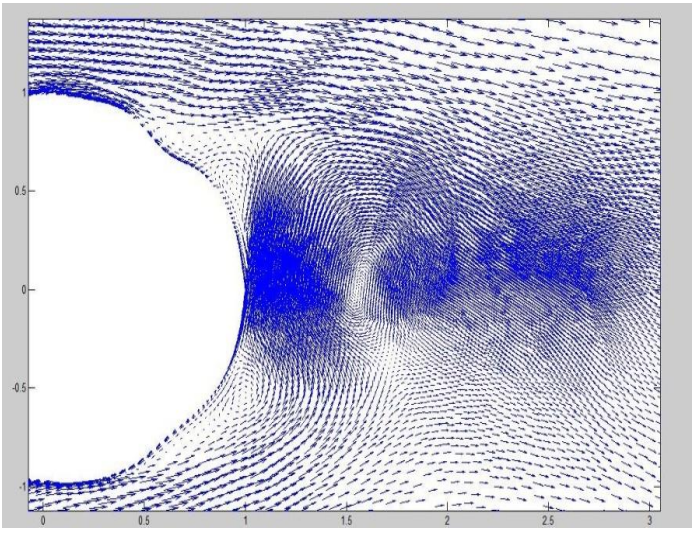


Figure 3.12. Fluid flow around unit circle

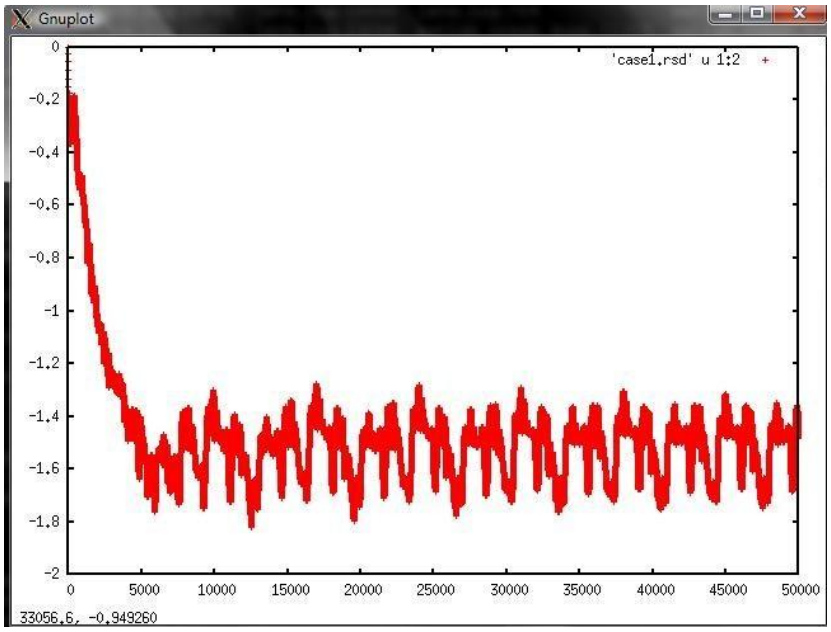
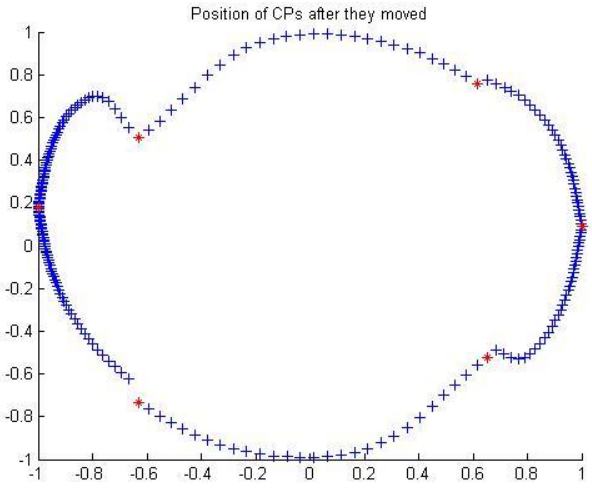
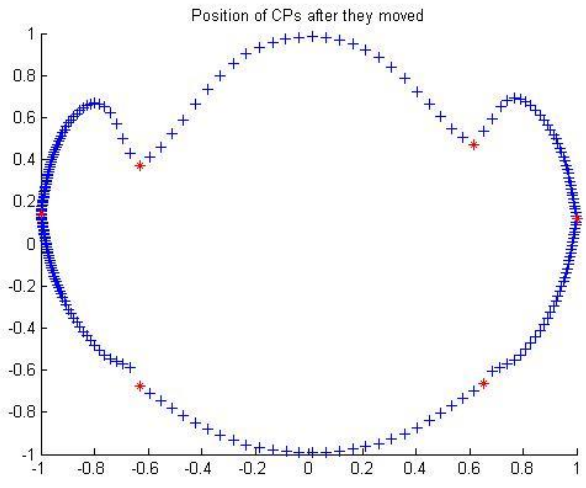


Figure 3.13. CFD residual Vs number of solver iterations

After deciding on number of CFD solver iterations, with problem definition above, everything was ready to proceed the first part, or the RBF mesh movement part of the algorithm. Being quite ambitious about this new implementation, maximum y displacement for control nodes was set to 0.45. After this choice some of the initial nests obtained with mesh movement shown in Figure 3.14.





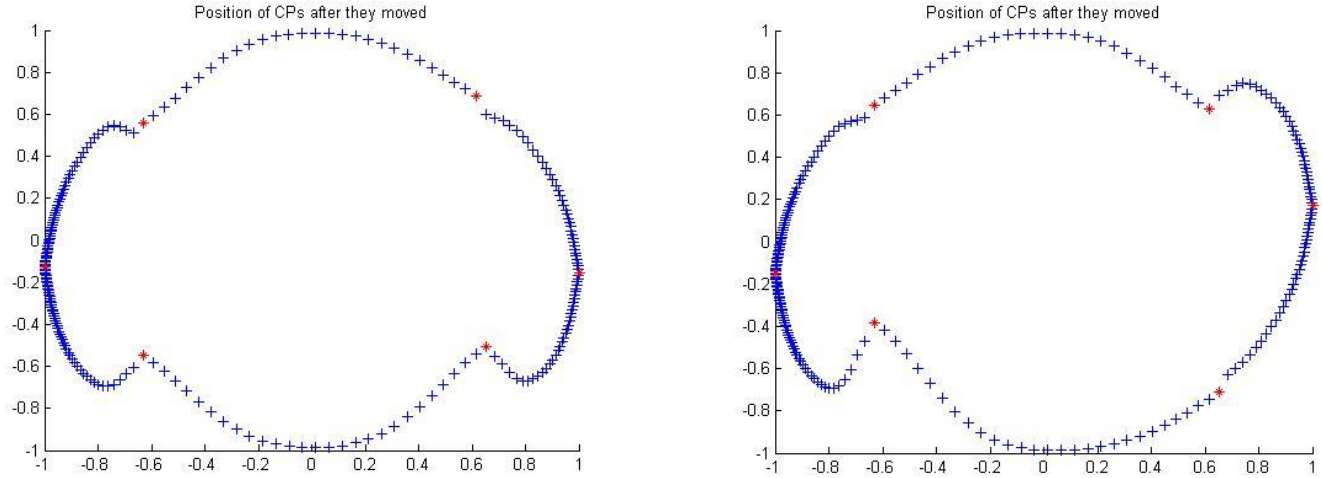
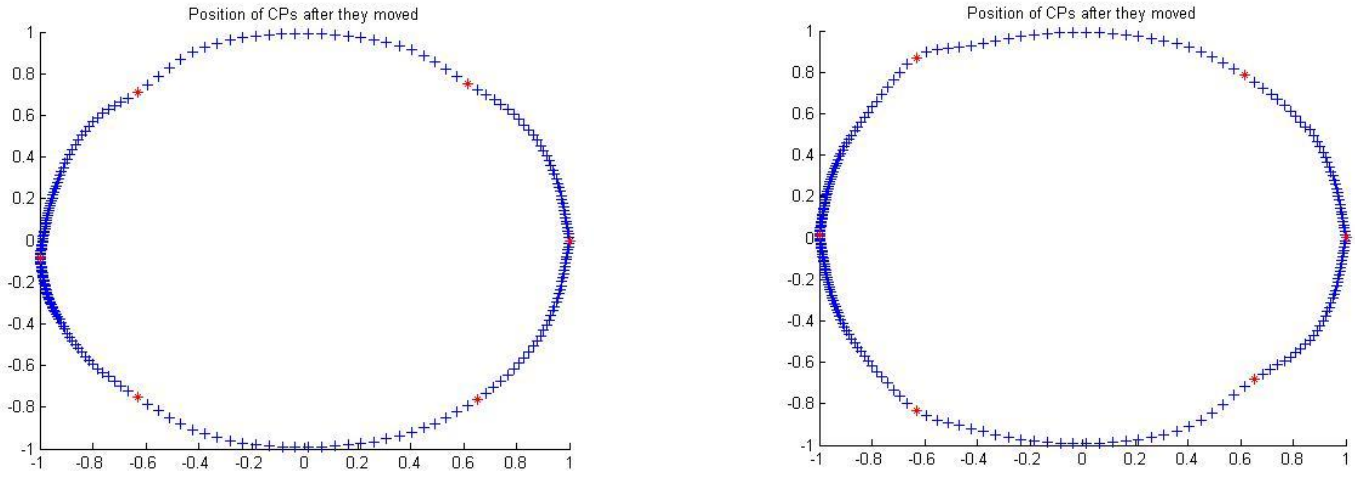


Figure 3.14. Initial nests after large deformation

Can be seen from Figure 3.14, mesh movement with such a huge displacement values was not successful. Unpleasant humps and discontinuous boundary geometry was observed. After these studies; it was learned that there should be a limit for maximum allowed displacement for RBF mesh movement algorithm to be successful. Taking this lesson; maximum allowed displacement for control nodes was set to 0.1 which gave the initial nests shown in Figure 3.15.



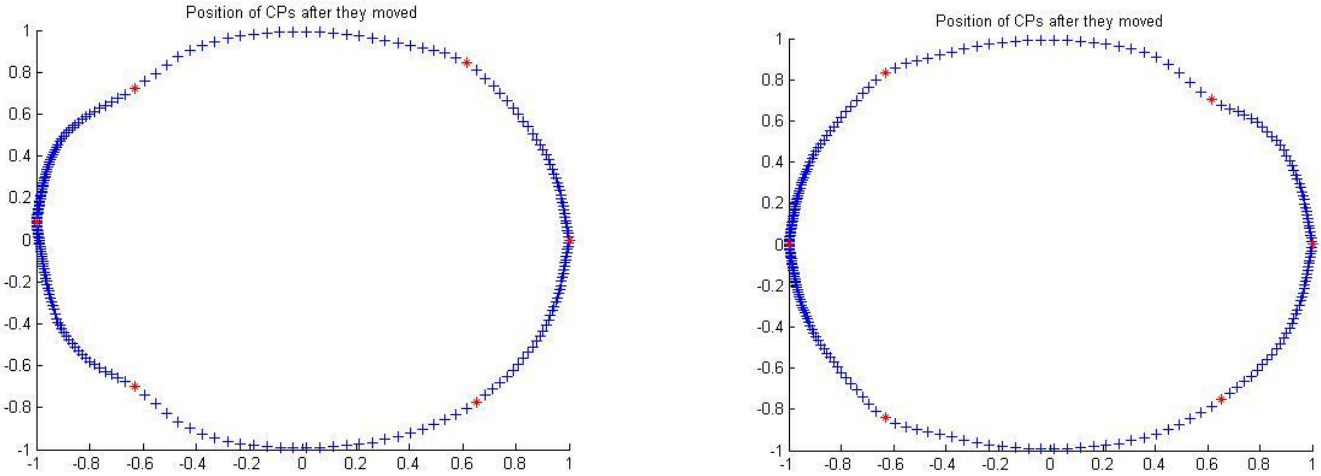


Figure 3.15. Initial nests after small deformation

With 0.1 maximum allowed y displacement; initial nests were with nice smooth boundary and without unpleasant humps. With 20 initial nests like the ones shown in Figure 3.15; everything was ready to move to the second or optimization part of the algorithm.

3.2.2. MCS WITH POD ON ARBITRARY GEOMETRY

After obtaining initial nests; MCS with POD implemented on unit circle geometry. While problem definition stayed the same; number of cuckoo iterations was set to 100. The results and optimum geometry obtained presented in Figure 3.16.

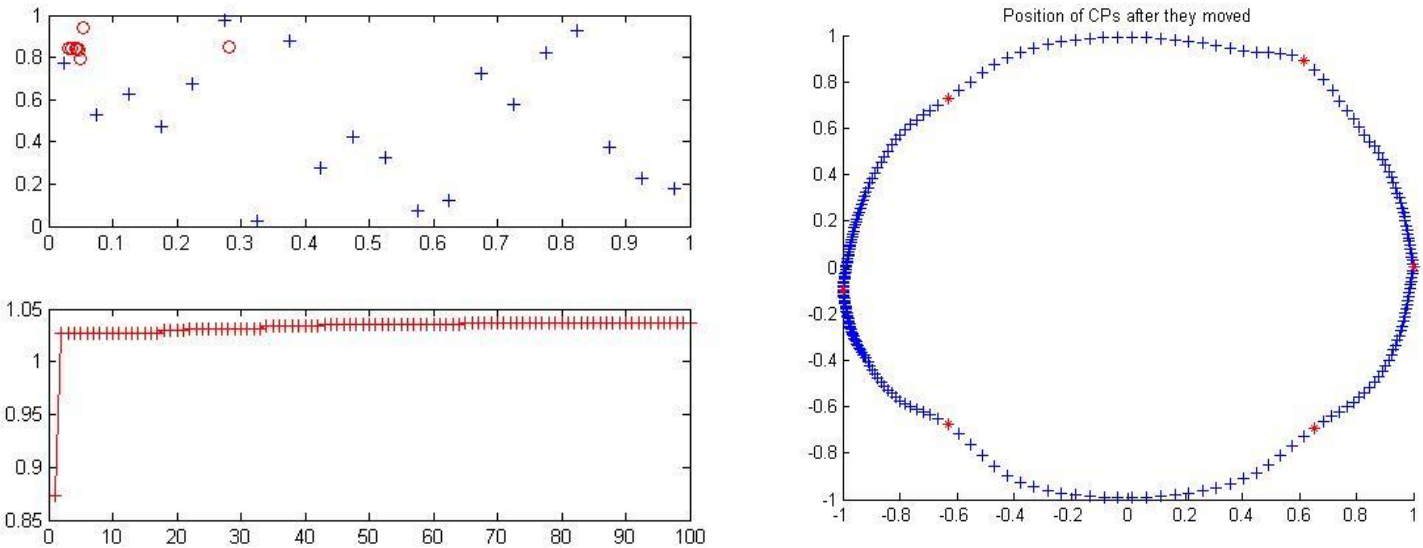


Figure 3.16. Evolution history (left) optimum geometry (right)

In Figure 3.16 on the left; the evolution history of the algorithm was shown with original nest positions in problem domain with blue +'s and final nest positions with red o's at the top; and fitness (lift drag ratio) evolution of the best nest on the bottom. It should be remembered that only 2 dimensions out of 6 was able to shown on top graph. When lift drag ratio was examined; it can be seen that it is started with the value of 0.8765 and improved up to 1.0327 at the end of the 100th cuckoo iteration. The circle geometry, on the other hand, improved to the geometry shown on the right side of Figure 3.16. Since the optimum geometry obtained needs to be improved further (it was far different than aerofoil geometry); it brings the idea of Multiple Modified Cuckoo Search or so called MMCS algorithm to the minds. MMCS suggests; further improving MCS; by starting a new MCS algorithm, with taking optimum geometry obtained from previous MCS as a starting geometry. Idea behind MMCS is; at second cycle, obtaining at least the same fitness value and same optimum geometry at the end of first cycle or luckily; ending up with further improved fitness values and geometries. Figure 3.17 shows the flowchart of MMCS algorithm for better understanding.

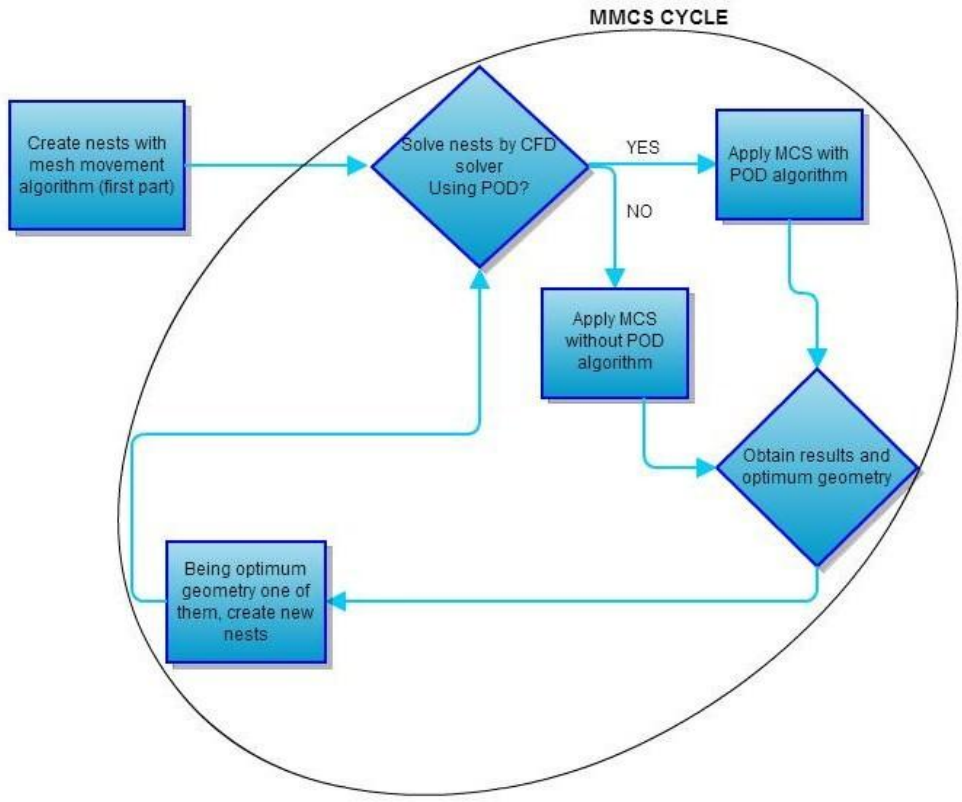


Figure 3.17. MMCS with/out POD flowchart

To implement MMCS with POD algorithm on the arbitrary geometry problem; the optimum geometry shown at Figure 3.16 was taken as a starting geometry for the new MMCS cycle. Including the optimum geometry above (Figure 3.16, on the right) 19 more initial nests were created with following exact same mesh movement algorithm and solved with university cluster. Control points for the new cycle, interestingly, located randomly to be strict to the totally heuristic optimization algorithm idea with MMCS. Proceeding to optimization part of second MMCS cycle; following results shown in Figure 3.18 were obtained at the end of 100th cuckoo iteration.

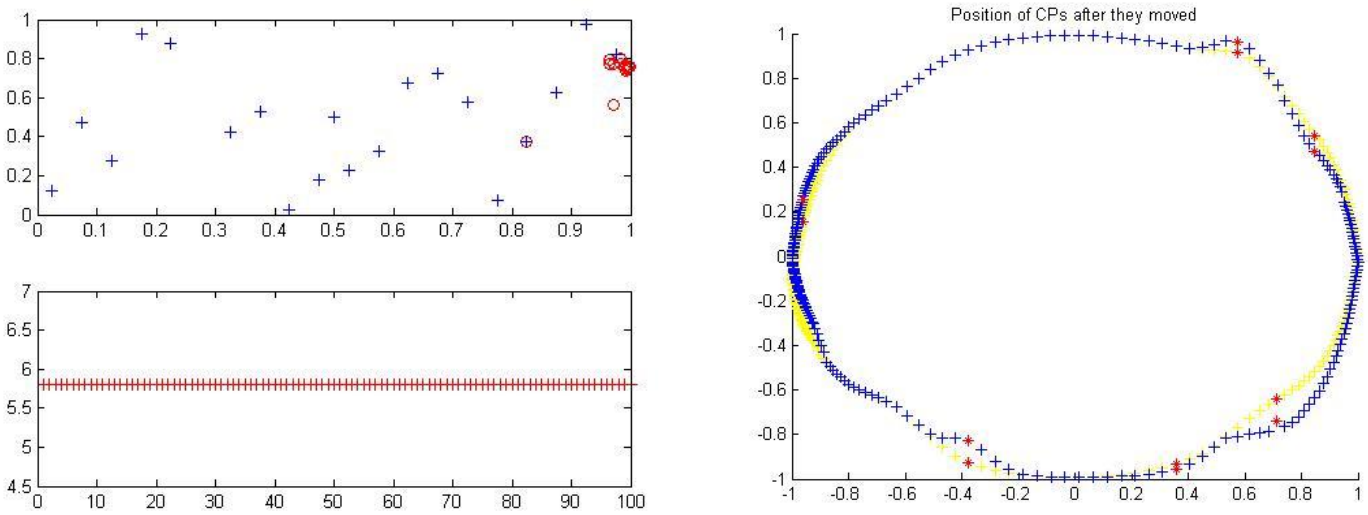


Figure 3.18. Evolution history (left) optimum geometry (right) after second MMCS cycle

In Figure 3.18 on the top left; original and final positions of the nests was shown whereas the fitness evolution of the best nest was on the bottom left side of the figure. It can clearly be seen there was no improvement in lift drag ratio; stayed still at 5.8072 throughout 100 iterations. Lets now try to see the logic behind this. After the first MMCS cycle; optimum geometry plus 19 more initial geometries (nests) created and sent to CFD solver. From the first cycle, reader may remember, maximum lift drag ratio obtained with POD was 1.0327. However when proceeded to second part of second MMCS cycle; starting value for lift drag ratio was 5.8072. This difference between lift drag ratios from the end of first cycle to the beginning of second cycle was because; 1.0327 was the value obtained with POD reconstruction (100th cuckoo iteration of first cycle), however; 5.8072 was the value calculated with 'real' pressure values, immediately after CFD solution (first cuckoo iteration of second cycle). So, it can be said; as a conclusion; the lift drag ratio calculated as 1.0327 with POD reconstructed pressure values was actually 5.8072 when calculated with real

pressure values. Obviously POD underestimates lift drag values and, this was why, throughout 100 iterations, it was not possible to end up with a lift drag ratio higher than 5.8072; since throughout 100 cuckoo iterations, lift drag ratio was calculated using pressure values 'guessed' with POD reconstruction. Although it was not possible to improve the fitness of best nest at second MMCS cycle; to move further, it was fair enough to check second best nest fitness evolution history to see if it was possible to pass 1.0327 (POD estimated) level obtained at the end of first cycle with POD reconstruction. See Figure 3.19 for fitness evolution history of second best nest at second MMCS cycle.

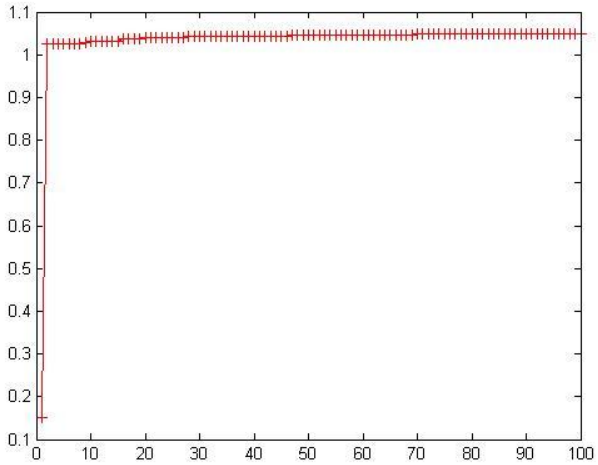


Figure 3.19. Second best nest fitness evolution, MMCS second cycle

At Figure 3.19 the maximum lift drag ratio reached by second best nest was 1.0495 which means it was possible to improve lift drag ratio further. Taking second best nest solution as a reference, third MMCS cycle initiated. As it was done previously, second best nest plus 19 more nests were created being strict to problem definition with randomly located control nodes. After second part (optimization part) was completed; the results were like in Figure 3.20.

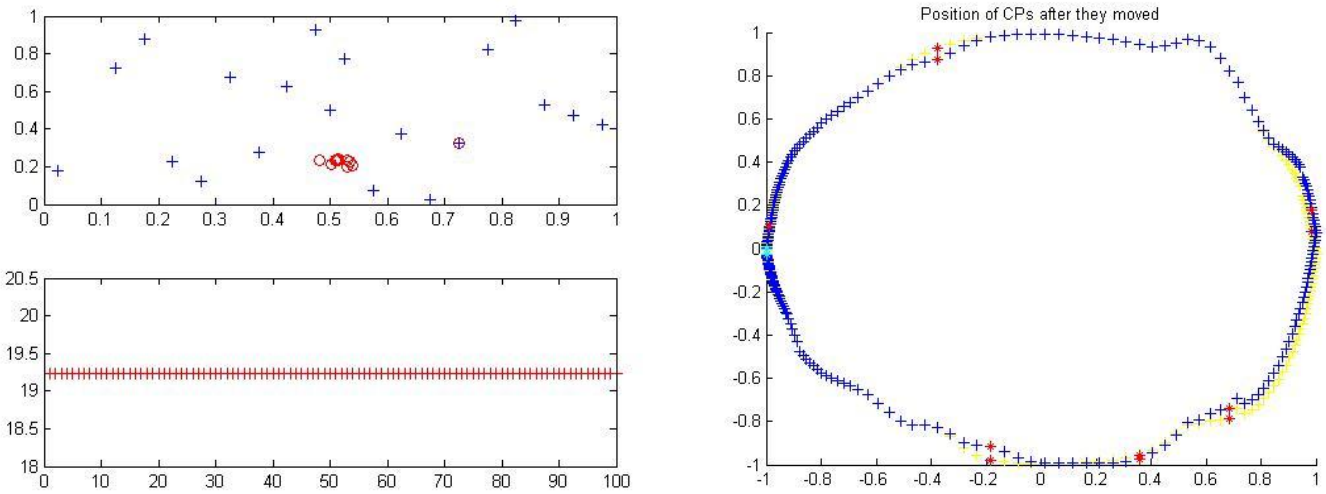


Figure 3.20. Evolution history (left) optimum geometry (right), after third MMCS cycle

Obviously, the maximum lift drag ratio further improved to 19.2323 at the end of third MMCS cycle and because of the same reason mentioned at second MMCS cycle; there was no jumps in lift drag ratio evolution at third cycle. Like it was done at the second cycle; to proceed to fourth cycle evolution history of the second best nest was examined. Figure 3.21 shows the fitness evolution of the second best nest which reaches to 1.0221 at the end of 100th cuckoo iteration. That means it was not possible to further improve fitness at the third cycle (max lift drag ratio of second best nest at second cycle was 1.0495) and under this conditions it could not make any sense to proceed to fourth cycle. As a bottom line, it can be said that MCS with POD stacked at the end of third cycle, furthermore; by looking at the optimum geometries obtained at the end of each cycle it can clearly be seen they were quite far from being aerofoil geometry with no distinctive 'squeeze' on starting unit circle geometry.

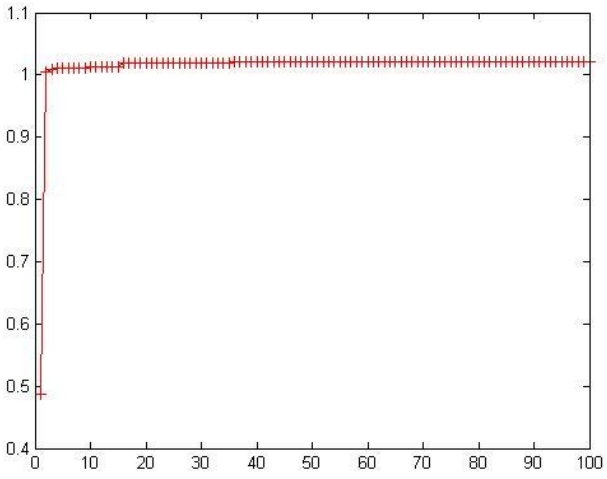


Figure 3.21. Second best nest evolution, MMCS third cycle

Table 3-7 below shows the summary of the results obtained in Section 3.2.2.

Table 3-7. MMCS with POD, summary of results

MMCS			
Fitness (lift drag ratio)	cycle1	cycle2	cycle3
with POD reconstruction with 'guessed' pressure values	1.0327	1.0495	1.02214
without POD after CFD solution with 'real' pressure values	5.8072	19.2323	19.2323

All discussions and results done so far in Section 3.2.2 leads to one simple conclusion. MMCS with POD does not work well on arbitrary geometry large deformation problems with initial geometry being unit circle. To see if the problem was POD or not; following section is devoted to examine same arbitrary geometry large deformation problem using MMCS without POD algorithm.

3.2.3. MMCS WITHOUT POD ON ARBITRARY GEOMETRY

When the studies done on arbitrary geometry using MMCS with POD algorithm did not give any convincing results, same arbitrary geometry problem decided to solve using MMCS without POD. By this way, first of all, aim was to end up with general purpose optimization algorithm for large deformation problems as well as to understand if the source of unsatisfactory results of Section 3.2.2 was POD reduction order method or not.

Lets remember here again the arbitrary geometry problem description. Unit circle with 6 control points located on it was chosen as the starting geometry in viscous air flow with Ma number being 0.8 and number of solver iterations being 10,000. All control nodes were allowed to displace in y direction only with maximum displacement value being 0.1. Other than these, number of nests was set to 20 and 10 MMCS cycles decided to run. Number of cuckoo iterations; for first 5 MMCS cycles, was set to 10, while for the remaining 5 MMCS cycles it was set to 20 and control nodes were relocated randomly at the end of each MMCS cycle.

By following the exact same RBF mesh movement algorithm explained at Section 3.2.1, initial nests again obtained to use within MMCS without POD algorithm. Initial nests were look like the ones shown in Figure 3.15 although they were not the same; because as mentioned before, mesh movement algorithm was randomly seeded in the beginning to keep the whole algorithm totally heuristic.

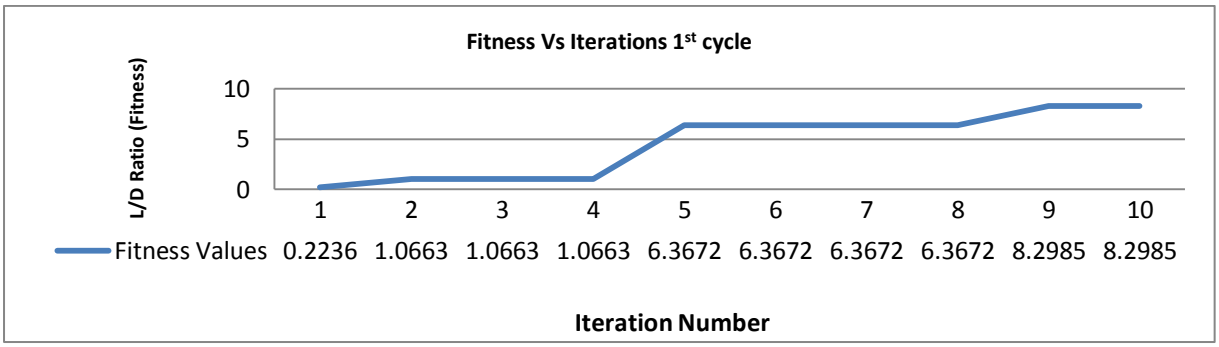
MMCS without POD was a time demanding algorithm, since at each cuckoo iteration; nests had to be solved by CFD solver to obtain nodal pressure values, rather than 'guessing' pressure values at nodes using POD. On the other hand, surely it gives trustable results since there was no estimation throughout the whole algorithm. Table 3-8 below added to represent the difference between time demands of MMCS with and without POD as well as to show how much time it was needed for one MMCS cycle.

Table 3-8. Time requirement of MMCS with/out POD

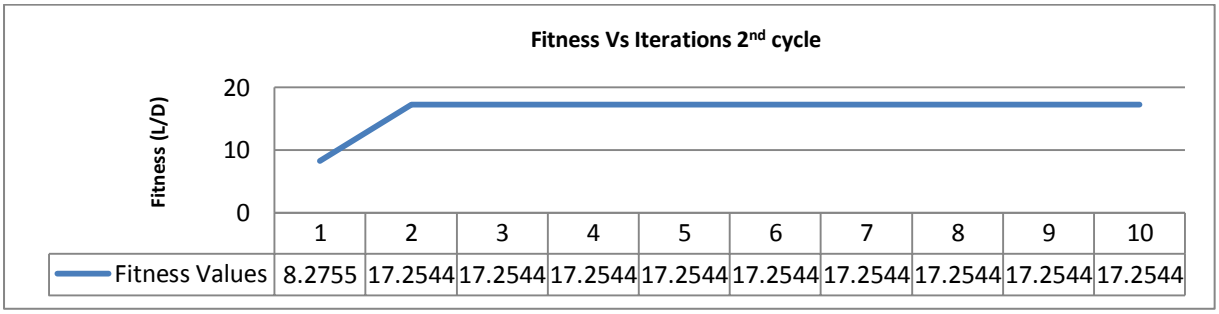
	Number of Control Nodes	Number of Nests	Number of Cuckoo Iterations	Wall Time (optimization)	Total Wall Time for 1 Cycle
With POD	6	20	100	~ 15 min/100 iter	~15 min
Without POD	6	20	10 or 20	~45min/1 iter	~450 or 900 min.

Surely using POD saves a lot of time however it is proved that the results obtained with POD were not satisfactory at all. Now lets implement MMCS without POD and examine the results.

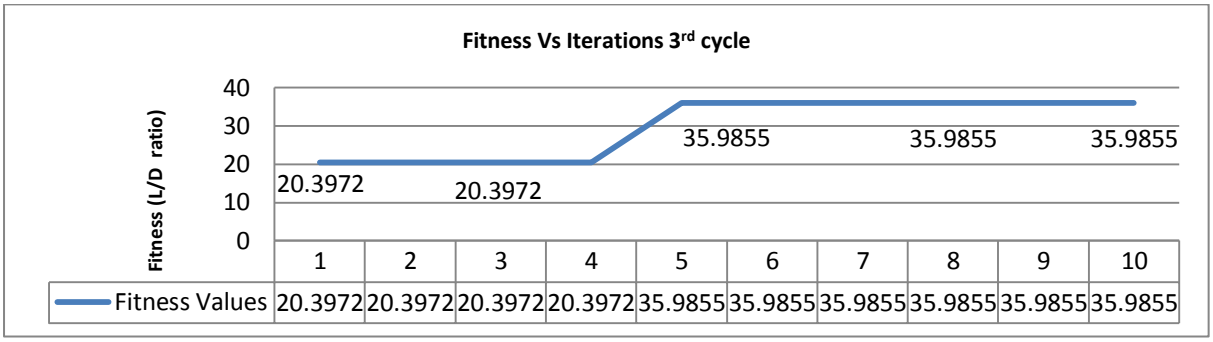
After obtaining the initial nests using regular mesh movement algorithm, finally, MMCS without POD applied on arbitrary unit circle. Output files obtained at each cycle were attached in Appendix section of the thesis; here the fitness evolution history and optimum geometries obtained at each cycle was presented.


Fitness Vs Iterations 1st cycle
L/D Ratio (Fitness)
10
5
0
1 2 3 4 5 6 7 8 9 10
Fitness Values 0.2236 1.0663 1.0663 1.0663 6.3672 6.3672 6.3672 6.3672 8.2985 8.2985
Iteration Number

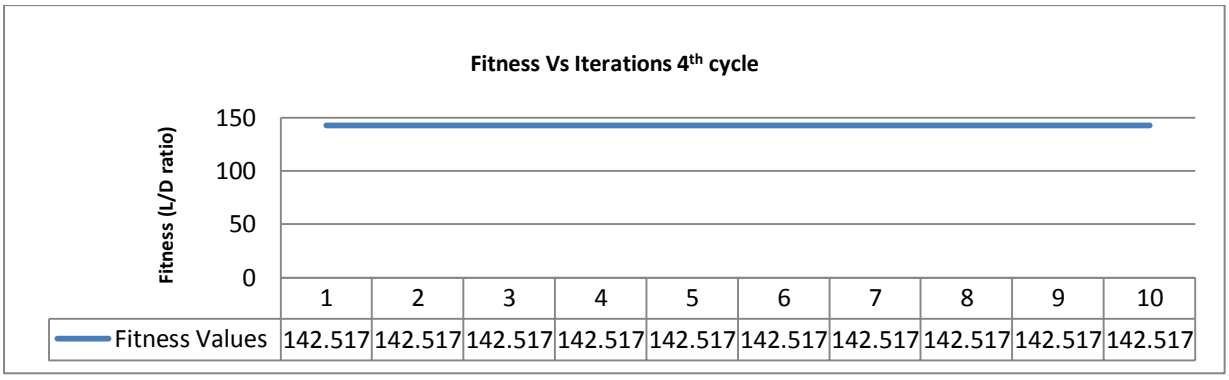
(a)


Fitness Vs Iterations 2nd cycle
Fitness (L/D)
20
10
0
1 2 3 4 5 6 7 8 9 10
Fitness Values 8.2755 17.2544 17.2544 17.2544 17.2544 17.2544 17.2544 17.2544 17.2544 17.2544

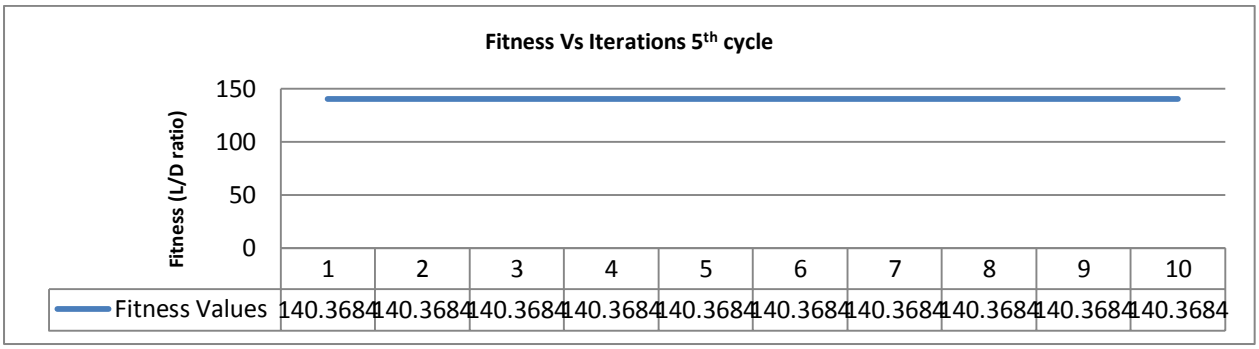
(b)


Fitness Vs Iterations 3rd cycle
Fitness (L/D ratio)
40
30
20
10
0
20.3972
20.3972
35.9855
35.9855
35.9855
1 2 3 4 5 6 7 8 9 10
Fitness Values 20.3972 20.3972 20.3972 20.3972 35.9855 35.9855 35.9855 35.9855 35.9855 35.9855

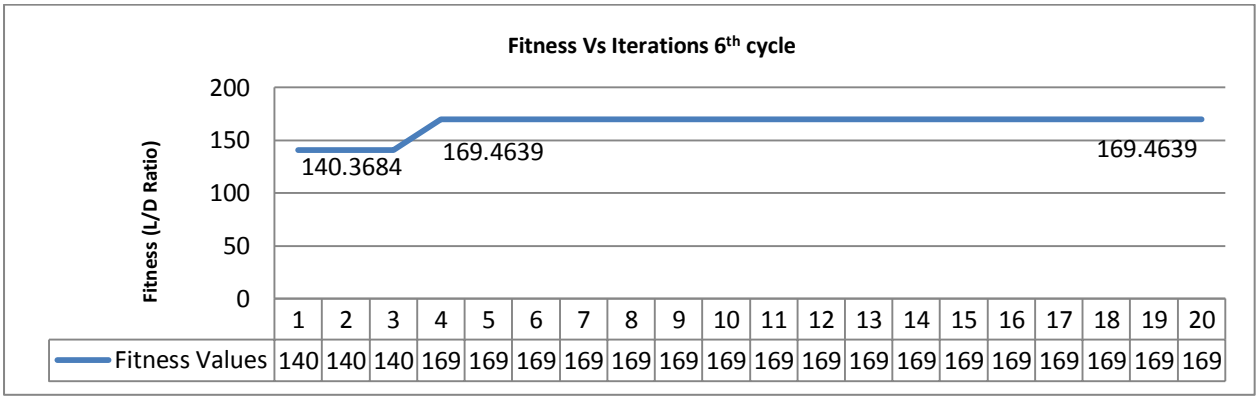
(c)


Fitness Vs Iterations 4th cycle
Fitness (L/D ratio)
150
100
50
0
1 2 3 4 5 6 7 8 9 10
Fitness Values 142.517 142.517 142.517 142.517 142.517 142.517 142.517 142.517 142.517 142.517

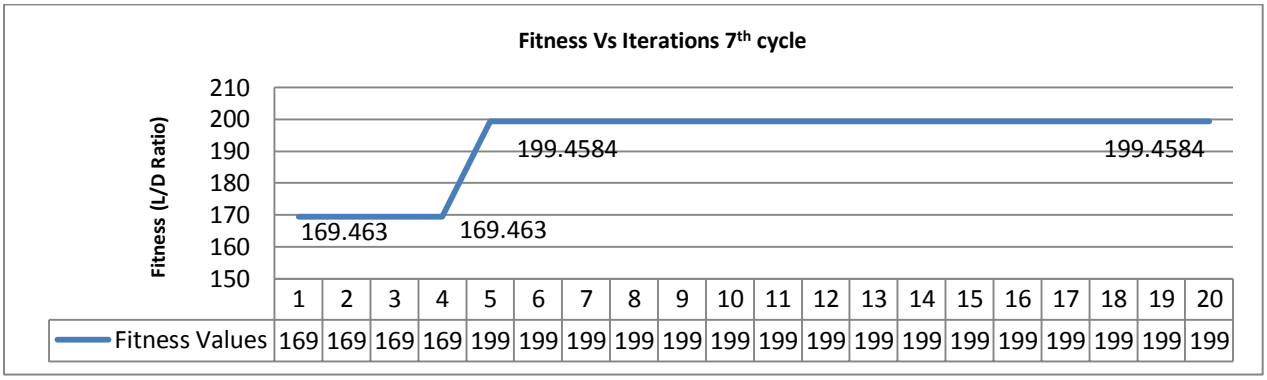
(d)


Fitness Vs Iterations 5th cycle
Fitness (L/D ratio)
150
100
50
0
1 2 3 4 5 6 7 8 9 10
Fitness Values 140.3684 140.3684 140.3684 140.3684 140.3684 140.3684 140.3684 140.3684 140.3684 140.3684

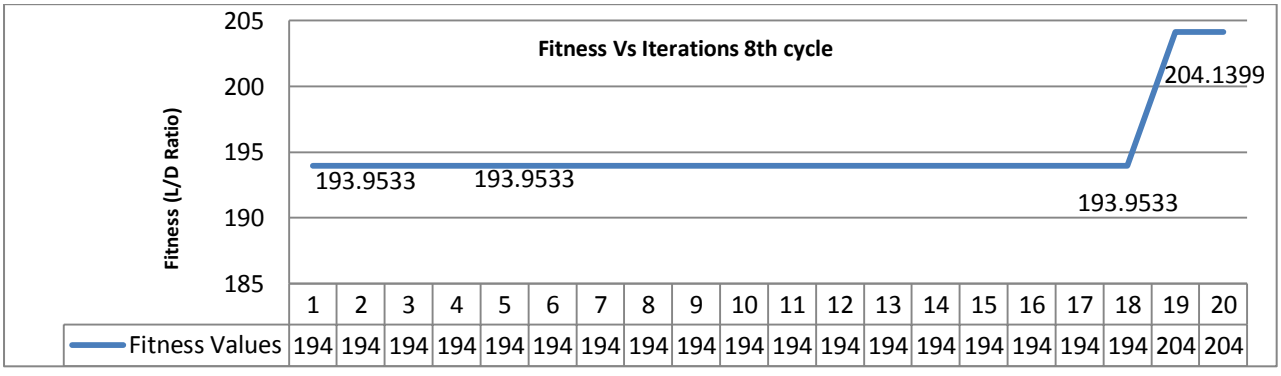
(e)


Fitness Vs Iterations 6th cycle
Fitness (L/D Ratio)
200
150
100
50
0
140.3684
169.4639
169.4639
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Fitness Values 140 140 140 169 169 169 169 169 169 169 169 169 169 169 169 169 169 169 169 169

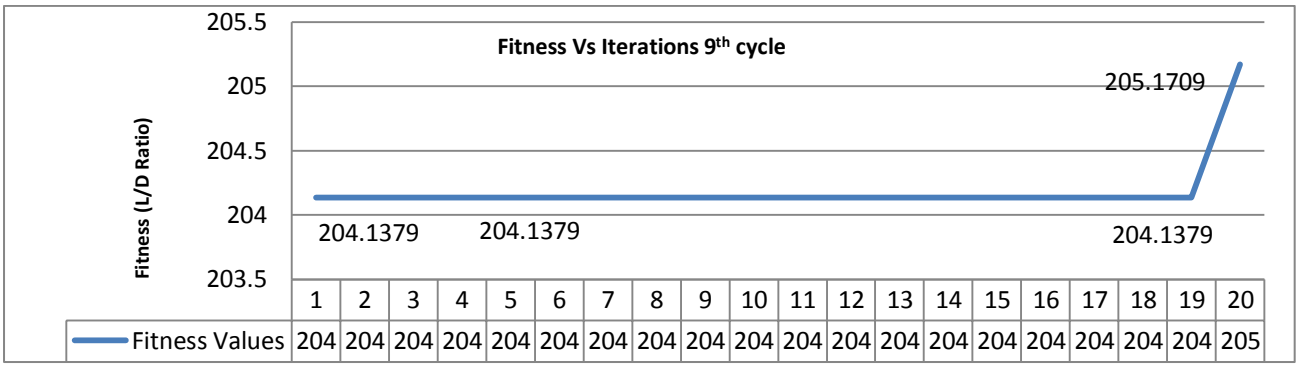
(f)


Fitness Vs Iterations 7th cycle
Fitness (L/D Ratio)
210
200
190
180
170
160
150
169.463
169.463
199.4584
199.4584
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Fitness Values 169 169 169 169 199 199 199 199 199 199 199 199 199 199 199 199 199 199 199 199

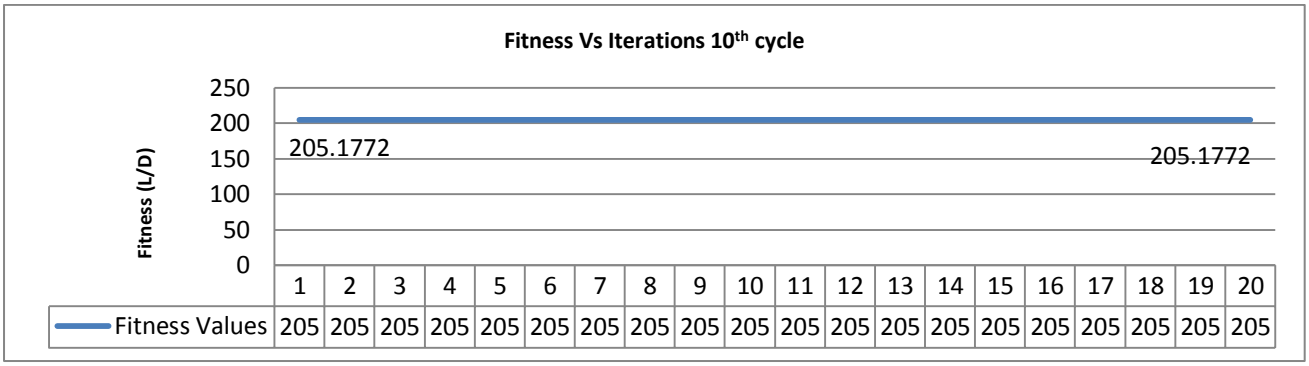
(g)


Fitness Vs Iterations 8th cycle
Fitness (L/D Ratio)
205
200
195
190
185
193.9533
193.9533
193.9533
204.1399
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Fitness Values 194 194 194 194 194 194 194 194 194 194 194 194 194 194 194 194 194 194 204 204

(h)



(i)



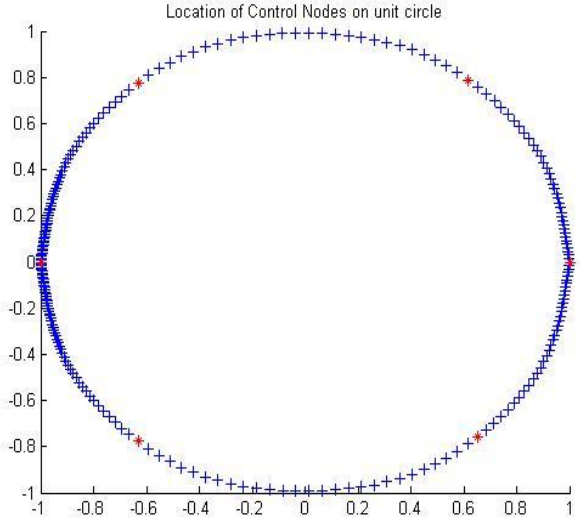
(j)

Figure 3.22. MMCS without POD fitness evolution through 10 cycles

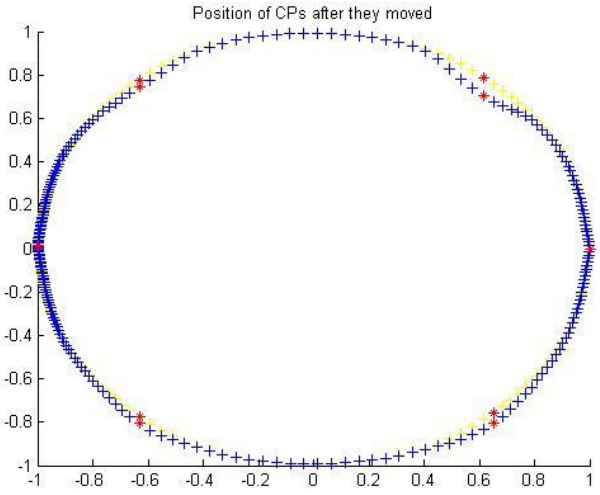
Figure 3.22 above shows the fitness evolution of MMCS without POD algorithm through 10 cycles. Starting from fitness (lift drag ratio) value of 0.2236 in the first cycle; algorithm was able to improve fitness up to 205.1772 at the tenth and last cycle. It can be said the algorithm completed the job well, improved the lift drag ratio values in each cycle bit by bit. It may be surprising to get reach the value of 205.1772 at the end since starting geometry was a unit circle evolving to an aerofoil and for simple aerofoils lift drag ratio is quite smaller than 205. This was because the lift drag ratio was calculated simply just using nodal pressure values (Equation 1.2 and Equation 1.3) and many other factors such as skin friction in drag calculation was ignored and in 2D, it is possible to get such high values especially when skin friction was ignored.

When examined closely, it can be observed that each round starts with final fitness value of previous round and improves it. This was the idea of MMCS which at least keeping the fitness the same, as in the previous cycle or improving it further. The small discrepancy between the last fitness value of former cycle and the first fitness value of the subsequent cycle may take

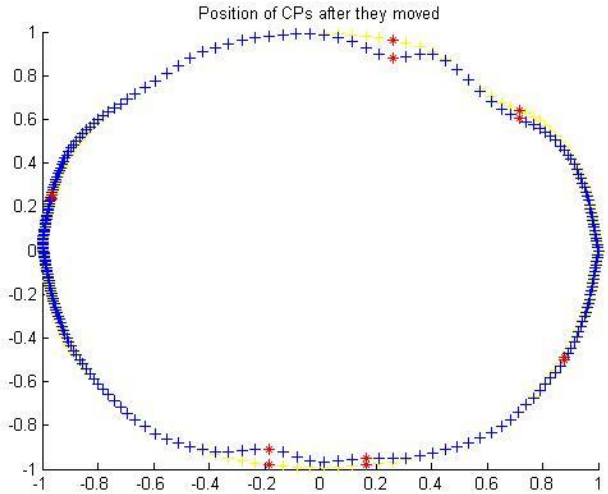
the attentions. The source of this discrepancy is simple. When one cycle ends, reader may remember, optimum geometry (which gives max. fitness value) of that cycle was taken and 19 more nests were created for the subsequent cycle. When those 20 nests; including the optimum geometry of former cycle, solved with CFD solver, because of the turbulent behaviour of residual, solver most probably stop not in exact same point as it did in the former cycle. Now lets see the evolution of unit circle geometry through 10 MMCS cycles and then examine it.



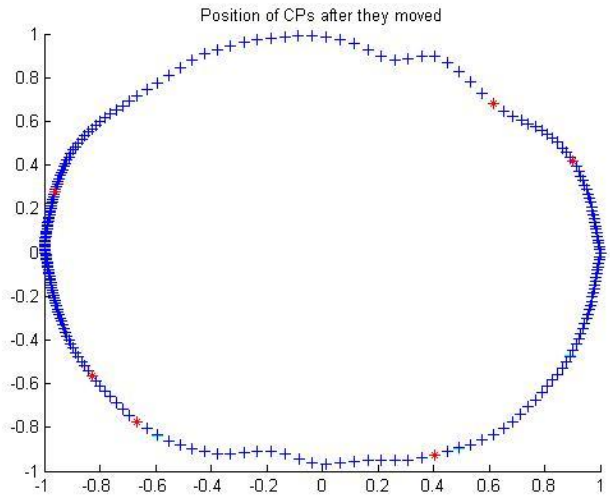
(a)



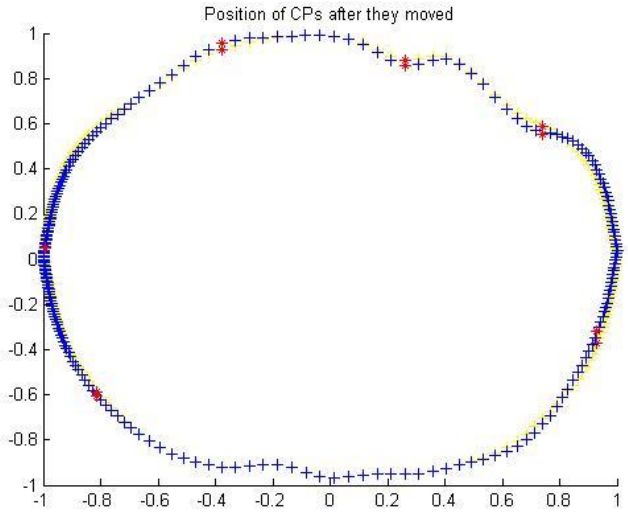
(b)



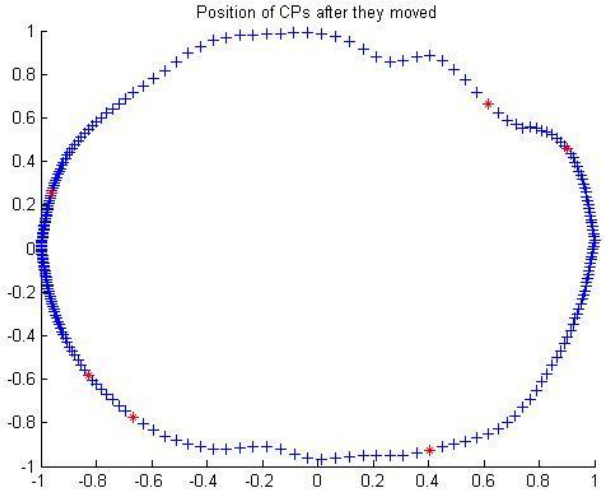
(c)



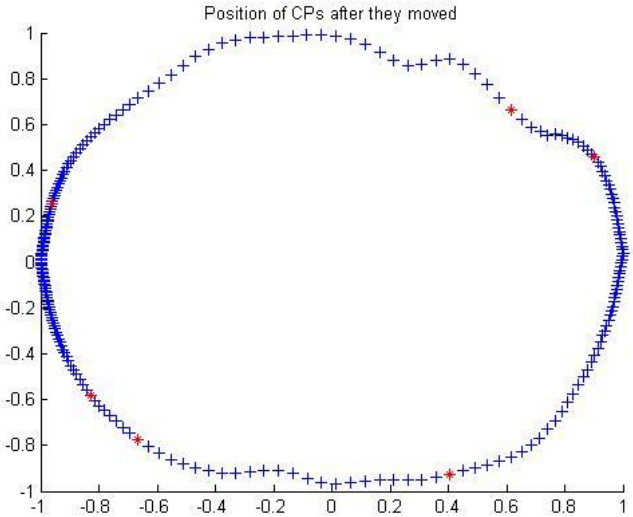
(d)


Position of CPs after they moved

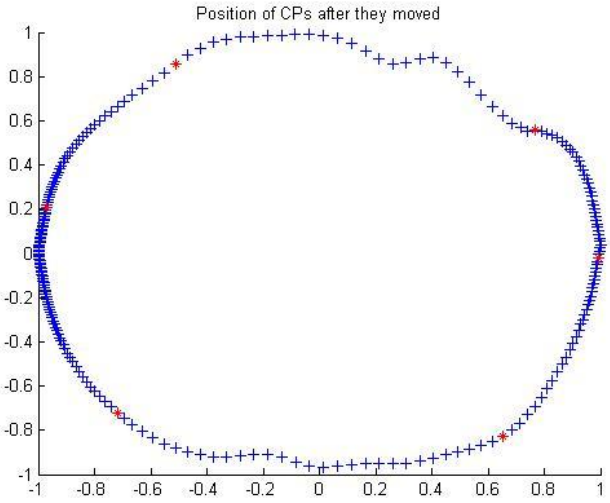
(e)


Position of CPs after they moved

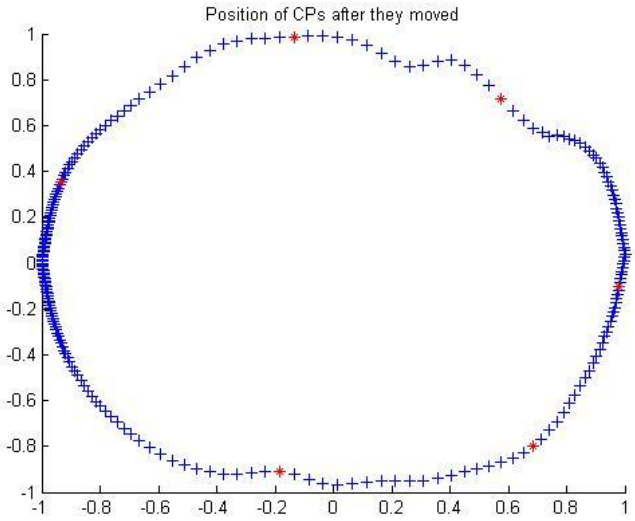
(f)


Position of CPs after they moved

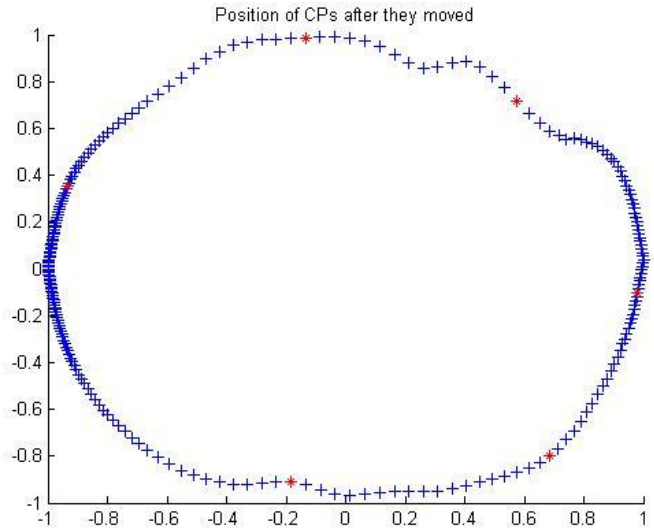
(g)


Position of CPs after they moved

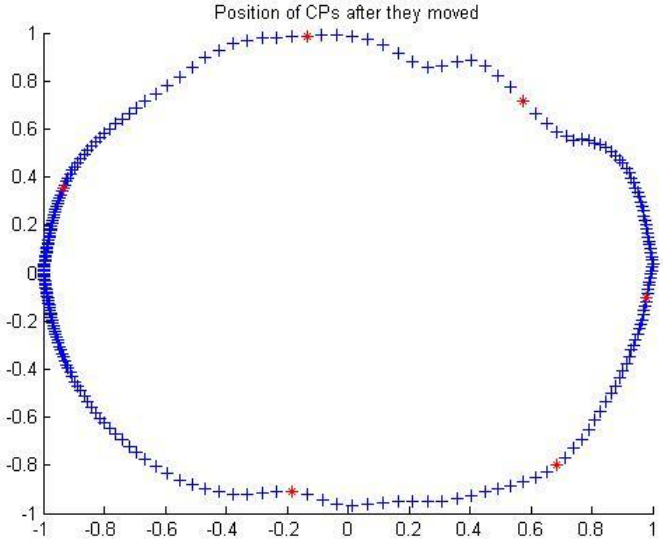
(h)



(i)



(j)



(k)

Figure 3.23. Evolution of history of arbitrary geometry thorugh 10 MMCS cycles (a) starting geometry (b) first (c) second (d) third (e) fourth (f) fifth (g) sixth (h) seventh (i) eighth (j) ninth (k) tenth cycles

Figure 3.23 above includes some important information. The evolution of arbitrary unit circle geometry from first cycle to the final cycle was presented. As a first deduction, the arbitrary unit circle geometry was displaced clearly in the desired direction or lets say it was 'squeezed'; on the other hand, the final geometry after 10^{th} cycle was not really look like an aerofoil geometry. So it can be said, MMCS without POD works on arbitrary geometry problem however, it needs to be run way more than 10 cycles to optimize an arbitrary geometry with such a big desired deformation, although it works, it is not the most efficient way for large deformation problems in the way that it is.

Mesh movement algorithm on the other hand, worked fine through 10 MMCS cycles; the final optimum boundary geometry was without humps or discontinuities. By looking at Figure 3.24, same can be said for the final mesh.

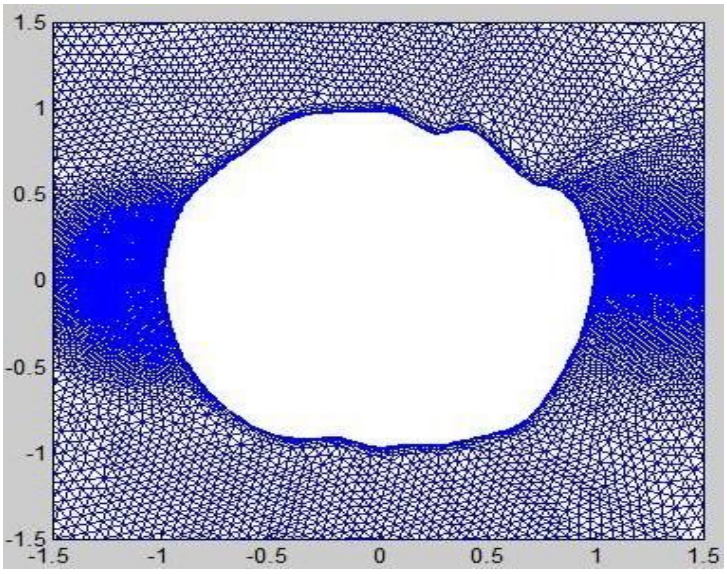


Figure 3.24. Computational mesh at the end of tenth MMCS cycle

Section 3.2.3 was devoted to examine arbitrary geometry large deformation problem using MMCS without POD; to discover if the POD construction was the problematic part of the algorithm after Section 3.2.2 MMCS with POD. At this point, it can be said MMCS without POD worked quite better than MMCS with POD; so it become obvious that POD was not the good choice for this type of problems. In the following section, MMCS with POD algorithm was going to be applied on simple aerofoil geometry to compare the results of the same algorithm on unit circle geometry to see, if the turbulent behaviour of the fluid around the circle was the cause for bad POD decomposition.

3.3. MMCS WITH POD ON AEROFOIL GEOMETRY

Applying MMCS with POD algorithm on aerofoil geometry was the last study of this thesis work. After discovering that POD did not work well on arbitrary unit circle geometry because of the turbulent fluid behaviour occurred at the downstream; now POD was decided to try on a geometry which has steady state viscous flow around it.

The aim was, as in the circle case; optimizing the lift drag ratio over the aerofoil boundary using MMCS with POD algorithm. The aerofoil geometry (See Figure 3.25) was quite small with chord length being equal to 1 and maximum thickness equal to 0.075. Ma number set to 0.5, 6 control points located on the boundary and each of them allowed to displace only on y direction with maximum displacement value being 0.01. To solve these 6 dimensional problem 20 nests were used. Angle of attack set to 0.0 degrees and stopping criteria for CFD solver set to -3.

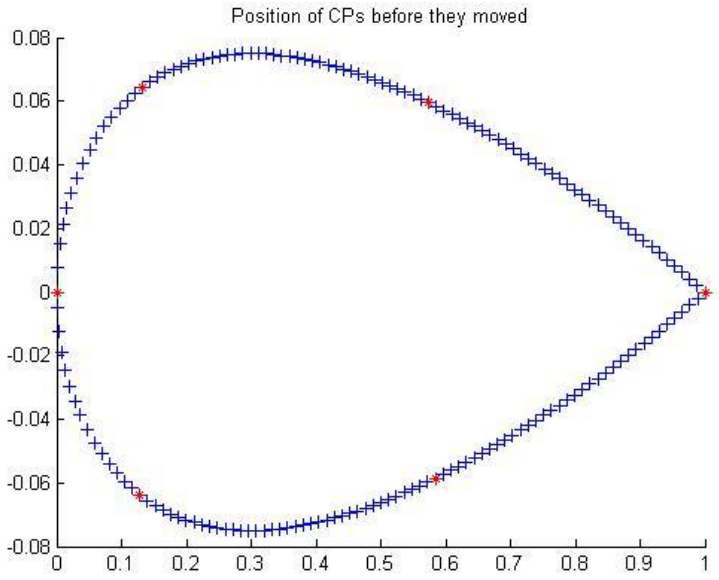


Figure 3.25. Aerofoil geometry

To show the difference between fluid flow and residual behaviour in circle geometry and aerofoil geometry; Figure 3.26 and Figure 3.27 were presented below.

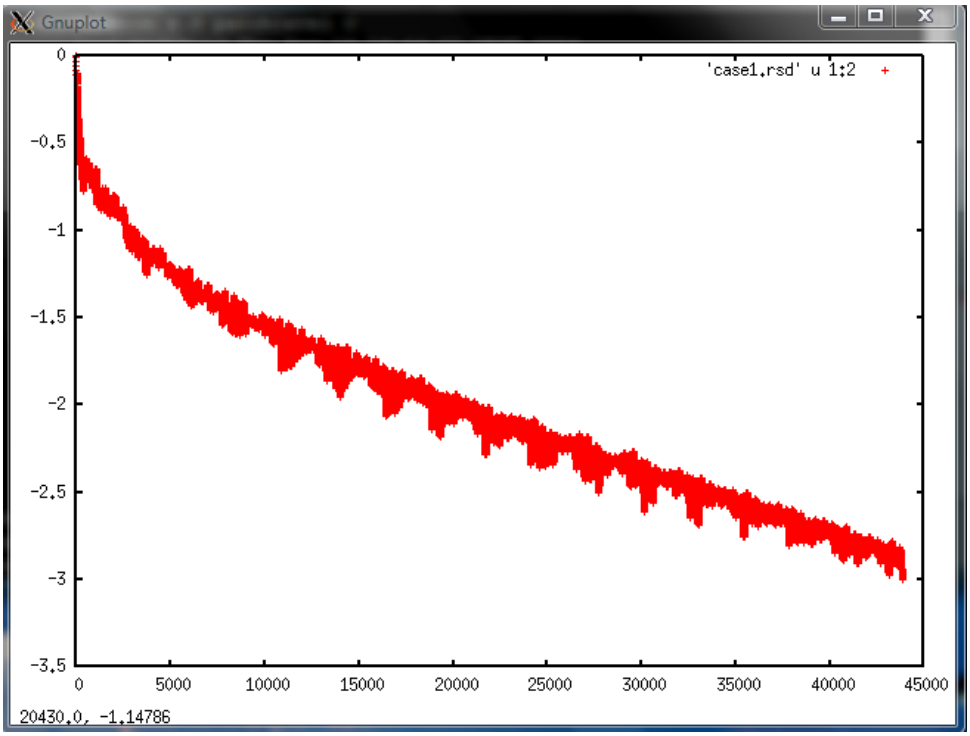


Figure 3.26. CFD residual behaviour, aerofoil geometry

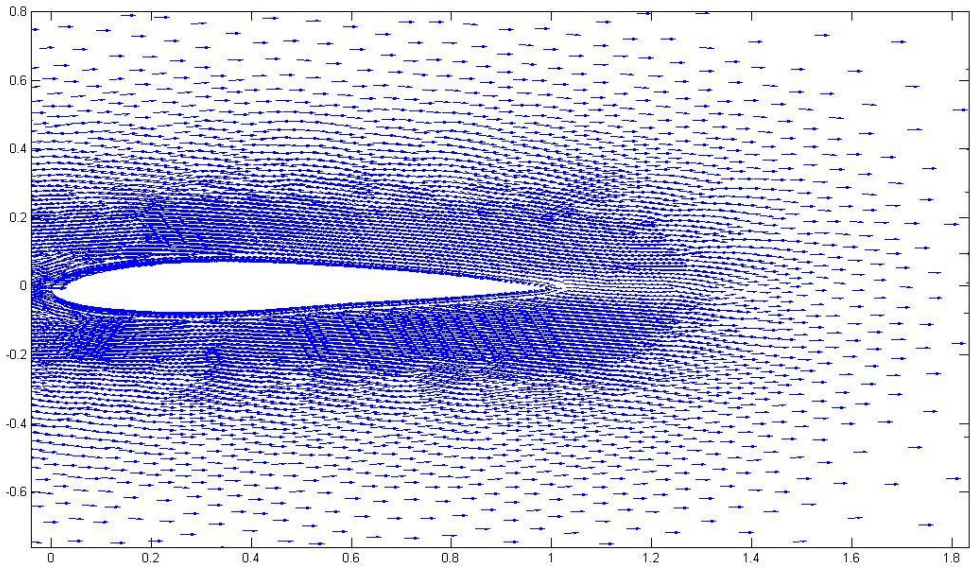


Figure 3.27. Fluid flow around aerofoil geometry

Clearly the fluid flow around aerofoil geometry was steady, without vortex shedding at the downstream and accordingly the evolution history of residual was without oscillations. Lets now examine, if POD works better in this kind of flow than it did with turbulent flow.

As usual algorithm started with creation of initial nests using RBF mesh movement algorithm. Some of the initial nests were like the ones shown in Figure 3.28.

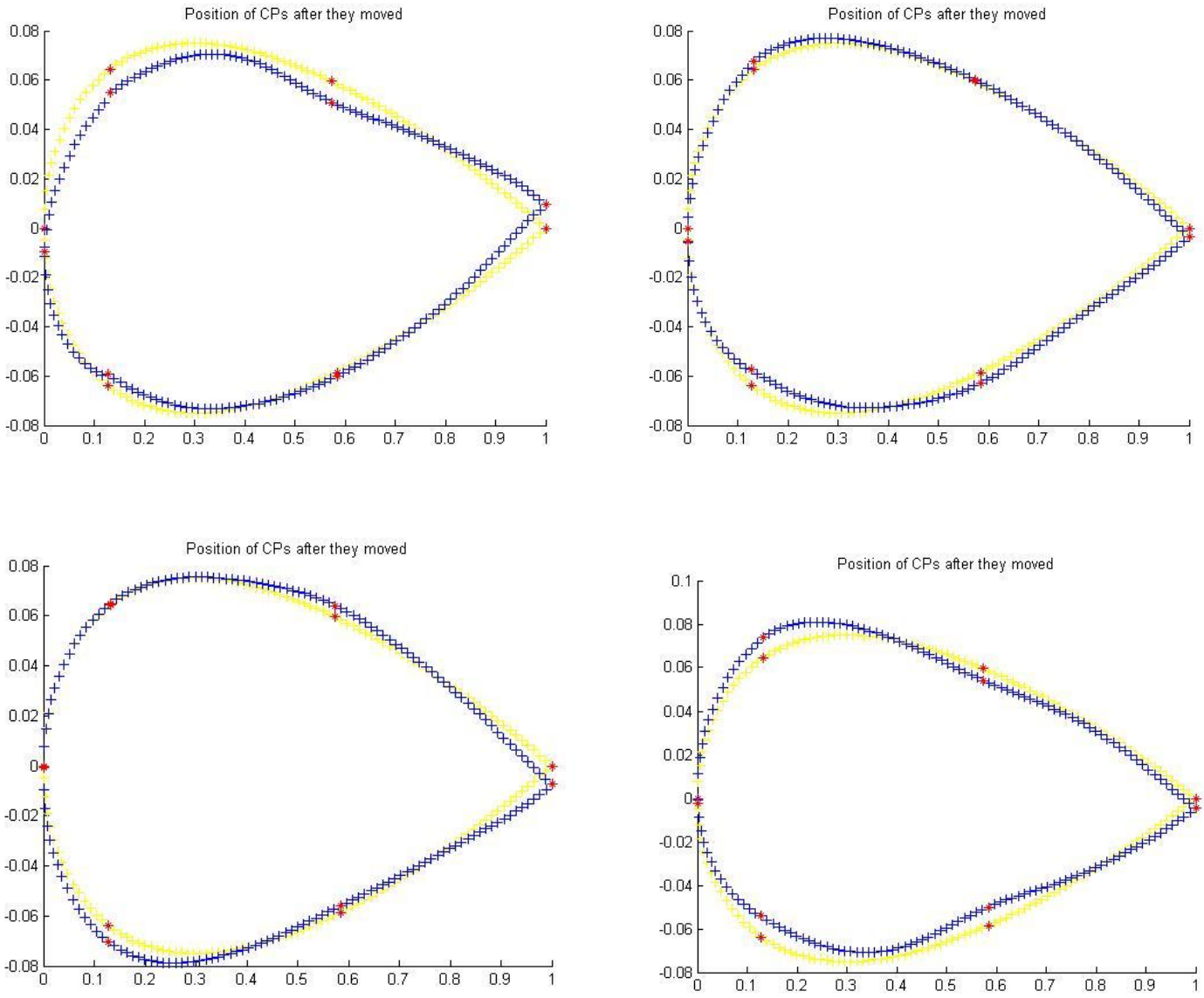


Figure 3.28. Initial nests

In the above Figure; yellow boundary was representing the starting geometry and blue boundary line was representing the displaced geometry. Obviously, RBF mesh movement algorithm worked fine, displaced boundaries were all smooth.

With the initial nests obtained and solved; optimization part of the algorithm initiated. In each of three MMCS cycles, location of control points were changed randomly and number of cuckoo iterations was set to 100.

The evolution history of the best nest and the nest positions as well as the optimum geometry obtained at the end of 1st MMCS cycle was shown in Figure 3.29.

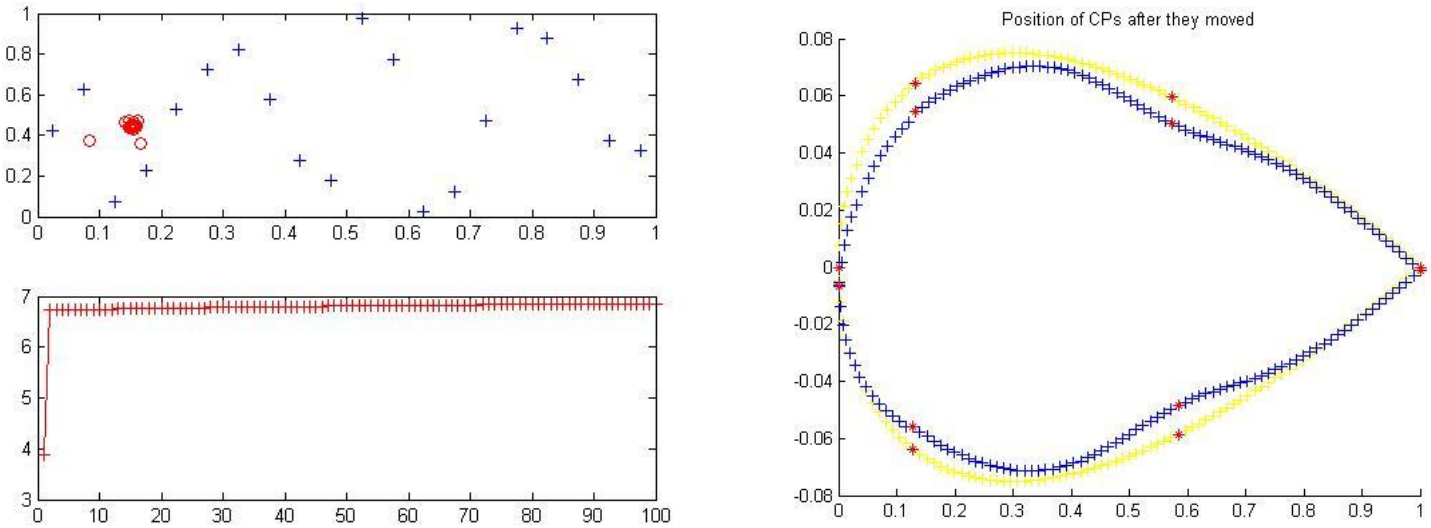


Figure 3.29. Nest positions and evolution history (left), optimum geometry (right)

As can be seen from Figure 3.29 above, fitness evolution of the best nest starts with a value of 3.96 and goes up to 6.75 at the end of 100 cuckoo iterations. When the best nest (with fitness being 6.75 according to MCS with POD calculations) sent back to CFD solver with 19 more recently created nests for second round; the real fitness value (the value calculated with nodal pressure values obtained after CFD solution) turn out to be 3.74. On the other hand, geometry was clearly squeezed in the desired direction with keeping the boundary smooth.

The second MMCS cycle, as expected, started with a fitness value of 3.74; and at the end of 100th cuckoo iteration that fitness value reached up to 7.33. Figure 3.30 below, represents the results of second round. The geometry was further squeezed with keeping boundary smooth and the fitness value was improved to 7.33 from 6.75. With all this nice results, taking the optimum geometry and newly created nests the optimization part of third and last cycle was initiated.

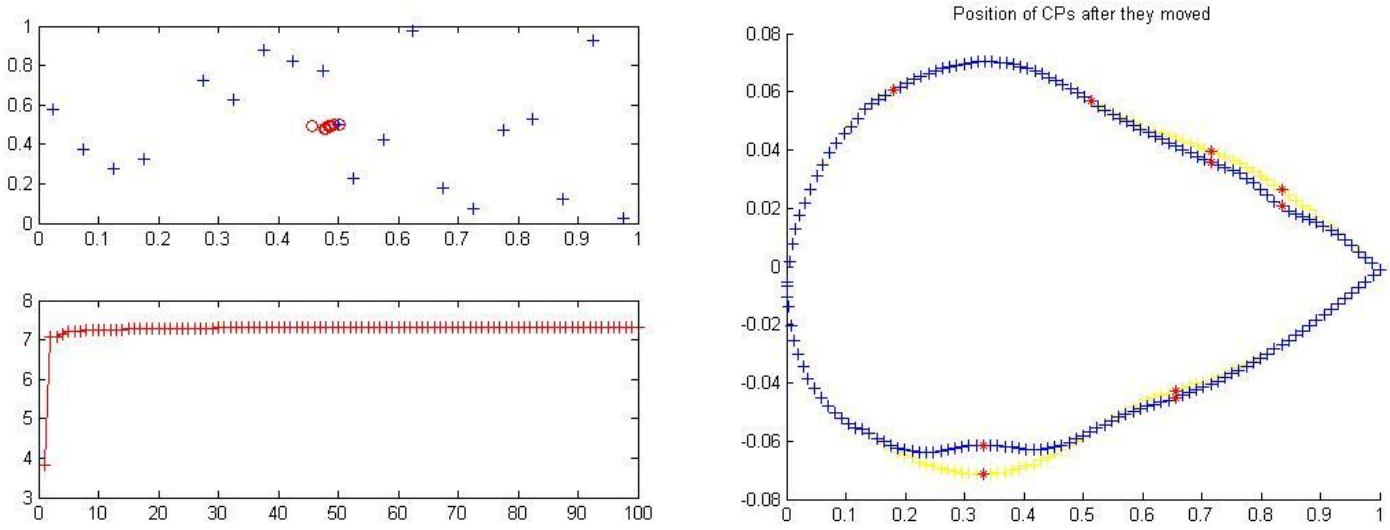


Figure 3.30. Evolution history and nest positions (left), optimum geometry (right)

The results obtained after third MMCS round were shown in Figure 3.31.

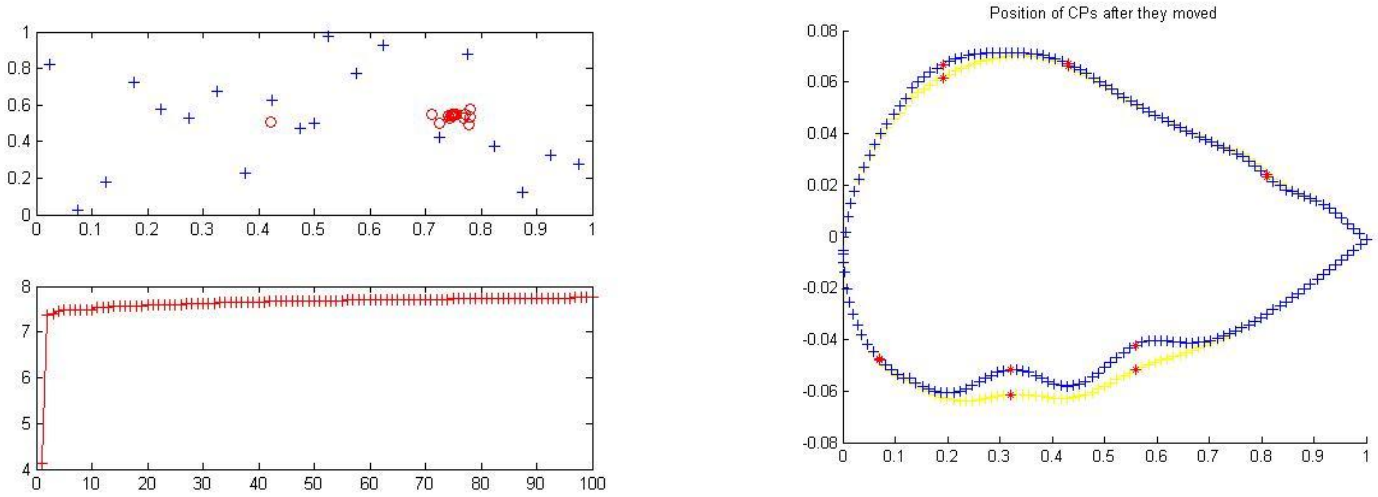


Figure 3.31. Evolution history and nest positions (left), optimum geometry (right)

At the end of third round, aerofoil geometry further squeezed in the desired way with improving fitness value to 7.7538 from 7.3344. Throughout three MMCS cycles, POD worked without any problem; by improving itself at the end of each cycle with a logical, expected optimized geometry at the end.

Throughout three MMCS rounds; it can clearly be seen that POD worked better than it did with circle geometry. This can be said by only looking at the optimum geometries obtained at the end of cycles but to present it scientifically Table 3-9 and Table 3-10 added below. In Table 3-9 and Table 3-10 the relative error for pressure and lift drag ratio (L/D) was presented. Relative error calculations done by using the values obtained at the end of each cycle with POD and the ones obtained in the beginning of each cycle with CFD solver.

Table 3-9. Error analysis, MMCS with POD, circle case

Circle Case	cycle 1	cycle 2	cycle 3
Relative Error (pressure)	0.7765	0.7355	0.6851
L/D (POD calculated)	1.0327	1.0495	1.0221
L/D (Solver calculated)	5.8072	19.2323	19.2323
Relative Error (L/D)	0.8222	0.9454	0.9469

Table 3-10. Error analysis, MMCS with POD, aerofoil case

Aerofoil Case	cycle 1	cycle 2	cycle 3
Relative Error (pressure)	0.6164	0.6348	0.6291
L/D (POD calculated)	6.7507	7.3344	7.7538
L/D (Solver calculated)	3.7498	4.1386	6.8990
Relative Error (L/D)	0.8003	0.7722	0.1239

Table 3-9 and Table 3-10 shows, POD worked better with a totally viscous and steady flow than it did with turbulent flow. The relative error was lower both in pressure and fitness in aerofoil case and that proves the argument.

This concludes Chapter 3; coming chapter, Chapter4, was devoted to conclusions based on studies presented here.

CHAPTER 4: CONCLUSIONS

4. CONCLUSIONS

4.1.CONCLUSIONS

Before start the conclusions about the work presented in this thesis; let's first remember what has been done so far. Here is the list reminds the main points covered:

- RBF mesh movement, MCS optimization and POD reduction order methods were explained in detail to make reader familiar with the concepts.
- Effect of input variables on the MCS with/out POD algorithm examined, logical input values determined.
- MCS with POD algorithm implemented on Bloodhound SSC air intake duct to optimize its geometry to minimize distortion at the engine inlet.
- Results obtained using MCS with POD on Bloodhound SSC were validated and compared with the results obtained after the implementation of MCS without POD on Bloodhound SSC.
- Optimizing an arbitrary geometry using MCS algorithm was set as a new challenge.
- To achieve the new aim, Multiple MCS (MMCS) was presented and implemented with POD on arbitrary unit circle geometry to optimize the lift drag ratio on the boundary.
- To validate MMCS with POD results on unit circle geometry; MMCS without POD applied on unit circle and results compared.
- To understand if POD works well on steady flow rather than it did on turbulent flow; MMCS with POD algorithm implemented on aerofoil geometry and results compared with MMCS with POD obtained on circle geometry.

These were the main points covered in this thesis; in the following part conclusions deducted were listed.

The conclusions learned out throughout this thesis work were listed here within the order that they mentioned.

- Number of solver iterations was an important design parameter which should be decided carefully for each problem. While optimizing Bloodhound SSC it was set to -3 (~20,000 iterations) ; however, in arbitrary geometry solutions it was set to 10,000.
- Increasing number of CFD solver iterations increases the wall time of the algorithm.
- MCS is a totally heuristic algorithm; means it is not possible to get the exact same optimized geometry with exact same fitness value twice.
- Increasing number of cuckoo iterations increases the wall time of the algorithm however it leads to better solutions.
- Increasing the number of nests with keeping problem dimension the same leads to better results however it increases the wall time.
- RBF type selected to use in POD reconstruction was an important parameter which has huge effect on solution. Choosing RBF as $\Phi(r) = |r|$ leads to the results closest to real ones.
- POD was a powerful tool to decrease wall time.
- MCS algorithm was trustable on subsonic and supersonic fluid regimes but not on supersonic regime because of changing particle physics.
- Air flowing around a unit circle creates vortexes at the downstream which causes oscillations in CFD solver residual.
- Radial Basis function mesh movement algorithm has some limits; large deformations should be covered step by step applying small displacements.
- MMCS algorithm was an efficient algorithm which worked fine on any geometry and helped to further improve MCS algorithm.
- MMCS with POD algorithm did not lead to desirable results on arbitrary unit circle algorithm because of poor POD reconstruction in turbulent flow.
- MMCS without POD worked fine and led to logical results; 'squeezed' the circle geometry in a sensible direction; however, it was not efficient enough to optimize such a large deformation problem in its current way; it would take a very long time for the algorithm to transform the arbitrary circle geometry to a realistic aerofoil geometry.
- MMCS with POD worked very well on optimizing the aerofoil geometry, led to logical, desired results efficiently.
- Applying POD algorithm on turbulent flow causes high errors, it is not efficient to use; however in steady state viscous flow POD is a powerful and useful tool to decrease the computational costs as well as the wall time enormously.

This concludes the Section 4.1 and conclusions of this work; following section devoted to some suggestions for the ones wants to work further on this area.

4.2. FUTURE SCOPE

As a continuation of the studies presented in this thesis, following suggestions can be followed:

- Improving the RBF mesh movement algorithm further to cover large deformations easily in one step, this can be done by searching different RBF types which may be more proper than Gaussian RBF for mesh movement.
- Improving the POD reconstruction in turbulent flow with applying some additional POD stabilization techniques.
- Improving the POD reconstruction with examining more than three RBF's examined here, to explore better RBF types.
- Selection of control nodes was always an important issue, at each MMCS cycle in this thesis work they were relocated randomly; however, by applying some neural networking studies on control point locations, relocation of them can be done more effectively.
- As a final suggestion, after completing the imperfections of the MCS with POD algorithm on 2D, extension of whole work to 3D would be useful.

References

[1] S. Walton, O. Hassan, K. Morgan, M.R.Brown., 2011. Modified cuckoo search: A new gradient free optimization algorithm. *Elsevier,* pp. 712-718.

[2] Liang. Y., 2001. Proper orthogonal decomposition and its applications- part i: Theory. *Journal of Sound and Vibration* .

[3] Felix, M. F., 2012. *Automatic optimization of the Bloodhound SSC air intake duct* , Swansea: s.n.

[4] A. de Boer, M.S. van der Schoot, L. Bijl., 2007. Mesh deformation based on radial basis function interpolation. *Elsevier.*

[5] Xiaochun Lu, B. T., 2010. *Research on mesh generation in the finite element numerical analysis based on radial basis function neural network.* s.l., s.n.

[6] J.C. Carr, W. Fright, R. K. Beatson, 1997. Surface interpolation with radial basis functions for medical imaging. *IEEE Publications* , pp. 96-107.

[7] Kara, C., 2012. *Mesh movement via Radial Basis Fucntions* , Swansea: s.n.

[8] Liu X, 2005. Fast dynamic grid deformation based on delaunay graph mapping. *Journal of Computational Physics* .

[9] Xin-She Yang, S. D., 2010. Engineering optimization by cuckoo search. pp. 1-17.

[10] Chatterjee, A., 2000. *An introduction to proper orthogonal decomposition,* s.l.: Current Science.

[11] Buhmann, M. D. ,2000, Radial Basis Functions

APPENDIX I: MAIN CODE OF MCS WITH POD ALGORITHM

Clear all

Take user input values

- Mach number
- Number of dimensions
- Number of nests
- Number of solver iterations
- Maximum x and y displacements

Randomly seed the algorithm

Generate Latin Hypercube Samples (LHC command on MATLAB©) (Scatter nests in problem domain)

Take each nest position and send it to mesh movement algorithm

Move the control nodes each time to the position offered by each nest using RBF mesh movement algorithm

Obtain initial nests (geometries)

Do pre-processing (creating .sol files) for initial geometries before solving them on cluster

Write input files (.inp) and send each initial geometry to the CFD solver

End of first (mesh movement) part of the algorithm

Calculate fitness values using the result files obtained from CFD solver

Extract modes of coefficients from the result files following POD algorithm

Sort initial nests by fitness and apply MCS on them (move the nests in the problem domain)

Create new nests (geometries using RBF mesh movement method) using the new nest positions

Proceed to nest cuckoo iteration

Guess the new fitness values for new nests following POD reconstruction algorithm

Sort the nests by fitness and start the new cuckoo iteration

APPENDIX I I: RESULT OUTPUTS OBTAINED USING MMCS WITHOUT POD ON CIRCLE GEOMETRY

```
< M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
              Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                             February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >>   Get fitness for new nests
>>    01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
>>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
Generation 1
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     5

The best configuration is:
  Columns 1 through 7

         0         0         0         0         0         0    0.0050

  Columns 8 through 12

   -0.0150   -0.0550   -0.0550   -0.0950   -0.0750

The highest L/D ratio is:
    0.2236

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
```

```
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
75769.master.cluster
75770.master.cluster
75771.master.cluster
75772.master.cluster
75773.master.cluster
75774.master.cluster
75775.master.cluster
75776.master.cluster
75777.master.cluster
75778.master.cluster
75779.master.cluster
75780.master.cluster
75781.master.cluster
75782.master.cluster
75783.master.cluster
75784.master.cluster
75785.master.cluster
75786.master.cluster
75787.master.cluster
75788.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20



Generation 10
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     2

The best configuration is:
  Columns 1 through 7

         0         0         0         0         0         0    0.0163

  Columns 8 through 12

   -0.0022   -0.0330   -0.0862   -0.0475   -0.0306

The highest L/D ratio is:
    8.2985
```

```
Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
76009.master.cluster
76010.master.cluster
76011.master.cluster
76012.master.cluster
76013.master.cluster
76014.master.cluster
76015.master.cluster
76016.master.cluster
76017.master.cluster
76018.master.cluster
76019.master.cluster
76020.master.cluster
76021.master.cluster
76022.master.cluster
76023.master.cluster
76024.master.cluster
76025.master.cluster
76026.master.cluster
76027.master.cluster
76028.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
```

```
< M A T L A B (R) >
                  Copyright 1984-2010 The MathWorks, Inc.
                Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                              February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >>   Get fitness for new nests
>>    01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
>>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
Generation 1
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
    16

The best configuration is:
     0     0     0     0     0     0     0     0     0     0     0     0

The highest L/D ratio is:
    8.2755

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
```

```
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
77028.master.cluster
77029.master.cluster
77030.master.cluster
77031.master.cluster
77032.master.cluster
77033.master.cluster
77034.master.cluster
77035.master.cluster
77036.master.cluster
77037.master.cluster
77038.master.cluster
77039.master.cluster
77040.master.cluster
77041.master.cluster
77042.master.cluster
77043.master.cluster
77044.master.cluster
77045.master.cluster
77046.master.cluster
77047.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20



Generation 10
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     2

The best configuration is:
  Columns 1 through 7

         0         0         0         0         0         0    0.0175

  Columns 8 through 12

   -0.0839   -0.0398   -0.0104    0.0345    0.0682

The highest L/D ratio is:
   17.2544

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
```

```
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
77208.master.cluster
77209.master.cluster
77210.master.cluster
77211.master.cluster
77212.master.cluster
77213.master.cluster
77214.master.cluster
77215.master.cluster
77216.master.cluster
77217.master.cluster
77218.master.cluster
77219.master.cluster
77220.master.cluster
77221.master.cluster
77222.master.cluster
77223.master.cluster
77224.master.cluster
77225.master.cluster
77226.master.cluster
77227.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20



Warning: No display specified.  You will not be able to display graphics on the
screen.
Warning: No window system found.  Java option 'MWT' ignored

                            < M A T L A B (R) >
                  Copyright 1984-2010 The MathWorks, Inc.
                Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                             February 5, 2010
```

```
  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >>   Get fitness for new nests
>>    01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
>>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
Generation 1
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     3

The best configuration is:
  Columns 1 through 7

         0         0         0         0         0         0   -0.0050

  Columns 8 through 12

   -0.0350    0.0250    0.0250    0.0950   -0.0250

The highest L/D ratio is:
   20.3972

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
```

```
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
77414.master.cluster
77415.master.cluster
77416.master.cluster
77417.master.cluster
77418.master.cluster
77419.master.cluster
77420.master.cluster
77421.master.cluster
77422.master.cluster
77423.master.cluster
77424.master.cluster
77425.master.cluster
77426.master.cluster
77427.master.cluster
77428.master.cluster
77429.master.cluster
77430.master.cluster
77431.master.cluster
77432.master.cluster
77433.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20


Generation 10
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     1

The best configuration is:
  Columns 1 through 7

         0         0         0         0         0         0   -0.0000

  Columns 8 through 12

    0.0000   -0.0002    0.0011    0.0003   -0.0001

The highest L/D ratio is:
   35.9855

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
```

```
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
77594.master.cluster
77595.master.cluster
77596.master.cluster
77597.master.cluster
77598.master.cluster
77599.master.cluster
77600.master.cluster
77601.master.cluster
77602.master.cluster
77603.master.cluster
77604.master.cluster
77605.master.cluster
77606.master.cluster
77607.master.cluster
77608.master.cluster
77609.master.cluster
77610.master.cluster
77611.master.cluster
77612.master.cluster
77613.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20

Warning: No display specified.  You will not be able to display graphics on the
screen.
Warning: No window system found.  Java option 'MWT' ignored

                         < M A T L A B (R) >
               Copyright 1984-2010 The MathWorks, Inc.
             Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                          February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.
```

```
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >>   Get fitness for new nests
>>    01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
>>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
Generation 1
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
    17

The best configuration is:
  Columns 1 through 7

         0         0         0         0         0         0   -0.0250

  Columns 8 through 12

    0.0250   -0.0250   -0.0350    0.0550   -0.0150

The highest L/D ratio is:
  142.5166

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
```

```
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
77767.master.cluster
77768.master.cluster
77769.master.cluster
77770.master.cluster
77771.master.cluster
77772.master.cluster
77773.master.cluster
77774.master.cluster
77775.master.cluster
77776.master.cluster
77777.master.cluster
77778.master.cluster
77779.master.cluster
77780.master.cluster
77781.master.cluster
77782.master.cluster
77783.master.cluster
77784.master.cluster
77785.master.cluster
77786.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20


Generation 10
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     1

The best configuration is:
  Columns 1 through 7

         0         0         0         0         0         0   -0.0250

  Columns 8 through 12

    0.0250   -0.0250   -0.0350    0.0550   -0.0150

The highest L/D ratio is:
  142.5166

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
```

```
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
78015.master.cluster
78016.master.cluster
78017.master.cluster
78018.master.cluster
78019.master.cluster
78020.master.cluster
78021.master.cluster
78022.master.cluster
78023.master.cluster
78024.master.cluster
78025.master.cluster
78026.master.cluster
78027.master.cluster
78028.master.cluster
78029.master.cluster
78030.master.cluster
78031.master.cluster
78032.master.cluster
78033.master.cluster
78034.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20

Warning: No display specified.  You will not be able to display graphics on the
screen.
Warning: No window system found.  Java option 'MWT' ignored

                          < M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
               Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                             February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
```

```
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >>   Get fitness for new nests
>>    01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
>>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
Generation 1
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
    17

The best configuration is:
     0     0     0     0     0     0     0     0     0     0     0     0

The highest L/D ratio is:
  140.3684

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
78058.master.cluster
78059.master.cluster
78060.master.cluster
78061.master.cluster
78062.master.cluster
```

```
78063.master.cluster
78064.master.cluster
78065.master.cluster
78066.master.cluster
78067.master.cluster
78068.master.cluster
78069.master.cluster
78070.master.cluster
78071.master.cluster
78072.master.cluster
78073.master.cluster
78074.master.cluster
78075.master.cluster
78076.master.cluster
78077.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20



Generation 10
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     2

The best configuration is:
     0     0     0     0     0     0     0     0     0     0     0     0

The highest L/D ratio is:
  140.3684

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
```

```
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
78280.master.cluster
78281.master.cluster
78282.master.cluster
78283.master.cluster
78284.master.cluster
78285.master.cluster
78286.master.cluster
78287.master.cluster
78288.master.cluster
78289.master.cluster
78290.master.cluster
78291.master.cluster
78292.master.cluster
78293.master.cluster
78294.master.cluster
78295.master.cluster
78296.master.cluster
78297.master.cluster
78298.master.cluster
78299.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20

Warning: No display specified.  You will not be able to display graphics on the
screen.
Warning: No window system found.  Java option 'MWT' ignored

                          < M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
               Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                            February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >>   Get fitness for new nests
>>    01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
>>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
Generation 1
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     1
```

```
The best configuration is:
     0     0     0     0     0     0     0     0     0     0     0     0

The highest L/D ratio is:
  140.3684

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
80442.master.cluster
80443.master.cluster
80444.master.cluster
80445.master.cluster
80446.master.cluster
80447.master.cluster
80448.master.cluster
80449.master.cluster
80450.master.cluster
80451.master.cluster
80452.master.cluster
80453.master.cluster
80454.master.cluster
80455.master.cluster
80456.master.cluster
80457.master.cluster
80458.master.cluster
80459.master.cluster
80460.master.cluster
```

```
80461.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20




Generation 20
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     1

The best configuration is:
   1.0e-04 *

  Columns 1 through 7

         0         0         0         0         0         0    0.0555

  Columns 8 through 12

   -0.1121   -0.1587    0.1633   -0.5229   -0.8267

The highest L/D ratio is:
  169.4639

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
```

```
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
80830.master.cluster
80831.master.cluster
80832.master.cluster
80833.master.cluster
80834.master.cluster
80835.master.cluster
80836.master.cluster
80837.master.cluster
80838.master.cluster
80839.master.cluster
80840.master.cluster
80841.master.cluster
80842.master.cluster
80843.master.cluster
80844.master.cluster
80845.master.cluster
80846.master.cluster
80847.master.cluster
80848.master.cluster
80849.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20

Warning: No display specified.  You will not be able to display graphics on the
screen.
Warning: No window system found.  Java option 'MWT' ignored

                          < M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
               Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                             February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >>   Get fitness for new nests
>>    01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
>>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
Generation 1
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
    12

The best configuration is:
     0     0     0     0     0     0     0     0     0     0     0     0
```

```
The highest L/D ratio is:
  169.4630

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
80871.master.cluster
80872.master.cluster
80873.master.cluster
80874.master.cluster
80875.master.cluster
80876.master.cluster
80877.master.cluster
80878.master.cluster
80879.master.cluster
80880.master.cluster
80881.master.cluster
80882.master.cluster
80883.master.cluster
80884.master.cluster
80885.master.cluster
80886.master.cluster
80887.master.cluster
80888.master.cluster
80889.master.cluster
80890.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
```

```
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20


Generation 20
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     1

The best configuration is:
   1.0e-04 *

  Columns 1 through 7

         0         0         0         0         0         0   -0.1742

  Columns 8 through 12

    0.3068    0.2989   -0.1460   -0.1371    0.9543

The highest L/D ratio is:
  199.4584

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
```

```
Preprocessing case 20 ...
Finished preprocessing
81287.master.cluster
81288.master.cluster
81289.master.cluster
81290.master.cluster
81291.master.cluster
81292.master.cluster
81293.master.cluster
81294.master.cluster
81295.master.cluster
81296.master.cluster
81297.master.cluster
81298.master.cluster
81299.master.cluster
81300.master.cluster
81301.master.cluster
81302.master.cluster
81303.master.cluster
81304.master.cluster
81305.master.cluster
81306.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20

Warning: No display specified.  You will not be able to display graphics on the
screen.
Warning: No window system found.  Java option 'MWT' ignored

                          < M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
               Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                             February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >>   Get fitness for new nests
>>    01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
>>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
Generation 1
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     8

The best configuration is:
     0     0     0     0     0     0     0     0     0     0     0     0

The highest L/D ratio is:
  193.9533

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
```

```
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
81368.master.cluster
81369.master.cluster
81370.master.cluster
81371.master.cluster
81372.master.cluster
81373.master.cluster
81374.master.cluster
81375.master.cluster
81376.master.cluster
81377.master.cluster
81378.master.cluster
81379.master.cluster
81380.master.cluster
81381.master.cluster
81382.master.cluster
81383.master.cluster
81384.master.cluster
81385.master.cluster
81386.master.cluster
81387.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20

Generation 20
  Sorting nests by fitness
```

```
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
    18

The best configuration is:
   1.0e-05 *

  Columns 1 through 7

         0         0         0         0         0         0    0.0422

  Columns 8 through 12

    0.3470    0.0640    0.0629    0.0216    0.0012

The highest L/D ratio is:
  204.1399

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
81847.master.cluster
81848.master.cluster
81849.master.cluster
81850.master.cluster
81851.master.cluster
81852.master.cluster
```

```
81853.master.cluster
81854.master.cluster
81855.master.cluster
81856.master.cluster
81857.master.cluster
81858.master.cluster
81859.master.cluster
81860.master.cluster
81861.master.cluster
81862.master.cluster
81863.master.cluster
81864.master.cluster
81865.master.cluster
81866.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20




Warning: No display specified.  You will not be able to display graphics on the
screen.
Warning: No window system found.  Java option 'MWT' ignored

                          < M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
               Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                             February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >>   Get fitness for new nests
>>    01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
>>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
Generation 1
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     2

The best configuration is:
     0     0     0     0     0     0     0     0     0     0     0     0

The highest L/D ratio is:
  204.1379

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
```

```
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
81914.master.cluster
81915.master.cluster
81916.master.cluster
81917.master.cluster
81918.master.cluster
81919.master.cluster
81920.master.cluster
81921.master.cluster
81922.master.cluster
81923.master.cluster
81924.master.cluster
81925.master.cluster
81926.master.cluster
81927.master.cluster
81928.master.cluster
81929.master.cluster
81930.master.cluster
81931.master.cluster
81932.master.cluster
81933.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20


Generation 20
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
```

```
Finish cuckoo search
The best case is case:
     1

The best configuration is:
   1.0e-03 *

  Columns 1 through 7

         0         0         0         0         0         0   -0.0040

  Columns 8 through 12

   -0.1713    0.0017    0.0038    0.0021    0.0140

The highest L/D ratio is:
  205.1709

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
82319.master.cluster
82320.master.cluster
82321.master.cluster
82322.master.cluster
82323.master.cluster
82324.master.cluster
82325.master.cluster
82326.master.cluster
```

```
82327.master.cluster
82328.master.cluster
82329.master.cluster
82330.master.cluster
82331.master.cluster
82332.master.cluster
82333.master.cluster
82334.master.cluster
82335.master.cluster
82336.master.cluster
82337.master.cluster
82338.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20

Warning: No display specified.  You will not be able to display graphics on the
screen.
Warning: No window system found.  Java option 'MWT' ignored

                          < M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
               Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                             February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
>> >> >> >> >>   Get fitness for new nests
>>    01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
>>
>> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >> >>
Generation 1
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
    18

The best configuration is:
     0     0     0     0     0     0     0     0     0     0     0     0

The highest L/D ratio is:
  205.1772

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
```

```
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
82613.master.cluster
82614.master.cluster
82615.master.cluster
82616.master.cluster
82617.master.cluster
82618.master.cluster
82619.master.cluster
82620.master.cluster
82621.master.cluster
82622.master.cluster
82623.master.cluster
82624.master.cluster
82625.master.cluster
82626.master.cluster
82627.master.cluster
82628.master.cluster
82629.master.cluster
82630.master.cluster
82631.master.cluster
82632.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20



Generation 20
  Sorting nests by fitness
  Treating discarded cuckoos ...
  Treating top cuckoos ...
Finish cuckoo search
The best case is case:
     1

The best configuration is:
     0     0     0     0     0     0     0     0     0     0     0     0
```

```
The highest L/D ratio is:
  205.1772

Writing mesh 1/20
Writing mesh 2/20
Writing mesh 3/20
Writing mesh 4/20
Writing mesh 5/20
Writing mesh 6/20
Writing mesh 7/20
Writing mesh 8/20
Writing mesh 9/20
Writing mesh 10/20
Writing mesh 11/20
Writing mesh 12/20
Writing mesh 13/20
Writing mesh 14/20
Writing mesh 15/20
Writing mesh 16/20
Writing mesh 17/20
Writing mesh 18/20
Writing mesh 19/20
Writing mesh 20/20
Finish writing meshes
Preprocessing case 1 ...
Preprocessing case 2 ...
Preprocessing case 3 ...
Preprocessing case 4 ...
Preprocessing case 5 ...
Preprocessing case 6 ...
Preprocessing case 7 ...
Preprocessing case 8 ...
Preprocessing case 9 ...
Preprocessing case 10 ...
Preprocessing case 11 ...
Preprocessing case 12 ...
Preprocessing case 13 ...
Preprocessing case 14 ...
Preprocessing case 15 ...
Preprocessing case 16 ...
Preprocessing case 17 ...
Preprocessing case 18 ...
Preprocessing case 19 ...
Preprocessing case 20 ...
Finished preprocessing
82733.master.cluster
82734.master.cluster
82735.master.cluster
82736.master.cluster
82737.master.cluster
82738.master.cluster
82739.master.cluster
82740.master.cluster
82741.master.cluster
82742.master.cluster
82743.master.cluster
82744.master.cluster
82745.master.cluster
82746.master.cluster
82747.master.cluster
82748.master.cluster
82749.master.cluster
82750.master.cluster
82751.master.cluster
82752.master.cluster
Solving submitted to the cluster
Job Submitted, just wait
```

```
New result files obtained :)
  Get fitness for new nests
   01    02    03    04    05    06    07    08    09    10
   11    12    13    14    15    16    17    18    19    20
```