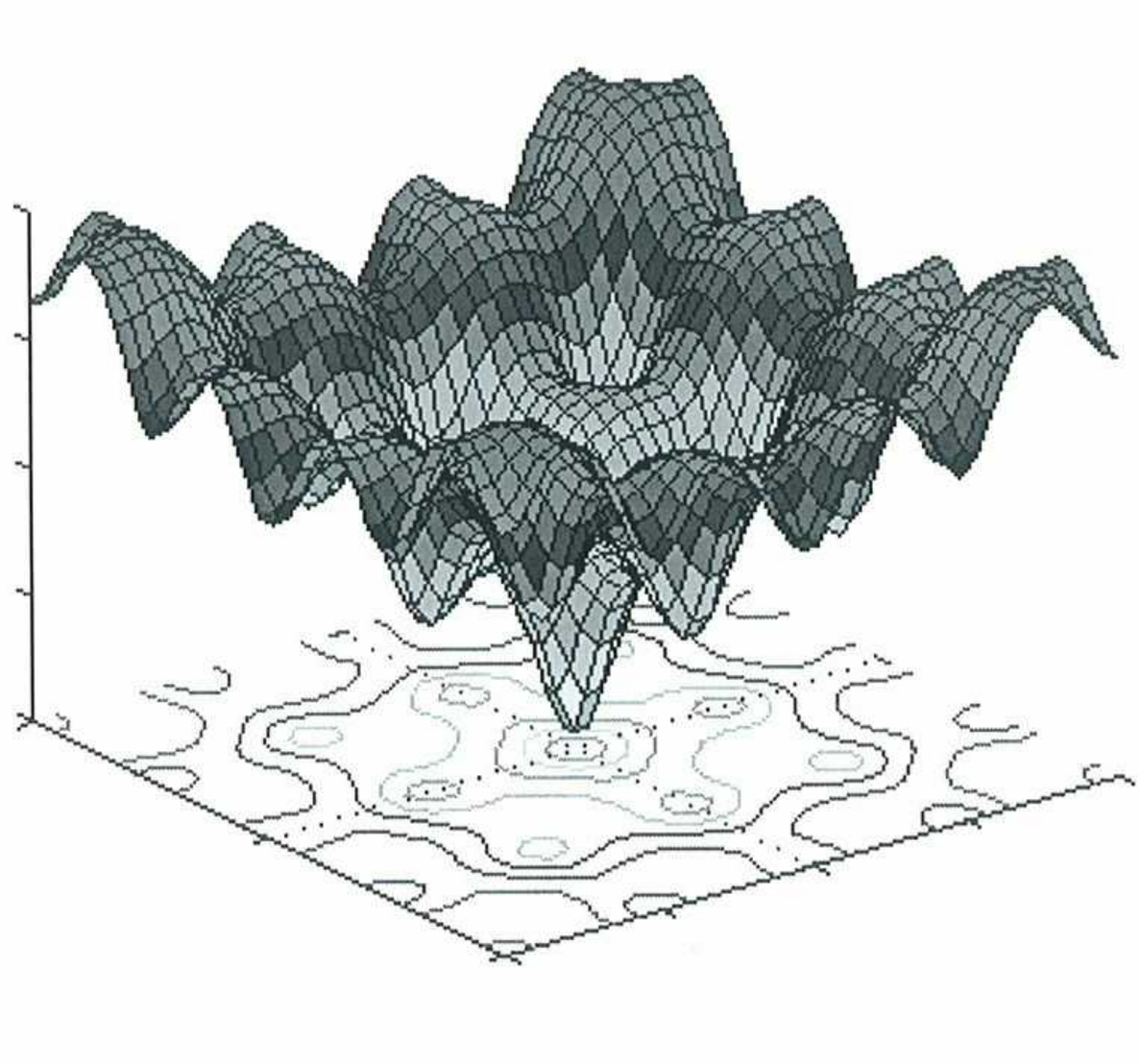
Evolutionary Methods for Optimal Shape Design

A. Fung
E. Pahl
G. Bugeda
E. Oñate



Publication CIMNE Nº-270, Juny 2005

Evolutionary Methods for Optimal Shape Design

A. Fung
Departament of Aeronautics
Imperial College of Science, Technology and Medicine, UK

E. Pahl, G. Bugeda and E. Oñate
International Center for Numerical Methods in Engineering
Gran Capitán s/n, 08034 Barcelona, Spain

Publication CIMNE Nº-267, May 2005

International Center for Numerical Methods in Engineering
Gran Capitán s/n, 08034 Barcelona, Spain

SUMMARY

An optimisation code called Optima has been developed at the International Centre for Numerical Methods in Engineering (CIMNE). This report validates two reputably robust Evolutionary Algorithms available in Optima and employs them on aerodynamic shape optimisation problems. The two schemes, Differential Evolution Scheme 1 (DE1) and Evolution Strategy coupled with Covariance Matrix Adaptation (ES-CMA), were tested and verified on three standard parametric optimisation objective functions through comparison against existing results. Analysis of the test data allowed trends to be established and from this, settings to enhance the performance of the algorithms were proposed and substantiated.

The algorithms and their suggested settings were applied on an inverse and a direct constrained shape optimisation problem involving NACA four digit aerofoils. The inverse task involves the recovery of an aerofoil profile through its pressure distribution and the direct problem entails the minimisation of an objective function containing a combination of lift and drag coefficients. Simulation of the aerofoil in turbulent flow was done using a Computational Fluid Dynamics (CFD) setup.

Finally, the results are presented and the ES-CMA method, using the settings CMA-B proposed in this paper, was found to be most robust. Suggestions for improvements and further work in the optimisation configuration and the problem simulation are also proposed.

ACKNOWLEDGEMENTS

I would first of all, like to thank Dr. Peiro from Imperial College and Dr. Oñate of CIMNE for making this exchange possible. I am also very grateful to the support from the personnel at CIMNE including Dr. Bugeda, the GiD and Tdyn teams and the administration staff for continuously resolving my queries. I would also like to express my appreciation to the hard working people in room C5, who introduced me to the lighter side of Barcelona, through the hill climbs and pre-race pasta dinners. Finally, and most importantly, I would like to express my sincere gratitude to Elke Pahl, without whom this project would not have been possible. The volume and quality of work in this report is a testimony to her endless support, guidance and friendship throughout the four months.

CONTENTS

1	**Introduction**	**1**
	1.1 Evolutionary Algorithms, the Stochastic Approach	1
	1.2 Aims & Objectives	1
	1.3 Report Content	2
2	**Basic Theory**	**3**
	2.1 General Optimisation Problem	3
	2.2 Evolutionary Algorithms	3
	2.2.1 Parameters	4
	2.2.1.1 Representation	4
	2.2.1.2 Population	4
	2.2.1.3 Evaluation/ Objective Function	4
	2.2.1.4 Constraints	5
	2.2.1.5 Termination Criteria	5
	2.2.2 Operators	5
	2.2.2.1 Selection	5
	2.2.2.2 Recombination	5
	2.2.2.3 Mutation	6
	2.2.2.4 Adaptation	6
	2.3 Evolution Strategies	7
	2.4 Covariance Matrix Adaptation	7
	2.5 Differential Evolution Scheme 1 (DE1)	11
	2.6 Optima	13
	2.7 XFOIL	14
	2.8 Pre/Post-processor GiD	14
	2.9 Tdyn – FEM Fluid Dynamics Solver	15
	2.9.1 Test Results and Overall Choice of Model	15
	2.9.2 Tdyn Limitations	18
3	**Problem Generation**	**20**
	3.1 Geometry Definition	20
	3.2 Problem Geometry Realisation	20
	3.3 Geometry Constraints	21
	3.4 Simulation and Evaluation of Objective Function	22
	3.4.1 Problem 1 – Inverse Problem Objective Function	23
	3.4.2 Problem 2 – Direct Optimisation Objective Function	23
4	**Validation of Optima**	**24**
	4.1 Standard Parametric Optimisation Objective Functions	24
	4.1.1 First De Jong Function	24
	4.1.2 Second De Jong Function	24
	4.1.3 Zimmermann Function	25
	4.2 Optima DE1	25
	4.2.1 Optima DE1 Validation	25
	4.2.2 Optima Best Settings	26
	4.3 CMA	27
	4.3.1 CMA Comparison	27
	4.3.2 CMA Best Settings	27

4.3.3 CMA Settings Suggestions 28
4.3.3.1 CMA-A 28
4.3.3.2 CMA-B 29
4.3.3.3 CMA-A&B Validation 29

5 Inverse and Direct Optimisation **30**
5.1 Problem 1 – Inverse Optimisation Results 30
5.2 Problem 2 – Direct Optimisation Results 32

6 Evaluation & Conclusions **35**
6.1 Standard Parametric Optimisation Objective Functions 35
6.2 Problem 1 36
6.3 Problem 2 36
6.4 Overall Summary of Optimisation Test Results 38
6.5 Model Simulation 39
6.6 Further Work & Suggestions 39

References **40**

LIST OF FIGURES & TABLES

2.2.1 Solution of general optimisation problem by an Evolutionary Algorithm 4
2.2.2 Gaussian mutation distribution 6
2.4.1 The evolution of the mutation distribution in the 2D Rosenbrock function optimization using CMA 10
2.5.1 Generation of trial vector $\underline{v}$ using DE1 11
2.5.2 DE1 offspring generation 12
2.5.3 Original crossover length probability 12
2.5.4 Optima DE1 crossover process 13
2.6.1 Optima interface 13
2.6.2 Optima operators and parameters 14
2.8.1 Gid & Tdyn solution process 15
2.9.1 Meshed Geometry 16
2.9.2 Close-up of meshed leading edge 16
2.9.3 Tdyn turbulence models comparison 17
2.9.4 Eddy Viscosity Distribution for Smagorinski Model 18
2.9.5 Eddy Viscosity Distribution for Kε Lam Bremhorst Model 18
2.9.6 Eddy Viscosity Distribution for k-ω SST Model 18
2.9.7 Comparison of NACA0012 Cp Distributions for Tdyn k-ω SST and XFOIL 19
3.1.1 Aerofoil and general notation 20
3.2.1 CV & aerofoil 21
3.2.2 Material and fluid boundary assignment 21
3.2.3 Boundary Conditions 21
3.2.4 Meshed View 21
3.4.1 Optimisation process 22
4.1.1 First De Jong function 24
4.1.2 Second De Jong function 24
4.1.3 Zimmermann visualization with plan view 25
4.2.1 Optima DE1 results 25
4.2.2 DE1 best settings 26
4.3.1 CMA Comparison with DE1 using 3 Test Functions 27
4.3.2 Suggested Strategy Variable Settings by Authors 27
4.3.3 CMA best settings no. of times converged out of 10 28
4.3.4 CMA best settings nfe to converge or meet halting criterion 28
4.3.5 CMA-A settings 29
4.3.6 CMA-B settings 29
4.3.7 CMA-A&B no. of times converged out of ten 29
4.3.8 CMA-A&B nfe to converge or meet halting criterion 29
5.1 Work matrix 30
5.1.1 Comparison between target and results at 59 generations for Problem 1 30
5.1.2 Normalised best individual fitness for all 6 schemes for Problem 1 30
5.1.3 Aerofoil Profiles at generation 59 for Problem 1 31
5.1.4 Pressure distributions at generation 59 for Problem 1 31
5.2.1 Comparison of results from different algorithms tested on Problem 2 at Generation 55 32
5.2.2 Normalised best individual fitness for 6 schemes for Problem 2 32
5.2.3 Aerofoil profiles at generation 59 for DE1 tests on Problem 2 33
5.2.4 Aerofoil profiles at generation 59 for CMA tests on Problem 2 33

5.2.5	Pressure distributions at generation 55 for DE1 tests on Problem 2	34
5.2.6	Pressure distributions at generation 55 for CMA tests on Problem 2	34
6.1	No. of times converged out of ten	35
6.2	Nfe to converge or meet halting criterion	35
6.3.1	Problem 2, Best DE1 and ES-CMA results produce similar profile	37
6.6.1	3D model of a wing with fluid conditions assigned	39

1 Introduction

The need for shape optimisation is a major concern transcending many engineering fields, ranging from finding the optimum blade shape for a wind turbine to the optimum inlet nozzle shape to a die in a casting process. Engineers strive not only to enhance a single property in the design process, but aim to find a global optimum.

Traditionally, such optimisation processes might stem from trial and error and engineering experience. However, these inefficient and ambiguous methods have given way to the more reliable influences of numerical methods. These simultaneously analyse and optimise possible solutions in multi-variable problems and thus make optimal design an automatic and a more reliable process.

A program called Optima, written by Elke Pahl and under development at the International Centre for Numerical Methods in Engineering (CIMNE) will be used for such optimisation processes and utilises Evolutionary Algorithms to this end. Optima can be considered as an optimisation tool where schemes with different operators and parameters can be experimented on with different problems. Particular schemes with a reputation for robustness in the shape optimisation field have been incorporated into Optima and will be tested on specific examples in this report.

1.1 Evolutionary Algorithms, the Stochastic Approach

Optimisation methods can be classified into two classes; Deterministic Methods and Stochastic Methods. The former produce the same results when started at the same point(s) of the search space, whereas the latter integrate randomness into the solution process so that in general, no two runs produce the same results. Stochastic algorithms have been proven effective, especially when conducting direct search global optimisation, where the general location of the minimum is not known. The sampling of random individuals across the entire feasible search space allows them to escape local optima and permits them to search a larger domain relative to deterministic methods.

A distinctive sub-set of Stochastic Methods is the Evolutionary Algorithm (EA). This is a term describing all algorithms employing numerical models based on evolutionary mechanisms and are founded on Darwin's Theory of Evolution which bases itself on survival of the fittest. EAs use the fundamental building blocks of evolution, namely recombination, mutation and selection, to develop the optimisation process. Some EAs utilise recombination, known as mating in biological terms, and although it is not a necessary requirement, most EAs employ mutation. This class of algorithm does not require gradient information of functions, is good with ´noisy´ problems, is relatively costly for very simple problems (due to their scattered search approach), but can be very effective when applied to more complex optimisation problems, as it will be shown in this report.

1.2 Aims & Objectives

The aim of this project is to validate and apply to a shape optimisation problem two types of Evolutionary Algorithms in Optima with the use of a Computational Fluid Dynamics solver for problem simulation. Three programs will be used in total for the optimisation process, namely GiD, Tdyn and Optima, and all have been developed or are under development at CIMNE. A fourth program, XFOIL, will be used to evaluate the accuracy of the GiD and Tdyn problem simulation setup. Validation and testing of Optima comes in the form of three standard parametric optimisation test functions, an inverse aerofoil optimisation problem and a direct aerofoil optimisation problem. The two Evolutionary Algorithms to be tested are the Differential Evolutionary 1 (DE1) and the Evolution Strategy combined with Covariance Matrix

Adaptation (ES-CMA) scheme. An attempt is also made to investigate the behaviour and performance of these methods on the three test functions in order to suggest settings to enhance their performance.

1.3 Report Content

Chapter 2: Basic Theory
The general optimisation problem is presented with background information on Evolutionary Algorithms, in particular DE1 and ES-CMA. Details of the optimisation program Optima, the comparison resource XFOIL and the geometry generator GiD are introduced. In depth theory, problem simulation and testing for the CFD program Tdyn are also summarised here.

Chapter 3: Problem Generation
The constrained aerofoil geometry is defined before a full test model is realized in GiD and Tdyn. The creation and setup of the optimisation loop to be used is described. This is followed by the definition of the inverse and direct aerofoil optimisation experiments to be tested in Chapter 5, referred to as Problems 1 and 2, respectively.

Chapter 4: Validation of Optima
Testing of DE1 and ES-CMA on three parametric optimisation objective functions is carried out. The DE1 results are compared with [13] and contrasted with CMA´s performance. The two DE1 parameters are varied to establish trends from which suggestions for enhanced efficiency settings can be made. Three sets of CMA suggested parameter values from the authors are tested, and again, from these results, suggestions are made for improved performance settings.

Chapter 5: Inverse & Direct Optimisation
The results from the constrained inverse and direct optimisation tests, Problems 1 and 2, are presented. Problem 1 involves the recovery of an aerofoil shape using pressure distributions and Problem 2 aims to minimise an objective function containing a combination of lift and drag coefficients.

Chapter 6: Evaluation & Conclusions
The results from Chapters 4 and 5 for the two Evolutionary Algorithms at different settings are compared and their performance examined with their formulation in mind. Explanations and postulations as to their success and failure on different problems are discussed. Finally, suggestions for further work are presented.

2 Basic Theory

2.1 General Optimisation Problem

We can describe the general optimisation problem as follows:

$$\min f(x)$$

$$\text{subject to } x \in S$$

x is the design variable vector, f is the objective function and S is the feasible design search space defined by the constraints:

$$g_i(x) = 0 \qquad i = 1,\ldots,m \qquad \text{equality constraints}$$
$$h_i(x) \leq 0 \qquad j = 1,\ldots,n \qquad \text{inequality constraints}$$

The feasible search space S contains a population number NP of individuals or design parameters x and a n dimensional or n degrees of freedom problem implies that each individual will contain n design parameters. Simply put, the goal of the global optimisation procedure is to find the individual whose parameters, when evaluated against the objective function, produce the globally optimal value.

Throughout this report, the optimisation problem will be described in terms of the minimisation of the objective function since any maximisation problem

$$\max f(x)$$

may also be written as a minimisation problem

$$\min -f(x)$$

and any inequality constraint

$$g(x) \leq 0$$

may be written as

$$-g(x) \geq 0\,.$$

2.2 Evolutionary Algorithms [1,2,3,4,5]

Evolutionary Algorithms (EA) operate on a population of potential solutions called individuals and apply the Darwinian principle of survival of the fittest. This is done in order to produce individuals in succeeding generations with improved fitness in accordance to the environment or problem domain and with the ultimate aim of finding an individual of global minimal fitness.

The basic structure of a single population evolutionary algorithm solution process is shown in Figure 2.2.1. The alogorithm is controlled by parameters contained in the red boxes and apply the operators shown in the yellow box to evolve the optimisation process.

After an initial population is chosen, the fitness of each individual in the feasible search space is evaluated and a new generation is formed through the evolutionary mechanisms described in Section 2.2.2 as operators. This process is repeated until we find the individual with minimal fitness.

Optima permits the easy tuning of the parameters and operators that make up an EA so that it may be in theory possible to generate a scheme more suited to the problem to be solved. Apart from being able to adjust all of the variables mentioned in Sections 2.2.1 and 2.2.2, two EAs have also been incorporated into Optima. The first EA involves Covariance Matrix Adaptation (CMA) whose methodology is explained in Section 2.3. The second has been developed by Rainer Storn and Kenneth Price, called the Differential Evolution Scheme 1 (or DE1) and will be introduced in more detail in section 2.5.

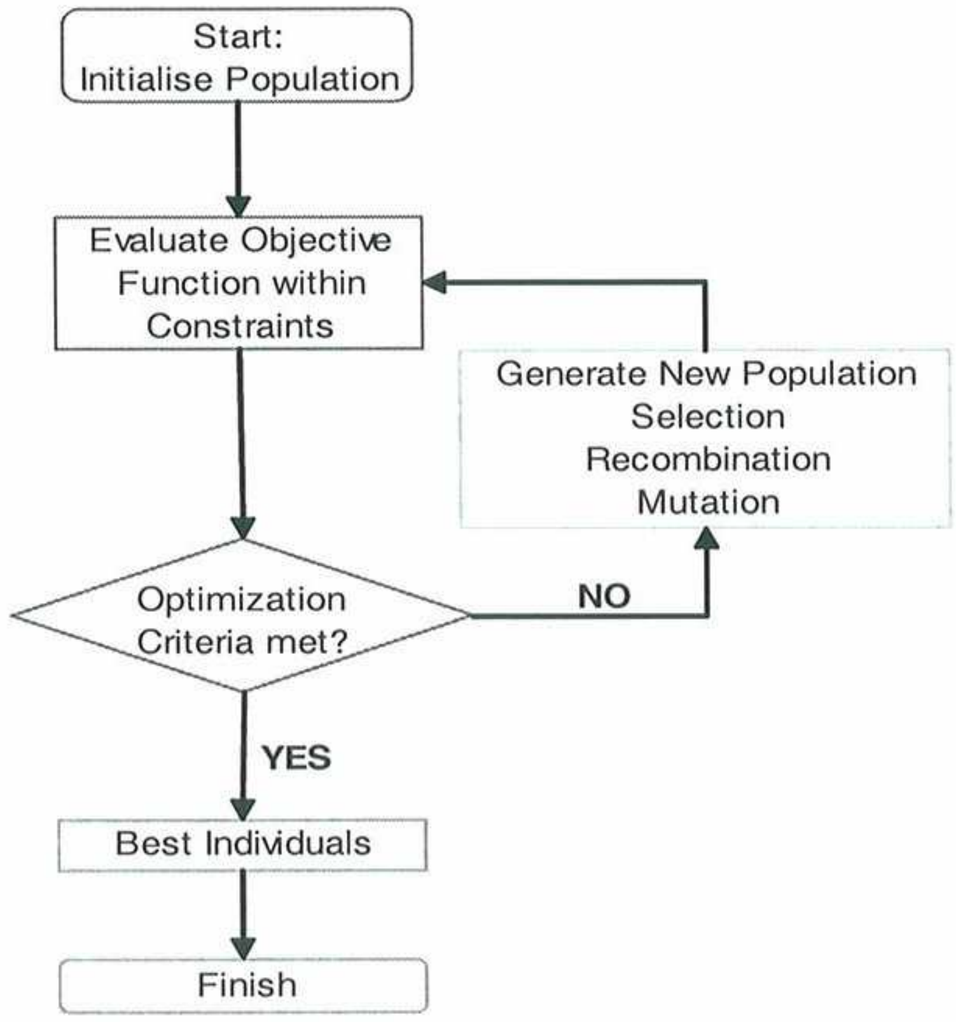


Figure 2.2.1 Solution of general optimisation problem by an Evolutionary Algorithm

2.2.1 Parameters:

2.2.1.1 Representation – It is important to represent an optimisation problem correctly when using EAs so that the continuous variation of parameters can sufficiently mirror the genetic variation and modelling that takes place during computation. Representation is the first consideration to be made before the application of the algorithm. Suggestions such as the use of n-dimensional floating point vectors for representing the variables, where n is the number of parameters to be optimised, should be considered carefully at this stage.

2.2.1.2 Population – Selecting a small population may restrict the possibilities for recombination (reproduction in biological terms) and also limit the search domain so that the global optimum may be less likely to be found. However, the use of a large population may prove costly from conducting the more expansive search. When constructing the initial population, it is also important to incorporate any known information about the search space, such as the presence of local minima so as to avoid the clustering of individuals during computation.

2.2.1.3 Evaluation/ Objective Function – This is the direct link between the EA and the optimisation problem. The evaluation function assesses the quality of generated solutions and assigns a fitness to the individuals accordingly. Since the calculation or problem simulation process can be the most time consuming part of the calculation, it is important to formulate the evaluation function in a suitable way with calculation cost in mind.

2.2.1.4 Constraints – The optimisation problems to be solved in this report involve multiple constraints which define the feasible search space. The algorithm to be used must also be able to treat non-feasible solutions. Without prior knowledge of where the global optimum lies, one must assume that it might be located near an infeasible region. It is therefore necessary to carefully consider the strategy to deal with infeasible solutions before simply discarding them since this could lead to loss of information around potential global optima.

The possible alternatives to simple elimination of infeasible solutions in the calculation process include "repairing" solutions or keeping the solutions and assigning them an appropriate fitness. Some common fitness regimes include:

i. Introduction of a penalty function – A penalty function is assigned to the unfeasible solution in proportion to how far it moves into the infeasible region. This penalty function is then directly subtracted from the individual's fitness. This method is used in Problem 2.

ii. Barrier methods – Unfeasible solutions are assigned fitnesses of $\pm\infty$ to decrease the probability of selecting them. This is the method employed in Problem 1.

2.2.1.5 Termination Criterion – The termination criterion determines the running time of the algorithm. The criterion can either simply be finding certain number of feasible solutions or letting the algorithm run for a set number of steps.

2.2.2 Operators:

2.2.2.1 Selection – Selection occurs at two stages in the solution process. Firstly we have Mating Selection in which individuals from the population are chosen by one of many techniques and copied to a mating pool to produce offspring. Environmental Selection then decides which of the newly created offspring and which of the individuals from the previous generations will together form the new generation. The techniques below are applicable to both Mating Selection and Environmental Selection and are all available in Optima:

i. Best Individuals – choose the n best individuals and randomly choose m individuals so that $n+m=p$ where p is the size of the mating pool (for Mating Selection) or the new generation (for Environmental Selection).

ii. Binary Tournament – Randomly choose two individuals from the population and select the fitter individual. Continue with this process until p individuals have been selected.

iii. Modified Binary Tournament – Randomly select two individuals from the population and discard the solution with the worse fitness. Continue with this process until p individuals remain in the mating pool or in the population. The advantage of the modified method is that it guarantees the best solution is retained in the population even if it is not selected.

iv. Roulette Wheel – Each individual is mapped to adjoining segments of a line whose length is proportional to the individual's fitness. A random number is generated and the individual whose segment spans this number is selected. The selection process is repeated until the required number of individuals has been chosen.

2.2.2.2 Recombination – The two most common methods used for recombination are the Crossover Method and the Weighted Average Method:

i. Crossover Method – In this method, each offspring is produced from a combination of information from two or more parents. In turn, each parent may contribute varying numbers of genes to their offspring by segmentation of their total number of genes at crossover points which may vary from one to an *(n-1)* number of crossover points.

For multi-point crossover, the positions are either randomly chosen or user specified with no duplicates and then sorted into ascending order. The variables between successive crossover points are then exchanged between the parents to produce the new offspring.

The motivation for this method is to try to capture the genes or variables that give the best performance for a given problem and these may not lie adjacent to each other. This strategy also promotes the exploration of the search space, since the crossover process disturbs the convergence to fitter individuals early on in the search.

ii. Weighted Average Method – This combines information from two or more parents whose contributions to the offspring are adjustably weighted and tends to have a blending effect on the population.

2.2.2.3 Mutation – This changes all or some parts of an individual in order to form a new one. Each parameter to be mutated in x_i undergoes mutation by a small change Δx_i to become x_i', so $x_i' = x_i + \Delta x_i$.

i Gaussian Mutation – The mutation follows the Gaussian distribution from which it takes its name and is identical to the Normal Distribution:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \qquad \textit{Equation 2.2.1}$$

Here, σ and μ are the standard deviation and mean, respectively. We use the above distribution with a mean value of zero and the standard deviation set to the mutation step size which in essence gives $\Delta x_i \sim N(o, \sigma)$.

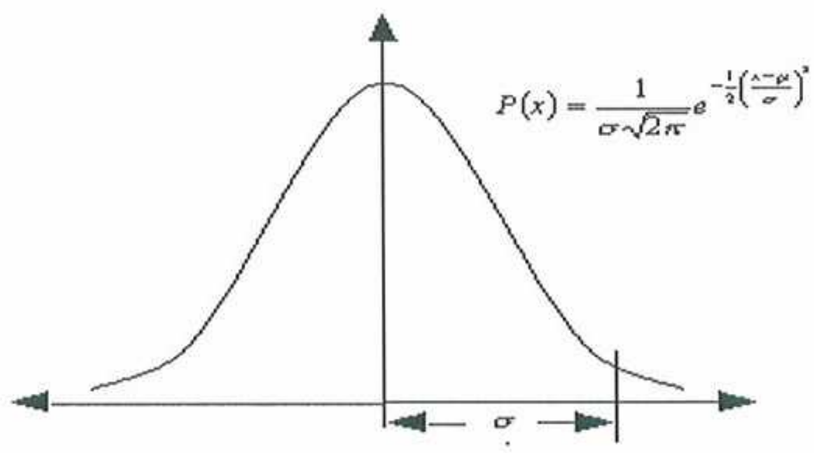


Figure 2.2.2 Gaussian mutation distribution

2.2.2.4 Adaptation – The mutation step, in the process of generating new individuals, can be adapted to suit the optimisation problem by changing σ. Mutation occurs either before or after recombination and the mutation step adaptation can be deterministic, where a predetermined set procedure is established to change the mutation step size during calculation. However, the more effective approach is to use self-adaptation, which implies that the mutation step size evolves with the solution so that the applied mutation step becomes tailored to the situation. Below is a global self-adaptive method which applies the same adaptation equally to all of the problem parameters:

i. 1/5 Success Rule [2] – This was developed by Rechenberg [6] for the (1+1) Evolution Strategy (see Section 2.3 for more details). For a population of *NP* individuals, the rule takes into account the ratio, p_s, of successful mutations to the total number of mutations in the previous t generations ($txNP$). The value of the ratio is then compared with a constant, γ, which is usually set to 1/5, and hence the name, so that if it is less than 1/5, the step size is decreased and if more than 1/5, the step size is increased as follows:

$$\sigma^{(g)} = \begin{cases} \sigma^{(g-n)}/c & \text{if} \quad p_s > \gamma \\ \sigma^{(g-n)}.c & \quad\quad\; p_s < \gamma \\ \sigma^{(g-n)} & \quad\quad\; p_s = \gamma \end{cases}$$

where $0.817 \leq c \leq 1$ and usually $c=0.85$ is set. A c value of 1.0 corresponds to no adaptation and the mutation step size is held constant. p_s is the success probability given by the relative frequency of successful mutations measured over some t generations to be set by the user. γ is the constant usually set to 1/5 but should be set lower for multiple optima problems. Since only one step size can be adapted with this method, all parameters in an individual are subjected to the same step size, otherwise known as global step size mutation.

This method was tested extensively and showed low convergence rates due to its global approach. For the aerofoil problem, which contains 3 parameters of different orders of magnitude, this adaptation technique is also not expected to perform well so that no further testing will be done using this technique on the problems presented below.

2.3 Evolution Strategies

Evolution Strategies (ES) were first developed in the 1960´s by Rechenberg and Schwefel at the Technical University of Berlin in support of wind tunnel experiments for aerodynamic shape optimisation purposes. ES are essentially a type of Evolutionary Algorithm with three main characteristics:

i. ES use a real-coding representation of the design parameters.

ii. ES depend on mutation, deterministic selection (and some types of ES use recombination, with recombination usually occurring before mutation) for their evolution.

iii. The ES used in this report use self-adaptation of mutation parameters.

Crossover and weighted average methods (see Section 2.2.2.2) are used for recombination and the mutation method used in this report comes in the form of Gaussian mutation (see Section 2.2.2.3). The self-adaptation mechanism to be used for the Gaussian mutation step size will be the Covariance Matrix Adaptation method (CMA) outlined in Section 2.4. The many different types of ESs can be categorised by:

(μ,λ)	Parents die off since they are not taken into account during selection.
$(\mu+\lambda)$	Parent generation taken into account during selection.
$(\mu/\rho,\lambda)$	ρ out of μ parents are recombined, parents die off.
$(\mu/\rho+\lambda)$	ρ out of μ parents are recombined and all parents are considered for selection.

Here, λ is the number of offspring generated per generation and μ is the population size.

2.4 Covariance Matrix Adaptation [2, 7, 8, 9, 10, 11]

Covariance Matrix Adaptation (CMA) is an individual parameter mutation step adaptation method designed to be very efficient when used in combination with an Evolution Strategy mentioned in Section 2.3. The formulation below is for (μ_I,λ) ES-CMA but the theory applies equally to other ES types.

The implementation of CMA involves two types of variables: An Object variable and Strategy variables. The Object variable is simply the individual to be optimised represented here by the vector x and the Strategy variables consist of the Covariance matrix, the global step size and other parameters affecting the mutation step size such as the random vector z. If we have a n-dimensional problem, implying an individual contains n parameters, each of these parameters is subject to mutation. The mutation step is a function of both a global step size σ and the covariance matrix C. In the initial stage, the global step size and the covariance matrix (this is initialized as the identity matrix) are the same for all of the parameters. The information about the selected mutation steps during the development in previous stages of the calculation is collected in parameters known as the evolution paths, s_c and s_σ which provide the feedback necessary to self-adapt C and σ, resecptively.

ES carries out the mutation by adding an arbitrarily normally distributed mutation step with zero mean. To do this an n-dimensional random vector $z^{(g+1)}$ is linearly transformed (where $z \sim N(0,I)$, I being the identity matrix) by the matrices $B^{(g)}$ and $D^{(g)}$. We have $B^{(g)}$ as an orthogonal matrix for generation g whose columns are the normalised eigenvectors of the covariance matrix and $D^{(g)}$ as a diagonal matrix whose elements are the square roots of the eigenvalues of $C^{(g)}$. [1] $B^{(g)}D^{(g)}z^{(g+1)}$ will then be normally distributed according to $N(0,C)$ and they are multiplied by the global step size to give the mutation step. The variables that are actually adapted by CMA using data from the evolution paths are σ, B and D. The mutation to produce a λ number of offspring for the object variable vector x is given below:

$$x_k^{(g+1)} = \langle x \rangle_\mu^{(g)} + \sigma^{(g)} \underbrace{B^{(g)} D^{(g)} z_k^{(g+1)}}_{\sim N(0,C^{(g)})} \qquad \text{Equation 2.4.1}$$

where $\langle x \rangle_\mu^{(g)} = \frac{1}{\mu} \sum_{i \in I_{sel}^{(g+1)}} x_i^{(g+1)}$ is the centre of mass of the selected individuals of generation g

μ is the population size and $I_{sel}^{(g+1)}$ is the set of indices of the selected individuals of generation g with $\left| I_{sel}^{(g+1)} \right| = \mu$.

$k = 1, \ldots, \lambda$

The actual adaptation now takes place at two different rates due to the time scales on which the mechanisms operate; the global step size should be able to change as quickly as the search process which only has to adapt n parameters while the covariance matrix has to collect selection information for $n(n+1)/2$ parameters before making notable changes.

Looking firstly at the C matrix adaptation, the summation or cumulation of selection information used in previous generations is first collected in the evolution path for generation $g+1$ as shown in Equation 2.4.2. s is normally distributed according to $s \sim N(0, ss^T)$ and the starting values of s are zero.

$$s_c^{(g+1)} = (1 - c_c) s_c^{(g)} + \sqrt{c_c (2 - c_c)} \underbrace{\frac{\sqrt{\mu}}{\sigma^{(g)}} \left(\langle x \rangle_\mu^{(g+1)} - \langle x \rangle_\mu^{(g)} \right)}_{= \sqrt{\mu} B^{(g)} D^{(g)} \langle z \rangle_\mu^{(g+1)}} \qquad \text{Equation 2.4.2}$$

[1] We note that $C = BDD^T B^T$.

where $c_c \in [0,1]$ is astrategy parameter that determines the accumulation time $1/c_c$ for the evolution path $s_c^{(g)}$. A value for $c_c=1$ means no accumulation takes place and no correlation information is incorporated as can be seen in the above equation that the term containing the evolution path information from the previous generation would dissapear. In general, small values of c_c are found to be more accurate. [2]

The information path is then used to adapt the covariance matrix for generation $g+1$ in the following manner:

$$C^{(g+1)} = (1 - c_{\text{cov}}) \cdot C^{(g)} + c_{\text{cov}} \cdot s_c^{(g+1)} \left(s_c^{(g+1)}\right)^T \qquad \text{Equation 2.4.3}$$

where $c_{\text{cov}} \in [0,1]$ is a time characteristic for the Covariance matrix and $1/c_{cov}$ can be considered its life span. For example, after $1/c_{cov}$ generations have passed, approximately $2/3^{rd}$ of the original information may be lost. Large values of c_{cov} mean faster adaptation since there would be less input from the Covariance matrix of the previous generation and increased use of information from the new generation's evolution path. On the other hand, smaller values correspond to increased reliability in the adaptation process with more utilisation of correlation information between successive steps. As a guideline, c_{cov} should be smaller than c_c to prevent oscillations due to the summation.

The C matrix is essentially used to give directional information to the successive mutation steps. The evolution path provides information of choices made for previous mutation steps and the C matrix is then used to direct the following mutation steps so as to minimise the difference between the actual evolution path and an evolution path produced by random selection in order to maximise efficiency[3]. Values from C are used to evaluate the mutation step size in the form of B and D which are calculated from it before being inputted in Equation 2.4.1.

The C matrix also incorporates information about the shape of the objective function into the solution process. To do this, C approximates the inverse of the Hessian[4] matrix H^{-1}, transforming the mutation distribution so that it matches the topology of the objective function. As an example of this property, we look at the CMA evolution of the mutation distribution for the 2D De Jong 2 or Rosenbrock function:

$$f(x_1, x_2) = 100 - \left(x_1^2 - x_2\right)^2 + (x_1 - 1)^2 \qquad \text{Equation 2.4.4}$$

The Covariance Matrix can be seen on Figure 2.4.1 as a circle on the bottom left as it is initialised as an identity matrix. The mutation distribution becomes adapted to Rosenbrock's topology.

---

[2] Suggestions of values to be used for CMA parameters are outlined and tested in Chapter 4, Section 4.3.2.

[3] Roughly speaking, parallel correlated succesive steps mean that they are progressing in the same direction and the same distance could be covered by fewer and longer steps. Conversely, anti-parallel steps cancel each other out and both can be are inefficient with respect to a single mutation step. By reducing the difference between the distribution of the actual evolution path and one produced under randon selection, we attempt to remove correlation between successive steps and hence increase the efficiency.

[4] The Jacobian Matrix of the derivatives $\partial f/\partial x_1, \partial f/\partial x_2, \ldots, \partial f/\partial x_n$ of a function $f(x_1, x_2, \ldots, x_n)$ with $x_1, x_2, \ldots, x_n$ is called the Hessian of f [12]. By approximating the Hessian matrix, C introduces information similar to second derivative information about the topology of the function without actually calculating it.

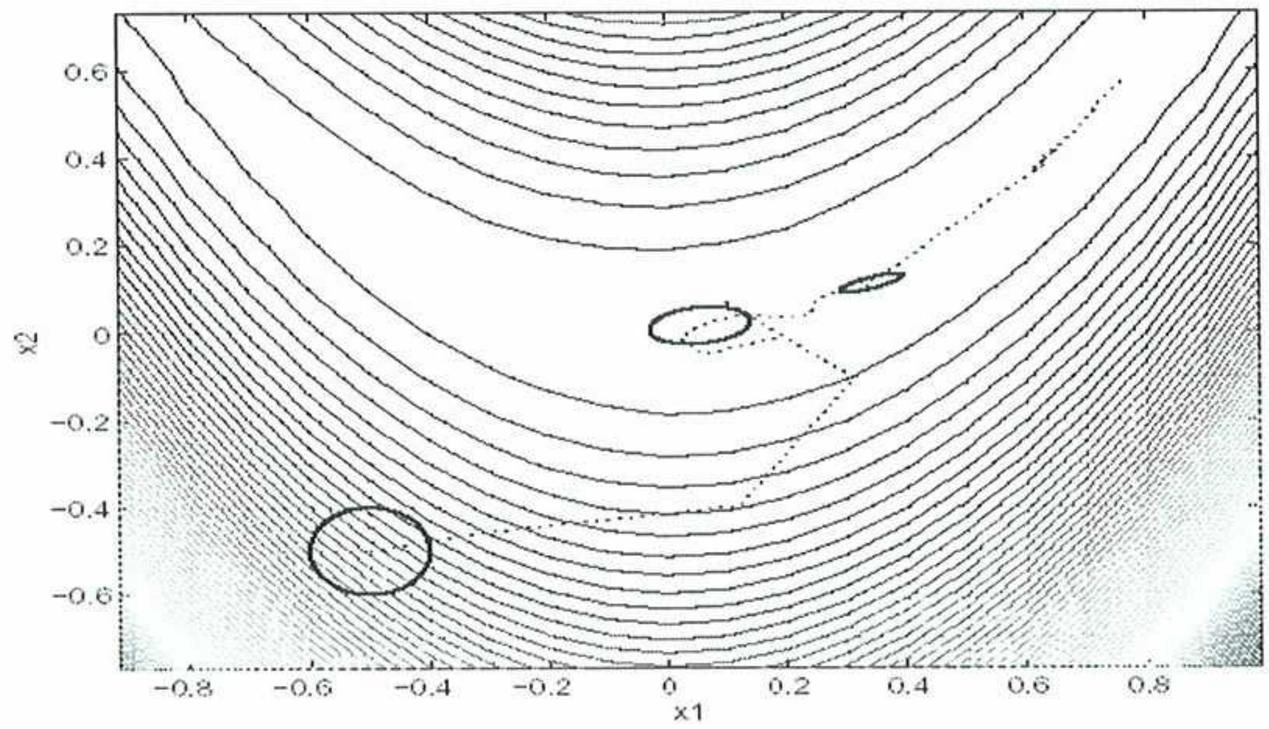


Figure 2.4.1 The evolution of the mutation distribution in the 2D Rosenbrock function optimization using CMA.[2]

Considering next the global step size, this is concurrently adapted by first calculating the evolution path $s_\sigma^{(g)}$ in a similar fashion to $s_c^{(g)}$, only here we do not use a scaling of $D^{(g)}$ since σ is a scalar:

$$s_\sigma^{(g+1)} = (1 - c_\delta) \cdot s_\sigma^{(g)} + \underbrace{\sqrt{c_\sigma (2 - c_\sigma)} \sqrt{\mu} B^{(g)} \langle z \rangle_\mu^{(g+1)}}_{= B^{(g)} \left(D^{(g)} \right)^{-1} B^{(g)^T} \frac{\sqrt{\mu}}{\sigma^{(g)}} \left(\langle x \rangle_\mu^{(g+1)} - \langle x \rangle_\mu^{(g)} \right)} \qquad \text{Equation 2.4.5}$$

The previous arguments made for c_c hold equally for c_σ in the above equation[5]. Equation 2.4.5 permits the calculation of the global step size for generation $g+1$ as follows:

$$\sigma^{(g+1)} = \sigma^{(g)} \cdot \exp\left(\frac{1}{d} \frac{\left\| s_\sigma^{(g+1)} \right\| - \hat{X}}{\hat{X}} \right) \qquad \text{Equation 2.4.6}$$

where $d>1$ is the damping parameter for the step size variation between successive generations. Larger values for damping can help to decrease premature step size reduction and d should be set larger than c_σ to avoid cumulation oscillations.
$\hat{X} = E\left[\left\| N(0, I) \right\| \right]$ is the expected length of a random vector with the normal distribution $N(0,I)$ and approximated by $\hat{X} = \sqrt{n}\left(1 - \frac{1}{4n} + \frac{1}{21 n^2}\right)$.

Whereas the Covariance matrix provides information to orientate the succeeding mutation steps, the global step size dictates the magnitude of the mutation step to be applied so that with the other right hand terms in Equation 2.4.1, the mutation step is given both direction and magnitude. CMA can also increase the global step size to "climb out" of local optima.

CMA has been shown to provide reliable coordinate system independent topology adaptation with improved convergence rates (especially on non-separable and/or badly scaled fitness functions.) Modifications on the basic original CMA code for large population problems have also been carried out [9] and concentrate on modifying terms held in Equation 2.4.3 with higher rank information being used in the covariance matrix and a modified c value.

---

[5] As will be seen in Section 4.3.2, the authors of the code suggest equal values for these two parameters by referring them both to the parameter c.

2.5 Differential Evolution Scheme 1 (DE1) [2,13]

The DE1 code contained in Optima is based on the algorithm developed by Storn and Price, described in [13]. It is a robust code and has been proven to out-perform other reputable methods such as the Adaptive Simulated Annealing approach and the Annealed Nelder & Mead method. DE1 uses a population, size NP, of vectors x_i containing n parameters, for each generation G.

$$x_{i,G}\ ,\ i = 0,1,2,...,NP-1 \qquad \text{Equation 2.5.1}$$

In the absence of information about the problem topology, the initial population is randomly generated within the constraints of the feasible search space. If a preliminary solution exists, the initial population is generated by adding normally distributed deviations to this nominal solution.

The idea behind DE is the generation of a trial vector $\underline{v}$ for each of the vectors $\underline{x_{i,G}}$ by combining a vector with the weighted difference vector between two other, randomly chosen individuals according to the differential operator:

$$\underline{v} = \underline{x}_{r1,G} + F\left(\underline{x}_{r2,G} - \underline{x}_{r3,G}\right) \qquad \text{Equation 2.5.2}$$

A 2D example of generating the trial vector $\underline{v}$ for an objective function with the shown contour lines, using DE1, is shown in the figure:

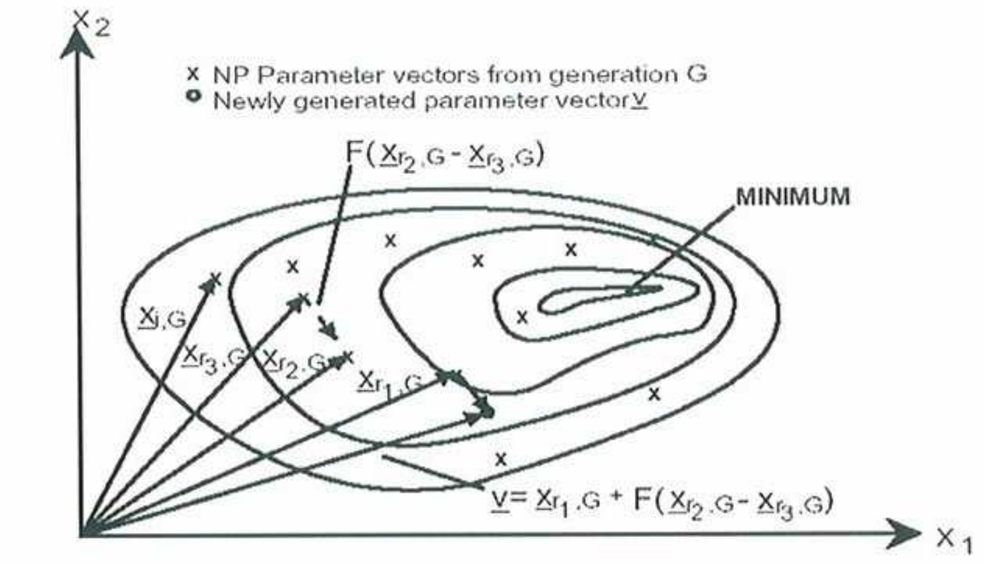


Figure 2.5.1 Generation of trial vector $\underline{v}$ using DE1.

In Equation 2.5.2, *r1, r2* and *r3* are randomly chosen integers in the range [0,NP-1] and are different to the running index *i*. *F* is a real and constant factor which controls the amplification of the differential variation between the two vectors. Together with *NP* and *F*, the crossover variable *CR* is the third control variable and dictates the crossover probability. The crossover occurs between the original vector $x_{i,G}$ and its trial vector $\underline{v}$ to form a vector $\underline{u}$. The vector $\underline{u} = \left(u_1, u_2 .....u_n\right)^T$ is formed from the two vectors according to the formula below, with *m* being randomly chosen from the range [*0,n-1*]:

$$u_j = \begin{cases} v_j \\ (x_{i,G}) \end{cases} \text{ for } \quad \begin{matrix} j = \langle m \rangle_n, \langle m+1 \rangle_n, ..., \langle m+L-1 \rangle_n \\ otherwise \end{matrix} \qquad \text{Equation 2.5.3}$$

Where the acute brackets $\langle\ \rangle_n$ denote the modulus function with modulus *n*. *n* is the number of parameters in each individual and *L* is an integer lying in the range [*0,n-1*], whose value is

equivalent to the number of parameters from $\underline{v}$ that are incorporated into $\underline{u}$. L has a probability distribution given by the crossover probability below:

$$P(L=\theta)=\frac{(CR)^{\theta}}{\sum_{\alpha=0}^{\alpha=n-1} CR^{\alpha}} \qquad \text{with } CR \in [0,1] \qquad \text{Equation 2.5.4}$$

Comparison between this new member $\underline{u}$ and the $\underline{x_{i,G}}$ member provides the test for whether to pass it to the new generation in the place of $\underline{x_{i,G}}$. The random choices for m and L are made for each trial vector $\underline{v}$ and an example for the whole DE1 offspring generation process is shown below in Figure 2.5.2 for $n=7, m=2,$ and $L=3$.

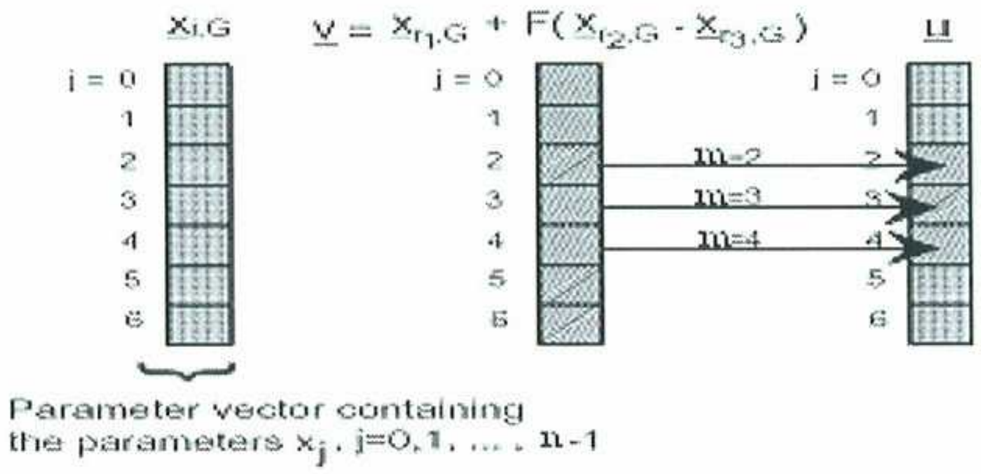


Figure 2.5.2 DE1 Offspring generation

The DE1 solution can therefore be seen to consist of a combination of a mutation process followed by crossover recombination. The right hand side of Equation 2.5.2 can be considered as the mutation step with F controlling its magnitude. A side benefit of this formulation is that as the whole population converges towards the optimum solution, the differential variation in the population decreases so that the mutation step size becomes smaller as we converge on a solution. This can be thought of as an implicit step size self-adaptation which greatly increases the efficiency of DE1.

As discussed above, the recombination process is brought about by random choices to decide the variables to be crossed over. For a 3 parameter problem, possible values of L are 0, 1 and 2. The graphical representation of the probability for this case is shown below. We see from Figure 2.5.3 that as CR is increased, then so is the probability of more of the D number of variables crossing over from the trial vector parameter into $\underline{u}$.

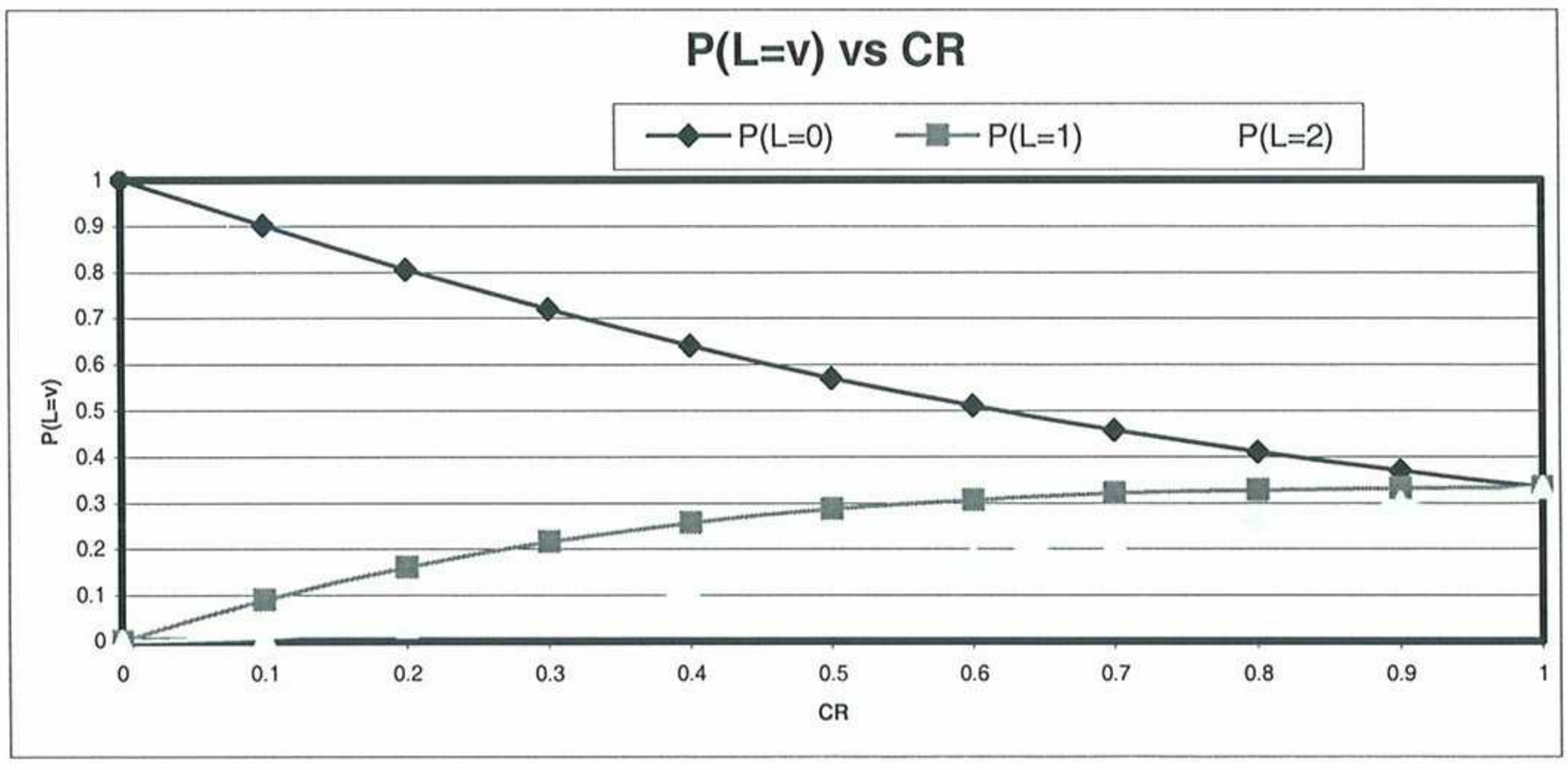


Figure 2.5.3 Original Crossover Length Probability

In Optima, a simplified version of the crossover process is used. For each degree of freedom a random number is generated. If this number is less than the set *CR* value, then the parameter from the trial vector, generated in the same way as before, is taken into $\underline{u}$, otherwise the parameter from the original vector is used in $\underline{u}$. This is basically represented in the diagram below:

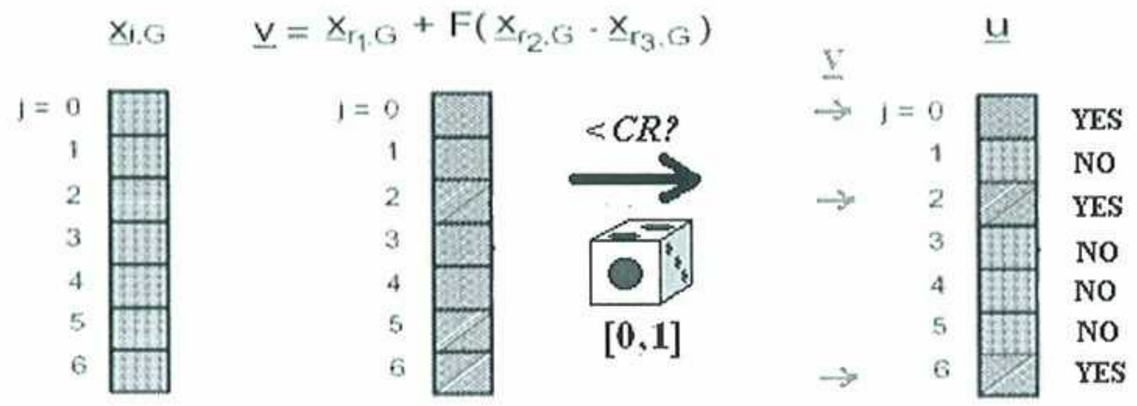


Figure 2.5.4 Optima DE1 crossover process

Formulating the crossover process in this way has the essentially the same end outcome as the original formulation in so much as that the higher we set the *CR* value, the higher the probability of increased crossover of trial vector parameters into $\underline{u}$. Using this method to control the crossover is also more transparent since by setting the *CR* to 0.8 for example, we can crudely say that over many runs, on average 80% of the vector $\underline{u}$ will consist of $\underline{v}$ parameters[6].

2.6 Optima

Optima is an optimisation tool written in JAVA and developed at CIMNE for the purpose of investigating the optimal application of the correct EA to a given optimisation challenge. It permits the user to easily investigate the control parameters for a numerical scheme that has been applied to a posed problem. In this report, it will be linked to a pre/post-processor and a FEM fluid interaction solver using Tool Command Language (TCL) in order to setup an optimisation loop. The interface for Optima is shown in Figure 2.6.1 below and allows access to the tree of variables available to construct EAs within Optima, described in Sections 2.2, 2.4, 2.5 and shown in Figure 2.7.1.

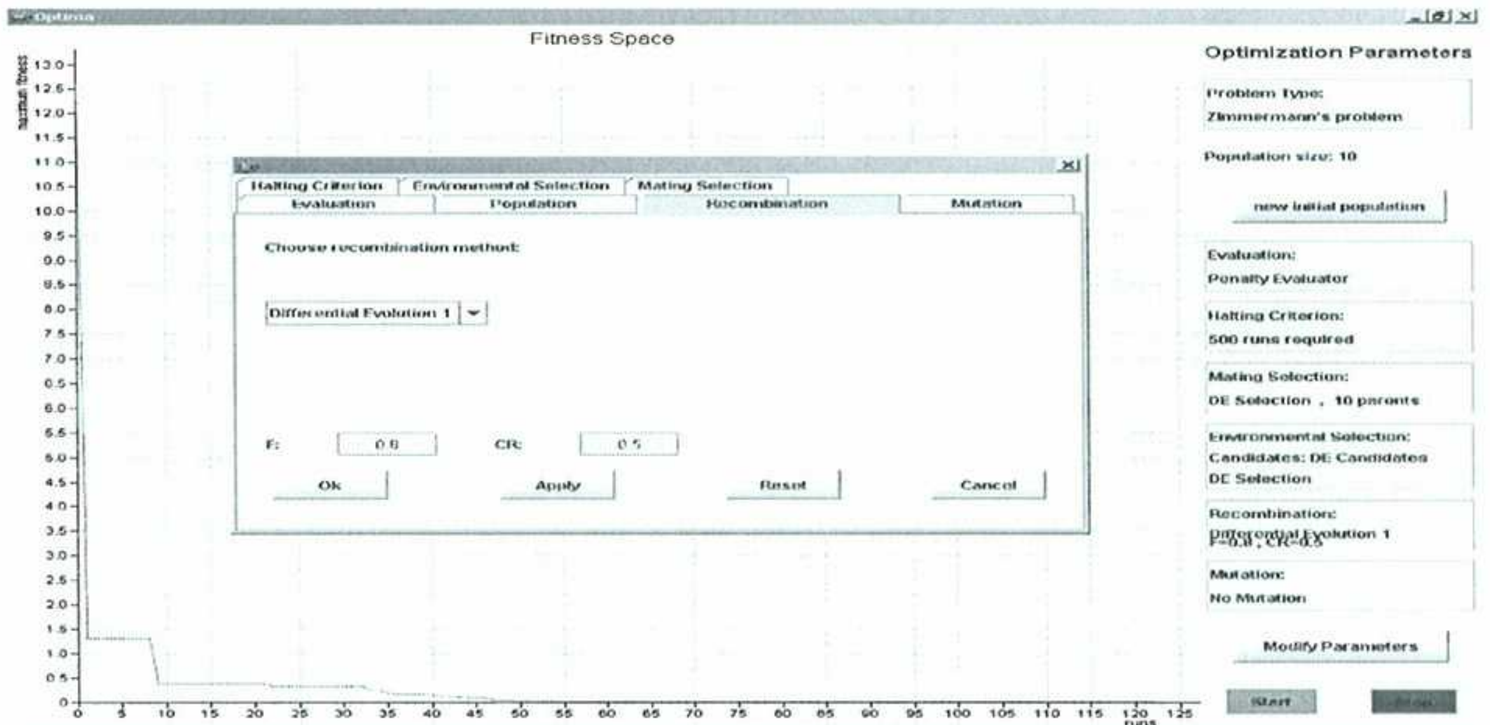


Figure 2.6.1 Optima Interface

[6] The main idea behind the two different methods is the same in that increasing CR increases the amount of crossover of $\underline{v}$ into $\underline{u}$ although the exact results are expected to differ since the original version incorporates much more randomness. For example, with Optima´s formulation we can say for certain that for *CR=0* there is no crossover and all of the original vector parameters are retained, for *CR=1.0* $\underline{u}$ will be identical to $\underline{v}$ and this is not the case with the original version.

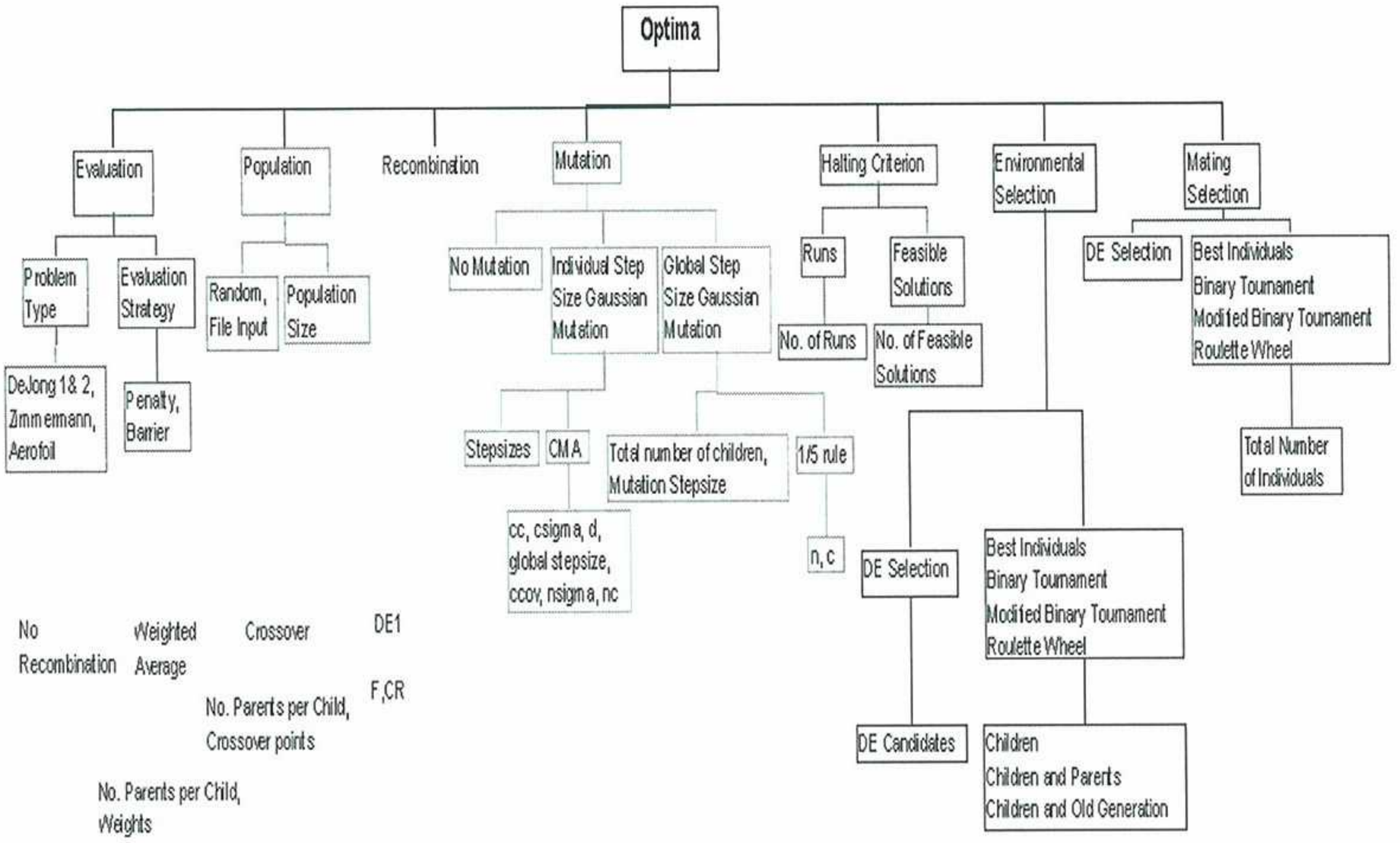


Figure 2.7.1 Optima Operators and Parameters

2.7 XFOIL [14]

The XFOIL aerofoil program was developed by Professor Mark Drela at MIT. Since its conception, it has been thoroughly scrutinized and undergone testing and development in order to establish the present version 6.94 of the program. This means that under non-extreme conditions, the data from this program can and will be used as a resource or test bed against which analytical results for aerofoils can be evaluated. Tdyn is to be used for problem simulation so XFOIL will only be used for comparison purposes. Here, the term extreme would correspond to when operating with significant separation for which the viscous model is insufficient or at speeds where the local Mach number at any point on the aerofoil exceeds 1.05 when shock losses become important and under-represented.

The inviscid formulation in XFOIL follows a linear-vorticity stream function panel method. The viscous system uses a two equation lagged dissipation integral boundary layer formulation. The boundary layer and transition equations are solved simultaneously with the inviscid flow, incorporating a Karma-Tsien compressibility correction, by a full-Newton method. The drag is calculated from the wake momentum thickness far downstream.

When solving given a specified angle of attack 'alfa', as is normally the case here, the wake trajectory is taken from an inviscid solution at that 'alfa'. Viscous effects generally act to decrease lift and change the trajectory, but in attached flow calculations with which the aerofoil designs in this report will be involved, the inaccuracy in the inviscid assumption is imperceptible.

2.8 Pre/Post-processor GiD

GiD is a pre-post processing tool in which attributes such as material types, loadings and conditions are assigned to the generated geometry prior to the existence of a mesh. Attributes can also be assigned to the meshed geometry but have to be redefined every time a new mesh is generated. The complete solution of a problem consists of the steps below where Tdyn is the FEM solver used and it is hoped that the combination of the two programs with Optima will lay the ground work to allow the optimisation of more complex geometry in the future:

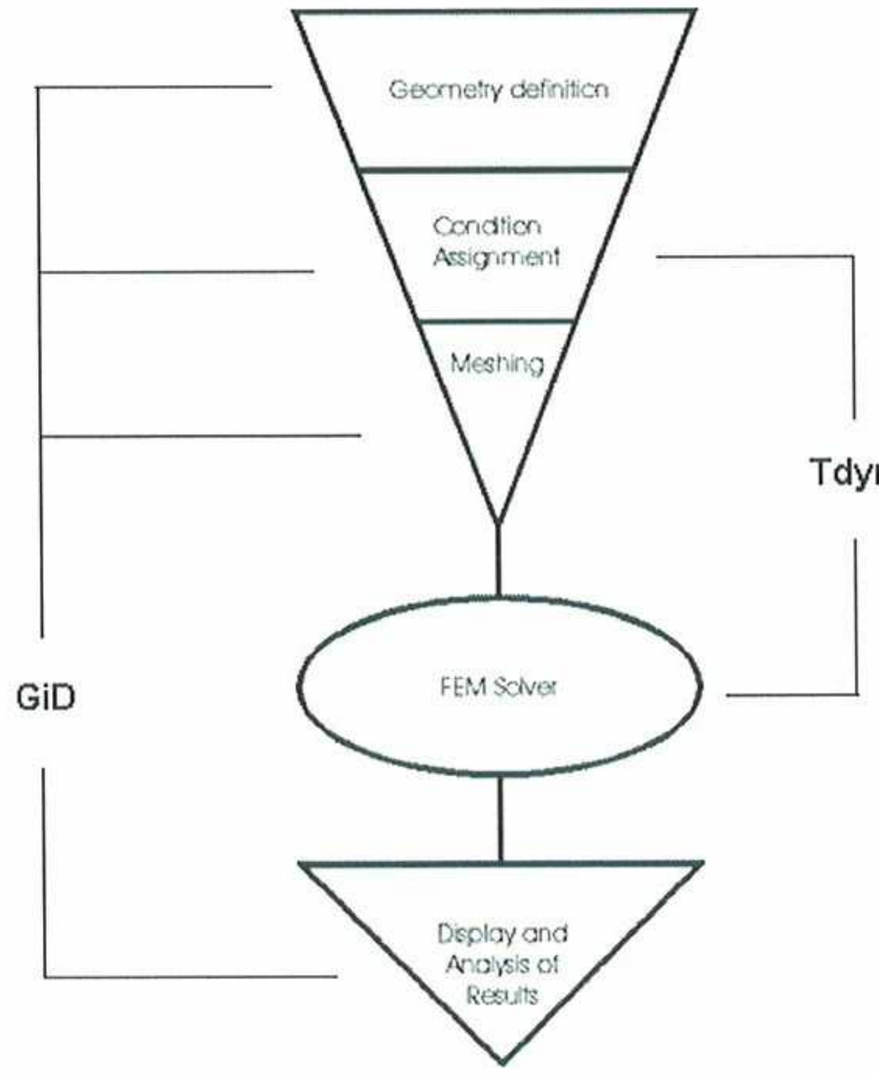


Figure 2.8.1 GiD Solution Process

2.9 Tdyn – FEM Fluid Dynamics Solver [15, 16,17, 18]

Tdyn is a Finite Element Method (FEM) fluid dynamics simulator and is a very important part of the optimisation process since it will be relied on to model a problem and produce the required results. The main findings are presented in this section but a full summary of the theory behind Tdyn, its calculation conditions and test results are contained in Appendix B.

Tdyn solves the three dimensional, incompressible and slightly compressible Navier-Stokes equations. The finite element method is used to space discretize the Navier-Stokes equations while an iterative implicit two steps Fractional Step type method is used for time discretisation. Stabilisation of convection dominated problems is done by the Finite Calculus method. The standard time discretisation of the final stabilised momentum balance equation produces the velocity and pressure equations which are solved iteratively by Tdyn to provide the required results.

After the geometry has been created, conditions specific to the Tdyn simulation and calculation are assigned before the problem is passed back to GiD for meshing. The conditions that need to be set for a viscous aerofoil problem include initial conditions, boundary conditions and turbulence models. Testing of conditions and models to establish an accurate model are discussed in the next section.

2.9.1 Test Results and Overall Choice of Model [19, 20, 21]

A problem was set up with a horizontal velocity only, specified as 68 m/s at the inlet, and pressure specified as zero on the other sides of the control volume. Air was assigned as the fluid with values corresponding to a density $\rho=1.225\ kg/m^3$, a viscosity $\mu=1.736e\text{-}5\ kg/ms$ and a Reynolds number of $Re \approx 4.8x10^6$. A law of the wall function was then specified on the zero angle of attack NACA 0012 aerofoil geometry [22] (refer to Chapter 3 for geometry generation details), a turbulence model was chosen and initial conditions were set with velocity equal to 68 m/s in the x direction. The completed problem is then meshed with an 0.0008 element size specified at the leading and trailing edges, 0.001 for the rest of the aerofoil surface and 2.0 for

the outer mesh with a transition rate between the aerofoil surface and outer mesh of 0.2. This resulted in a 15,000 line element unstructured mesh[7] and the problem is then run with 0.001 time steps for 400 iterations[8]. The meshed geometry is shown in the next two figures:

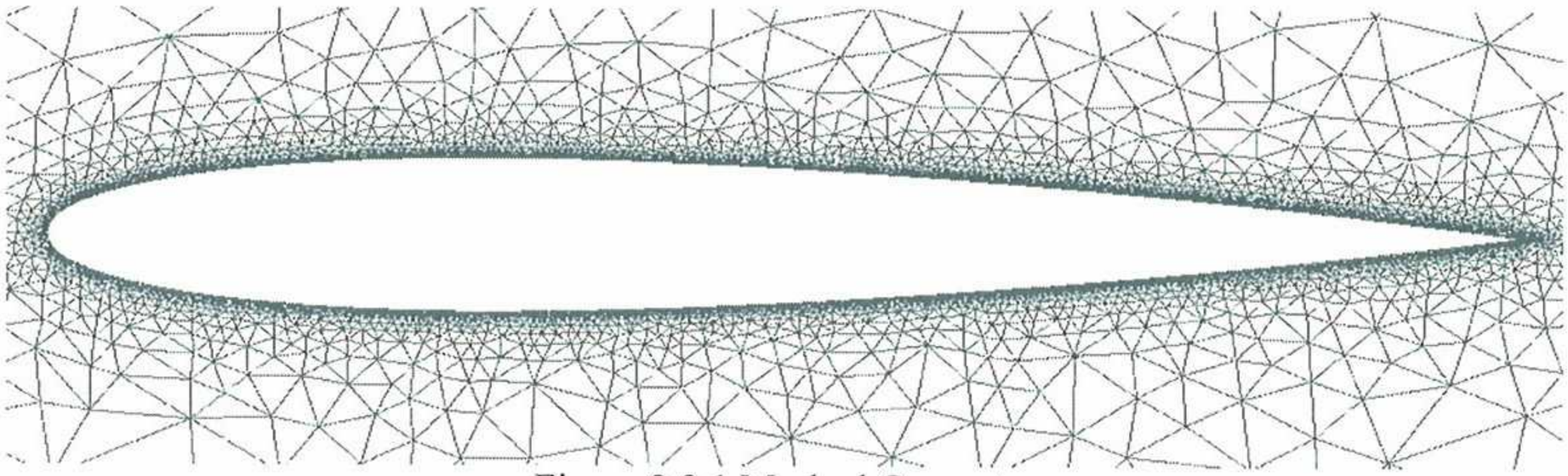


Figure 2.9.1 Meshed Geometry

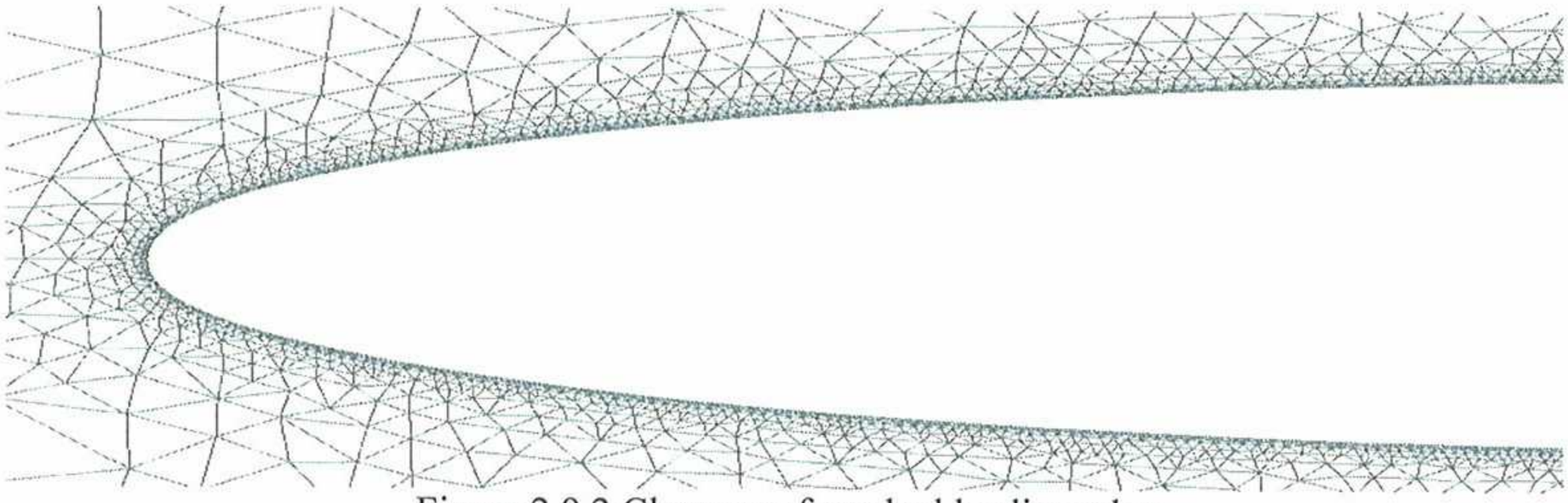


Figure 2.9.2 Close-up of meshed leading edge

The two main variables to be considered here are the fluid boundary assigned to the aerofoil surface and the turbulence model. A law of the wall model[9] is assigned to the aerofoil surface/ fluid boundary for two main reasons:

i The law of the wall uses analytical results based in the log law region and requires no meshing of, nor calculations in, the viscous sub-layer which would require a very large number of small elements in a fine mesh (as is the case for Direct Numerical Simulation) and high calculation cost.

ii Some of the current turbulence models used in Tdyn have not been formulated to fully model viscosity variations in the viscous sub-layer and wall functions are used here to bridge this region in the boundary layer more effectively[10].

With this in mind, the most practical model for aerofoil simulation is the YplusWall model. Using this wall function, the law of wall is applied with the non-dimensional distance y^+ as the user specified lower bound. Wall functions are formulated based on the logarithmic relation which holds true in the log law region and through this we can quite accurately predict the velocity there. At the first node of the mesh outside of the aerofoil surface, the log law is used to

---

[7] An unstructured mesh is used since the setting up of a structured mesh in GiD is too complicated, especially if it is to be used in an optimisation process so that it has to be generated automatically without user interference at every step in the optimisation procedure.

[8] Using 0.001 time steps and 400 iterations, the viscous and pressure forces required to calculate the lift and drag on the aerofoil can be seen to settle after this time period, for most of the turbulence models, in Appendix D.

[9] Boundary layer modelling is discussed in Appendix B.2.4

[10] Models such as the Spallart-Allmaras and k-ω SST models are known to be accurate in the viscous sublayer but models such as the k-ε model fail to precisely model separating flows and the high viscous stresses present in this low Reynolds number region.

calculate the velocity in the form of the non-dimensional time averaged velocity parallel to the wall, U^+. This is then used as a Dirichlet type boundary condition for the turbulence model solution of the governing equations (RANSE – see next paragraph) from the first node onwards. The upper limit of the log law range as defined in Tdyn is given roughly $y^+<100$. By having a low y^+ as the starting value of $y^+=32$ for the log law region, we have a large range over which the wall function is applied[11]. Also, by using a fine mesh, we can try to ensure that the first node lies in this specified region so that the combination of the wall function and the turbulence model accurately simulates turbulent flow.

Considering turbulence models next, these relate the mean flow variables to the unknown Reynolds Stresses, which result from a Reynolds Averaged formulation of the Navier-Stokes Equations (RANSE), through the use of an eddy viscosity[12]. The turbulence models are classified by the number of partial differential transport equations, usually describing turbulence velocity and length scale, that have to solved in addition to the RANSE to solve for eddy viscosity and complete the system of equations. For example, zero equation models use purely empirical representation of the mean flow variables and are least accurate whereas two equation models employ two transport equations and account for, to some level, flow history effects[13].

To find the most appropriate model for our problem simulation, a comparison of nine Tdyn models was conducted with the same settings in each test, in an attempt to establish a transparent selection process. The nine models include Smagorinski (Smag), Kinetic Energy Two Layers (KE2L), Spalart-Allmaras (SpAl), k-ε Launder Sharma (kεLS), k-ε High Reynolds (kεHRe), k-ω (kO), K-KT (KKT), k-ε Lam Bremhorst (kεLB) and k-ω Shear Stress Transport abreviated to SST (kOSST). The expected coefficients for the NACA 0012 aerofoil in such conditions were $CL = 0.0$ and $CD = 0.00506$ as given by XFOIL. The results for the comparison are given below in Table 2.9.3 where a breakdown of the forces and the type of model is given, i.e. a zero, one or two transport equations model.

The total forces in this table directly give the lift and drag forces since we are running with a zero angle of attack. Lift and drag coefficients, CL and CD respectively, are also provided and these have been calculated by dividing the total forces by the constant (0.5*p*V2). In Tdyn, the viscous and pressure forces were calculated by integrating the appropriate stresses and forces around the aerofoil surface. It is clear that there are substantial errors in the drag values for all of the used turbulence models which most likely stems from the inaccurate simulation of the x-component of the viscous forces. When we consider a balance of the lift coefficient and drag coefficient inaccuracies, it can be generally seen that zero and one equation models have better drag calculation performance with the exception of the Kinetic Energy Two Layers model, whilst, on the whole, the two equation models have a better lift approximation.

		1 Smag	2 KE2L	3 SpAl	4 kεLS	5 kεHRe	6 KO	7 KKT	8 KεLB	9 KOSST
No. Eqns		0	1	1	2	2	2	2	2	2
Total Pressure	x	43.837	243.6	44.14	43.8	43.799	43.298	44.588	39.196	43.318
	y	-19	-4.543	-8.777	-0.608	-0.621	0.778	-11.758	-0.762	-1.84
Viscous Force	x	12.121	3.864	16.76	39.31	39.308	38.124	16.212	39.542	31.671
	y	0.016	-0.002	0.017	-0.003	-0.003	-0.004	0.019	-0.005	-0.003
Total Force	x	55.958	247.4	60.9	83.11	83.107	81.422	60.799	78.738	74.988
	y	-18.741	-4.545	-8.76	-0.611	-0.624	0.774	-11.739	-0.767	-1.843
C_L		-0.007	-0.002	-0.003	-2E-04	-0.0002	0.0003	-0.0041	-3E-04	-0.0007
C_D		0.0198	0.087	0.022	0.029	0.02934	0.0287	0.0215	0.0278	0.02648
C_{DTdyn}/C_{DXFOIL}		3.9047	17.27	4.25	5.799	5.79913	5.6816	4.2425	5.4943	5.2326

Figure 2.9.3 Tdyn Turbulence Models Comparison

[11] The Tdyn specified lower limit for the log law region is approximately $y^+>30$.
[12] This is in accordance to the Boussinesq hypothesis which postulates that in analogy to molecular diffusion, the Reynolds stresses depend on the deformation rates of the mean flow.
[13] See Appendix B.2.3 for further information.

The pressure and velocity distributions are mostly similar for all of the models but it is interesting to observe the eddy viscosity simulation by each candidate model. Figures 2.9.4 and 2.9.5 show the eddy viscosity distribution for two of the turbulence models which perform well when considered relative to the other models. However, their eddy viscosity distributions are seen to be unexpected and inaccurate when compared to the distribution from the for k-ω SST Model shown in Figure 2.9.6. The velocity and eddy viscosity distributions along with the pressure force and viscous force time response graphs for the nine tested turbulence models are contained in Appendix D for easy comparison.

Figure 2.9.4 Eddy Viscosity Distribution for Smagorinski Model

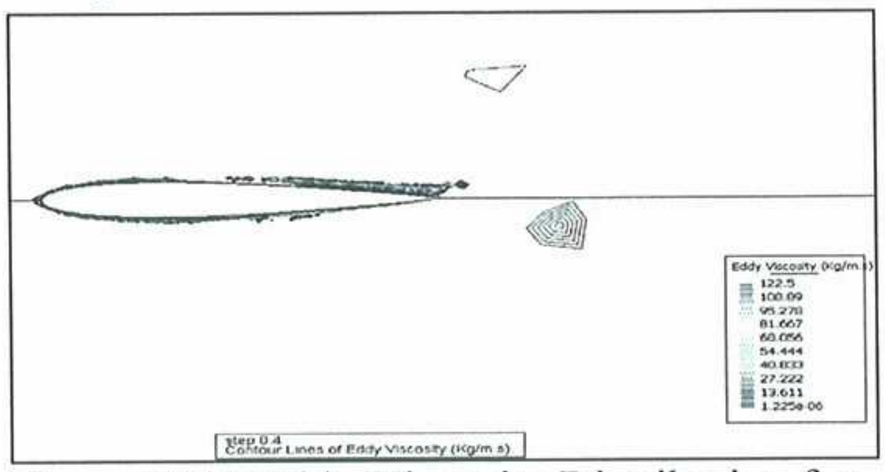


Figure 2.9.5 Eddy Viscosity Distribution for Ke Lam Bremhorst Model

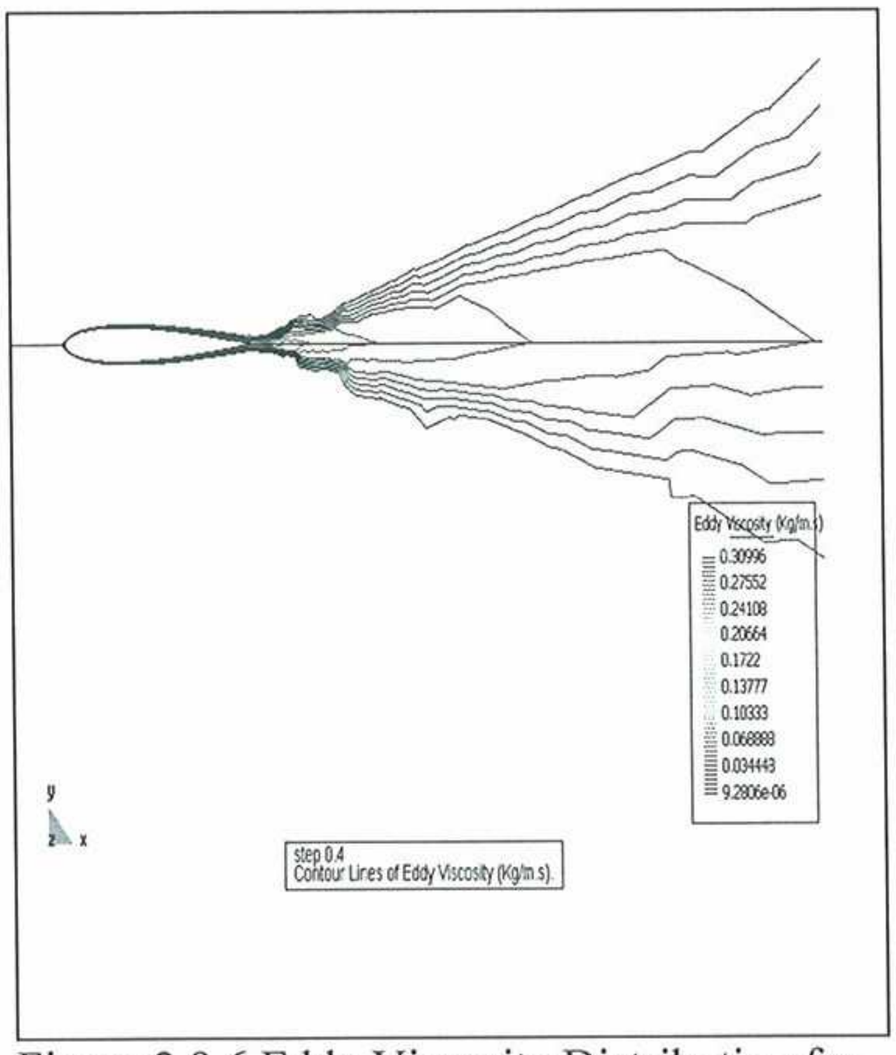


Figure 2.9.6 Eddy Viscosity Distribution for k-ω SST Model

On balance of all of these factors, the k-ω SST Model was chosen as the best turbulence model due to its more accurate lift, lower drag and realistic viscosity model. As will be seen in Section 2.9.2, the required pressure readings using this model are fairly accurate and when the mesh size was refined with smaller elements, the ratio C_{DTdyn}/C_{DXFOIL} decreased to around 4. In general literature, k-ω SST Model has also been seen to be more accurate than its popular rival, the k-ε model, for the types of problems considered in this report [23, 24].

2.9.2 Tdyn Limitations

There are several factors which effect the accuracy of the results produced by Tdyn: The program is founded on the solution of the incompressible Navier-Stokes equations and therefore we are restricted to test problems below certain Reynolds and Mach numbers such as *Re~7e06* and *M~0.3* (in standard sea level conditions) at which compressiblity effects start becoming important.

Working with the Finite Element Method, the meshing quality and properties are of obvious importance. For the automated aerofoil optimisation problem, the use of a structured mesh in GiD is overly complicated and when resorting instead to an unstructured mesh, it is sometimes not possible to achieve a completely symmetric mesh over the domain. This can lead to small assymetric loading conditions on the geometry, even when using very fine meshes. Another problem of the unstructured mesh is that it is difficult to control the placement of the first node and there is also currently no way in Tdyn to easily check that all of the first nodes surrounding the aerofoil surface fall into the required log law range. This could be a major cause of inaccuracies in the force calculations by the turbulence model.

Total forces acting on a body are calculated by integrating the normal pressure forces and the viscous forces on the body surface. The true body profile is simulated and discretised by mesh elements so that an inaccuracy may arise due to the discontinuous variation of the forces to be integrated. These values may then be compounded and multiplied to different extents by different turbulence models and could account, to a certain extent, for the erroneous values discussed in Section 2.9.1. The turbulence models themselves also have shortcomings due to their formulation according to the Boussinesq hypothesis limiting their effectiveness at describing streamline curvature which can be a problem especially when modelling flow over an aerofoil or wing.

The issues faced here are currently being resolved and the program is, as it stands, able to provide accurate pressure results for use in Problem 1 (the inverse optimisation task where an aerofoil profile is recovered using pressure values), as can be seen in Figure 2.9.7, but gives erroneous force values. Further tests were carried out using the k-ω SST model on a range of aerofoils where mesh elements of length 1e-4 to 1e-5 were assigned to the aerofoil surface in an attempt to ensure all of the first nodes closer to the surface and into the log law region. By doing so, a total of 30,000 line elements were used, but the results did not improve and they were seen consistently to produce errors or around four times the XFOIL value. With time considerations on mind, a simplified mesh which produced the same range of results as the previous tests was used, consisting of a fine mesh of size 0.001 near the wall and a coarse one further away, resulting in a much simplified mesh consisting of 4,000 line elements. This sparser mesh allowed the running efficiency of the optimisation process to be improved but even at these settings, a test in Problem 1 could take a day to solve. For Problem 2 (minimising an objective function containing a combination of lift and drag coefficients) the force values are most important. But it must be remembered that the focus in this report is on the optimisation procedure and this can still be done in a meaningful way even with these errors, if not quantitively then at least qualitatively for purposes of comparison of the mechanismns of the different algorithms and settings. When carrying out this second optimisation problem therefore, we can try to consider the drag value error as systematic noise and bear in mind that the force value will place a restriction on how low the fitness can be reduced to. Taking all of this into consideration and remembering that part of the set task was to incorporate this CFD solver into the optimisation process, we can still proceed with GiD and Tdyn for the problem representation and Optima as the optimisation tool in all discussed problems.

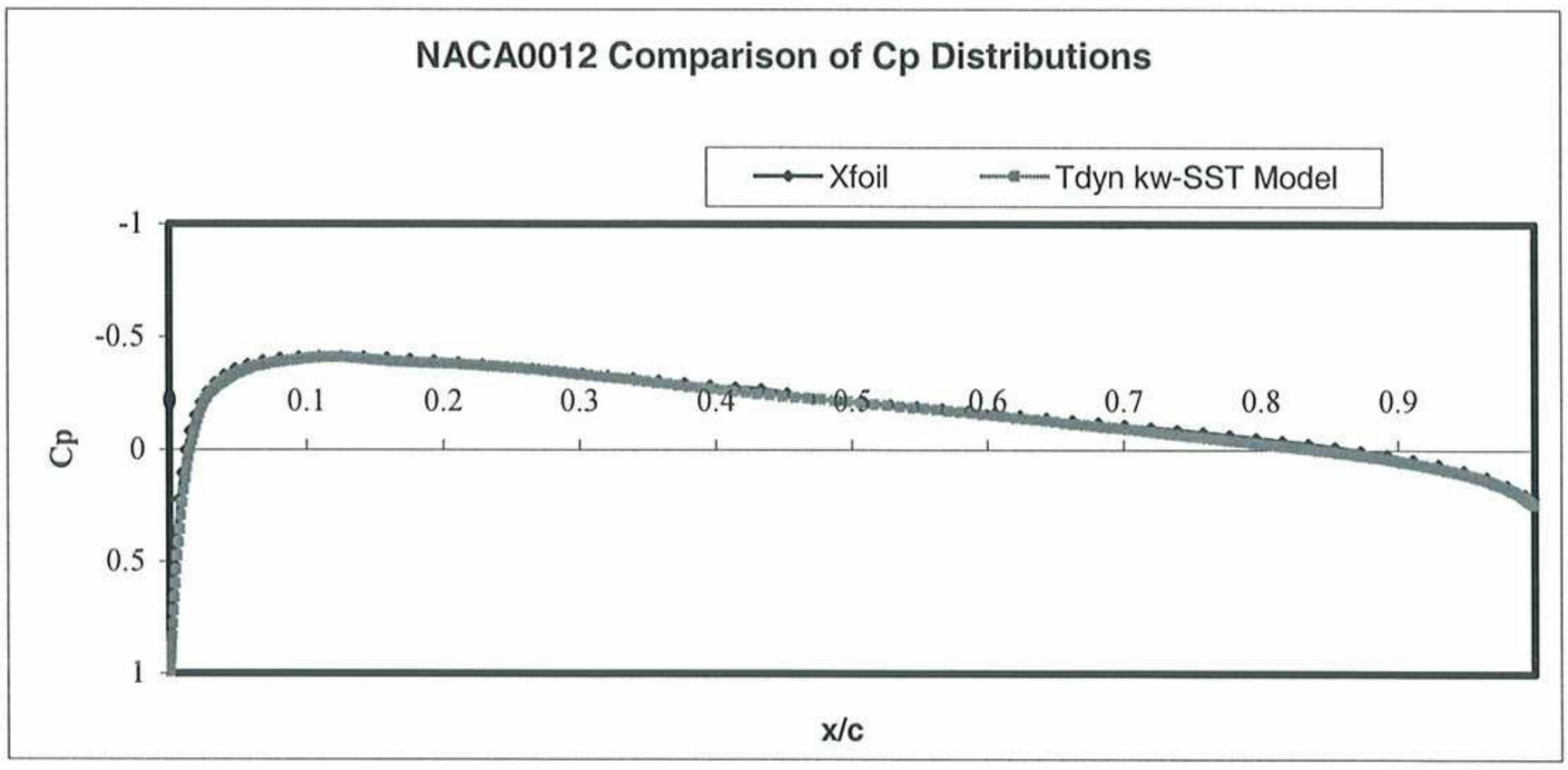


Figure 2.9.7 Comparison of NACA0012 Cp Distributions for Tdyn k-ω SST and XFOIL

3 Problem Generation

3.1 Geometry Definition

NACA 4 digit standard aerofoils are to be used in this report and the following 4 equations [25] suffice to accurately generate any possible permutation of this class of NACA aerofoil:

$$\pm y_t = \frac{t}{0.2}\left(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4\right) \qquad \text{Equation 3.1.1}$$

$$y_c = \frac{m}{p^2}\left(2px - x^2\right) \qquad \text{from } x=0 \text{ to } x=p \qquad \text{Equation 3.1.2}$$

$$y_c = \frac{m}{(1-p)^2}\left((1-2p) + 2px - x^2\right) \qquad \text{from } x=p \text{ to } x=c \qquad \text{Equation 3.1.3}$$

$$y_{TOTAL} = y_c \pm y_t \qquad \text{Equation 3.1.4}$$

where t is the maximum thickness, m is the position of maximum camber and p is the maximum camber, all of these being as a percentage of chord length.
y_t is the thickness distribution, positive for the upper surface and negative for the lower surface.
y_c is the camber distribution with Equation 3.1.2 applying to the section of the aerofoil from the leading edge up to the position of maximum camber and Equation 3.1.3 applying to the camber distribution from the position of maximum camber up to the trailing edge.

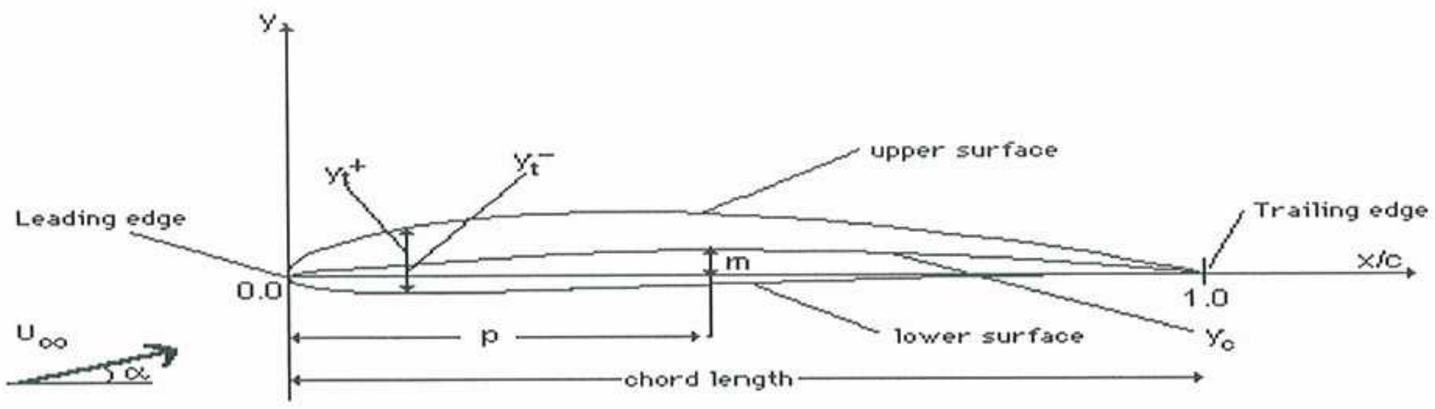


Figure 3.1.1 Aerofoil and general notation.

The total aerofoil profile is then given by Equation 3.1.4. Figure 3.1.1 shows the representation of a NACA aerofoil whose notation can be shown by taking NACA2415 as an example. The first digit represents maximum camber which here is 2% of the chord length, the second digit accounts for position of maximum camber which in this case is located at 40% of the chord. The last 2 digits give the maximum thickness which for the NACA2415 is 15% of the chord length.

3.2 Problem Geometry Realisation

Aerofoil profiles will be represented by Nurbs lines generated in GiD using 19 coordinate control points. Nurbs lines, are non-uniform rational B-splines which can interpolate between points to produce a smooth curve and the coordinate points used to generate the line are ensured to lie on the curve.

The geometry, consisting of an aerofoil in a control volume (*CV*), is then constructed as shown in Figure 3.2.1. The aerofoil shape described by the 19 coordinate points is constructed as a whole in the middle of two surfaces (marked by pink). By assigning air properties to the two surfaces (with $\rho=1.225kg/m^3$ and $\mu=1.7365\text{-}05kg/ms$) and YplusWall (with $y^+=32$) as the fluid

boundary/ wall function on the aerofoil surface as shown in Figure 3.2.2, it is possible to simulate an aerofoil moving in the Eulerian frame of reference.

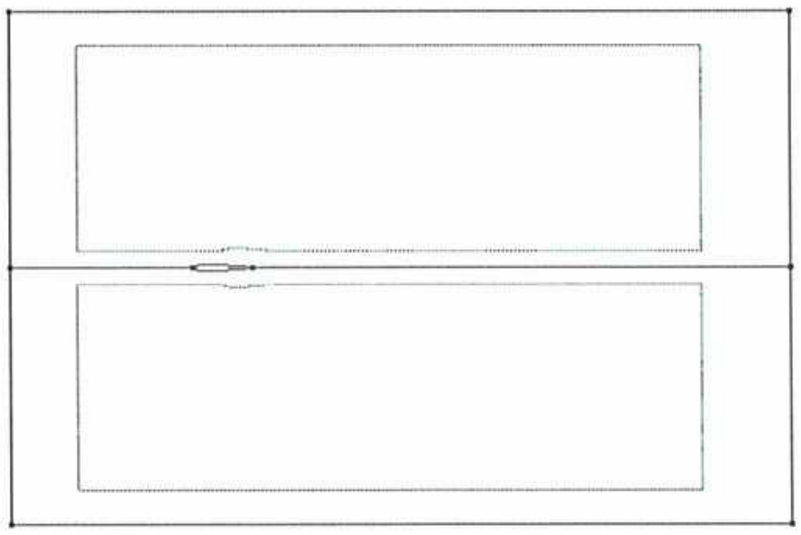


Figure 3.2.1 CV & Aerofoil

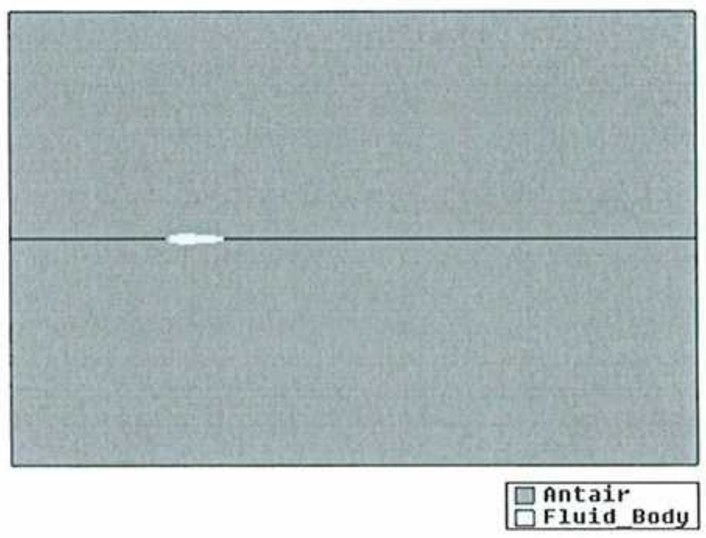


Figure 3.2.2 Material and Fluid Boundary Assignment

Boundary conditions are also assigned: Velocity is specified as zero vertical velocity and horizontal velocity as $V=68m/s$ or $M\approx0.2$ at the inlet on the left hand side. Pressure is specified as P *or* $P_\infty=0Pa$ as a reference value[14] on all other sides as shown in Figure 3.2.3. The initial conditions were set with horizontal velocity equal to 68m/s and the problem is then meshed as described in Section 2.9.1, shown here in Figure 3.2.4.

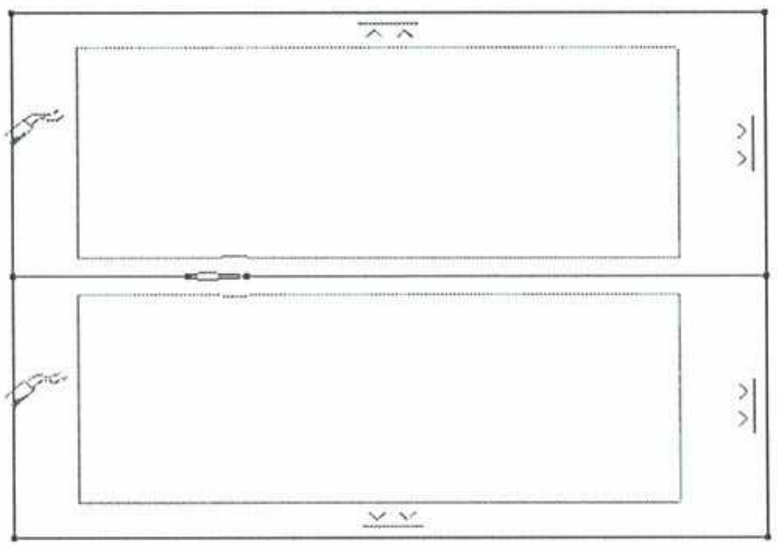


Figure 3.2.3 Boundary Conditions

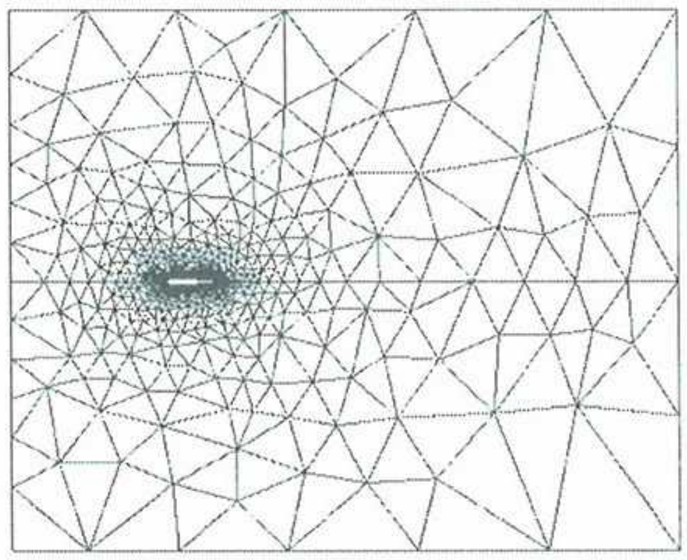


Figure 3.2.4 Meshed View

Similarly to Section 2.9.1, the time increment is set to 0.001 with 400 iterations. The turbulence is represented by the chosen k-ω SST Model and the results output is set at the 400th step in Ascii format.

3.3 Geometry Constraints

To ensure that any aerofoil geometry generated during the optimisation process and represented by the 19 coordinate points is realistic, several constraints have to be set:

i. To correctly represent the leading edge, three coordinates were specified in almost a vertical line to emulate tangency at the rounded leading edge.

ii. The y values for the upper surface always have to be greater than the lower surface y values to ensure positive thickness and no crossing over of the profile.

iii. The values m, p and t were specified to lie in the range $0<m,p,t<0.5$

---

[14] The pressure will be compared with Xfoil results which outputs coefficients of pressure, or Cp, given by $Cp = P - P_\infty / (0.5 * \rho * V^2)$, where P_∞ is the reference pressure and ρ is the fluid density. Having a reference pressure of zero simply means we do not have to scale the results by subtracting P_∞ every time.

iv. The trailing edge angle was specified to be greater than 13° according to Equation 3.3.1, since very small trailing edge angles are over idealised and can not be physically constructed in real life .

$$13^{O} < \tan^{-1}\left(\frac{y_1}{0.2}\right) - \tan^{-1}\left(\frac{y_2}{0.2}\right) \qquad \text{Equation 3.3.1}$$

where y_1 and y_2 correspond to the x-coordinate 0.8 for the upper and lower surfaces, respectively.

3.4 Simulation and Evaluation of Objective Function

Using Tool Command Language (TCL), Optimisation is created as a selectable problem type in GiD (the code for this is contained in Appendix A.1) and establishes the link between the Optima, GiD and Tdyn. Once this problem type is selected, the optimisation procedure described below is set into motion.

Two batch files, Coords.bch and Loop.bch, are pre-written in GiD command language and on activation of the Optimisation problem type, Loop.bch is executed. Coords.bch creates the aerofoil profile and is called by Loop.bch which assigns conditions, meshes, saves and calls Tdyn to solve the problem. For Problem 1 (Section 3.4.1), the basic running file for Tdyn is the .bas file and a second .bas file is written to tell Tdyn to output a projectname.dat file containing information about the meshed aerofoil nodes and coordinates which are compared against a projectname.res file generated by Tdyn containing the solved pressure values. This comparison allows the matching of pressure values to the 19 coordinate points defining the aerofoil. For Problem 2, (Section 3.4.2), the forces on the aerofoil to be used are automatically outputted by Tdyn into a projectname.flavia.for file.

Optima waits to receive the appropriate results files from Tdyn and evaluates the fitness of the geometries. Using the selected optimisation strategy, Optima generates trial solutions (new m, p and t values) using the four equations in Section 3.1 and subject to the geometry constraints in Section 3.3, in our three degree of freedom (DOF) problem.

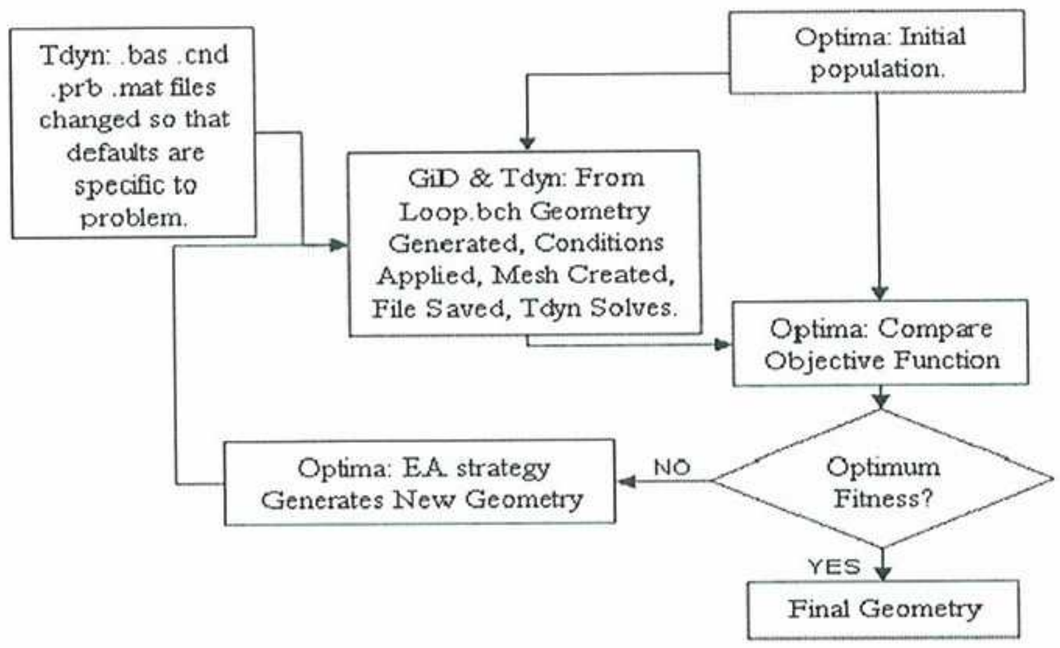


Figure 3.4.1 Optimisation process

The new suggested geometry is then handed to GiD and Tdyn and the bottom loop shown in Figure 3.4.1 is repeated until our halting criterion, which is 59 generations for Problem 1, and 55 generations for Problem 2, is reached. However, the efficiency of an algorithm will also be judged by how quickly it converges to a fitness of 1e-05 for the inverse problem. A summary of the GiD and Tdyn files that need to be generated and modified in order to set default values and run the optimisation problem are given below:

- Tdyn.bas – Execution file, this is read every time Tdyn is run and writes pressure values for every node in the meshed geometry.
- Tdyn-2.bas – Writes aerofoil coordinates and node numbers to Projectname.dat. (Appendix C.4)
- Projectname.mat – File containing default specifications of material properties i.e. air (called antair here) inside the control volume and also the fluid boundary/ wall function to be used. (Appendix C.6)
- Projectname.prb – File containing default settings for problem data such as total run time, time increments, results format and output step. The default turbulence model can also be chosen here along with the initial conditions for velocity and pressure. (Appendix C.5)
- coords.bch – Generates initial geometry. (Appendix C.2)
- loop.bch – Calls coords.bch and generates, saves and runs rest of problem. (Appendix C.3)

By having certain default settings, the optimisation run is automated can be looped without the need to re-specify any settings.

3.4.1 Problem 1 – Inverse Problem Objective Function

The inverse problem involves the recovery of an aerofoil shape, starting with a NACA0012 profile, by targeting the pressure distribution over the NACA2415, which is the desired final shape, guided by the geometry constraints of Section 3.3. For this problem, we aim to minimize the objective function, set out in Equation 3.4.1, which is formulated as the squared difference between the final target pressure distribution Cp_f and the current pressure distribution Cp_c. The pressures for the entire aerofoil are calculated by Tdyn and the values to be compared, which correspond to the 19 coordinate control points defining the geometry, are interpolated from the .res and .dat results output files.

$$F(x) = \sum_{i=1}^{19} \left(C_{Pc} - P_{Pf}\right)^2 \qquad \text{Equation 3.4.1}$$

3.4.2 Problem 2 – Direct Problem Objective Function

The second problem takes the form of Equation 3.4.2 below, where C_D is the drag coefficient, C_{Lc} is the current lift coefficient, C_{Lf} is the final target lift coefficient and P is a penalty function which gives the problem a realistic geometrical range. By multiplying the difference between the current lift coefficient and the final target lift coefficient by a large constant, we direct the problem towards optimising to achieve a certain lift and we also reduce the input of the erroneous drag values, discussed in Section 2.9.1, into the optimisation procedure.

$$F(x) = C_D + 20\left(C_{Lc} - C_{Lf}\right) + P \qquad \text{Equation 3.4.2}$$

The previously mentioned geometry constraints are retained and the penalty function places another guideline on the search and dictates that the aerofoil area A should comply with:

$$A_{\min} \le A \le A_{\max} \text{ with a penalty } P$$

$$P = \begin{cases} \left(A - A_{\max}\right)*100 & A > A_{\max} \\ \left(A_{\min} - A\right)*100 & A < A_{\min} \end{cases} \qquad \text{with } A_{max} = 0.10 \text{ and } A_{min} = 0.07.$$

The target lift coefficient for this problem is then specified, with a zero angle of attack setting in mind, as: $C_{Lf} = 0.19$.

4 Validation of Optima

The realisation of the DE1 code in Optima will first be verified against three well known test functions with the original DE1 test values available in the [13]. There are no reference values for ES-CMA but the algorithm will also be tested against the three functions and a comparison of its performance will be used as an indicator of it convergence properties when carrying out Problems 1 and 2. Following this, the parameters in DE1 and ES-CMA will also be varied in an attempt to establish the best settings to be used with these schemes. The tests in this chapter, done from section 4.2.2 onwards use the same initial population so that it is possible to see the effect of changing the settings and eliminate the effects of having a varying population.

4.1 Standard Parametric Optimisation Objective Functions

The functions to be tested are the First De Jong Function (Sphere), the Second De Jong Function (Rosenbrock´s Saddle) and the Zimmermann Function.

4.1.1 First De Jong Function

The formulation for this function is given below in Equation 4.1.1:

$$f_1(\underline{x}) = \sum_{j=0}^{2} x_j^2 \quad \text{with} \quad x_j \in [-5.12, 5.12] \qquad \text{Equation 4.1.1}$$

Figure 4.1.1 shows the graphical representation of the function with three degrees of freedom (DOFs), whose minimum is located at $f_1(\underline{0}) = 0$ and is considered easily found.

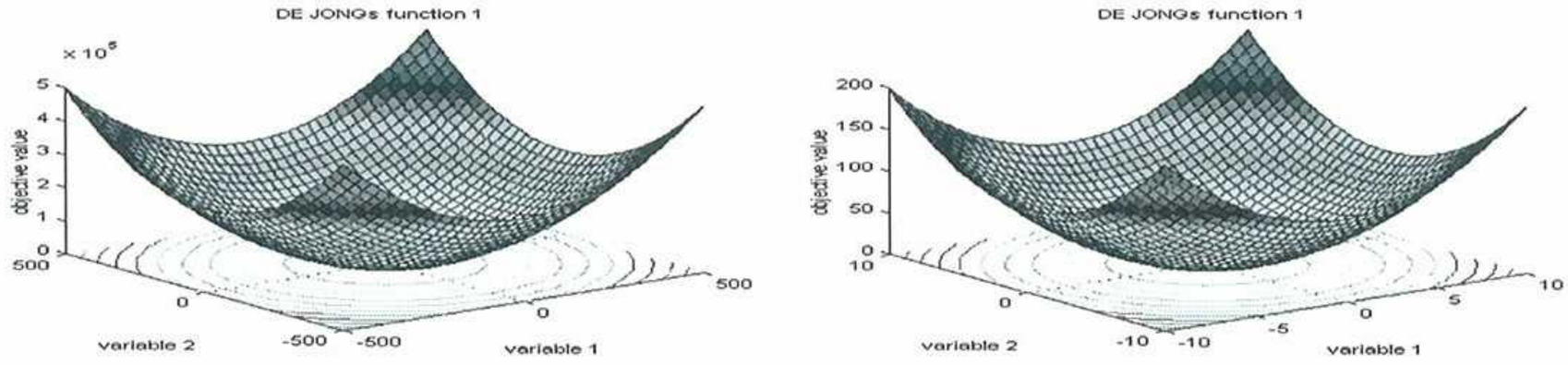


Figure 4.1.1 First De Jong Function

4.1.2 Second De Jong Function

This 2 parameter function is known to be problematic to solve. Its minimum is located at $f_2(\underline{1}) = 0$, its formulation is given in Equation 4.1.2 and its graphical representation in Figure 4.1.2.

$$f_2(\underline{x}) = 100 - \left(x_1^2 - x_2\right)^2 + \left(1 - x_1{}^2\right) \quad \text{with} \quad x_j \in [-2.048, 2.048] \quad \text{Equation 4.1.2}$$

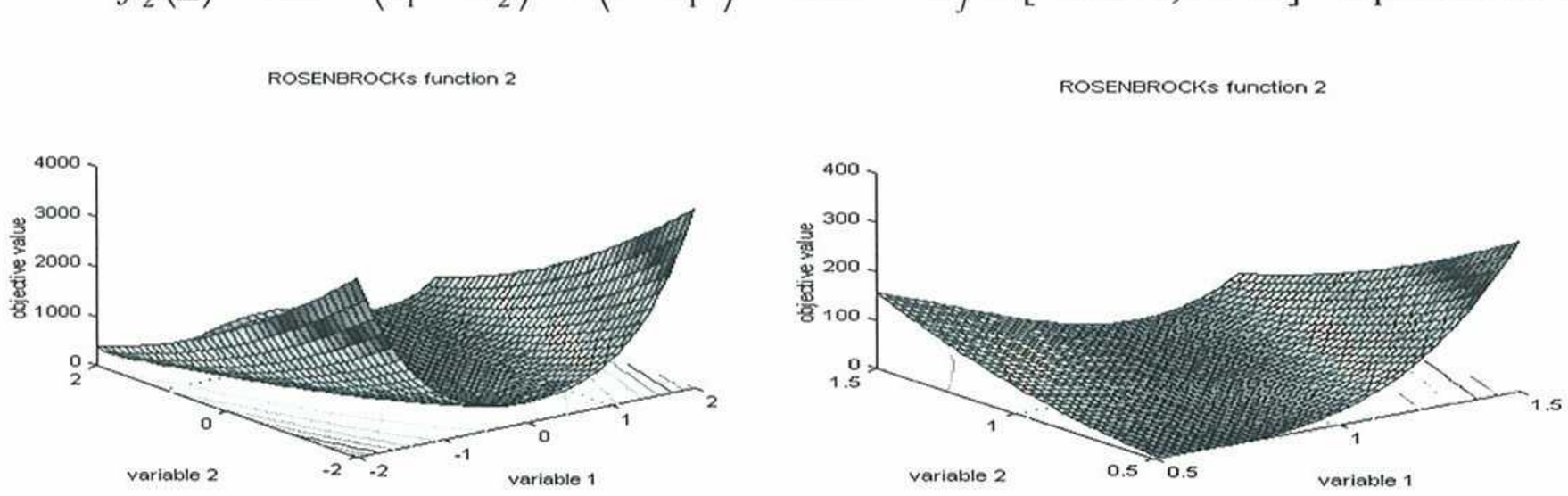


Figure 4.1.2 Second De Jong Function

4.1.3 Zimmermann Function

The function is given by the Equations 4.1.3, 4.1.4, and 4.1.5 and finding the minimum poses a difficult problem since the minimum, $f_3(7,2)=0$, is located at the corner of the constrained region defined by the 3 equations. This means that the algorithm has to be especially careful in the evaluation of infeasible solutions. Visualisations of the function and constraints are shown below. The point where all 3 functions overlap and Zimmermann is at a minimum, is given by the point at x_1=7 and x_2=2 on the right hand figure.

$$f_3(\underline{x}) = 9 - x_1 - x_2 \quad x_j > 0 \quad j = 1,2 \qquad \text{Equation 4.1.3}$$

$$(x_1 - 3)^2 + (x_2 - 2)^2 \leq 16 \qquad \text{Equation 4.1.4}$$

$$x_1 . x_2 \leq 14 \qquad \text{Equation 4.1.5}$$

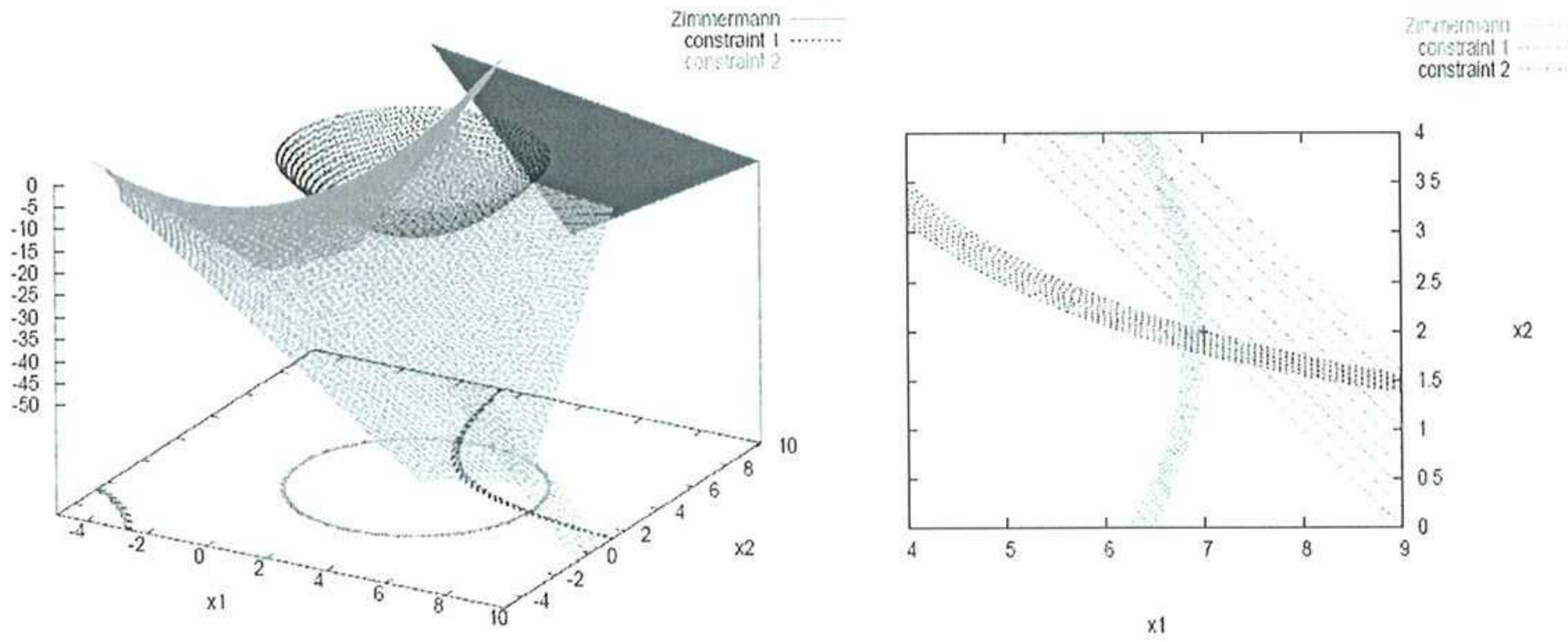


Figure 4.1.3 3D Zimmermann visualisation with a plan view.

4.2 Optima DE1

4.2.1 Optima DE1 Validation

When using the same sized population (*NP*), crossover parameter (*CR*) and differential variation amplification factor (*F*), the results, averaged over 10 trials, are shown in Table 4.2.1 below for the two versions of DE1. Given that this algorithm is stochastic, it can be seen that the number of function evaluations (*nfe*) required to reach the minimum for Optima DE1 is sufficiently similar to the DE1 proper. Having shown that Optima DE1 works in a very similar way to the original DE1 version, we can now be more certain that the two different formulations for the crossover process produce a similar performance.

	DE1				Optima DE1			
Function	NP	F	CR	nfe	NP	F	CR	nfe
De Jong 1	10	0.5	0.3	490	10	0.5	0.3	598
De Jong 2	6	0.95	0.5	746	6	0.95	0.5	761
Zimmermann	10	0.8	0.5	1559	10	0.8	0.5	1428

Table 4.2.1 Optima DE1 Results

4.2.2 Optima DE1 Best Settings

The parameters F and CR will now be varied between $0.1 \leq F \leq 1.0$ and $0.0 \leq CR \leq 1.0$ and tested on the same population, in an attempt to establish the best settings for DE1 over the range of test functions in Section 4.1. It is hoped then that these settings will give some indication of values that will also be the best or close to the best for further applications of the algorithm on other problems.

The matrices below are formed from the variation of F and CR. The left matrices contain the number of times the algorithm converged at a certain setting for the given test function out of 10 test runs. The right matrix provides the nfe taken to converge[15] or reach the halting criterion of a maximum 5000 nfe for 10 runs.

Following a criterion of only accepting settings which yield 90% convergence or above, we can "grey-out" the unacceptable entries in the matrix, leaving the acceptable results in the white boxes. By transferring this coloured template onto the right matrix, we can see which settings yielded acceptable results with the minimum nfe performed, which can therefore considered the best settings.

Function	Number of Times Converged out of 10											No. of Functn Evaluatns to Convergence Avg over 10 with 5000 NFE Halting Criterion										
		F=0.1	F=0.2	F=0.3	F=0.4	F=0.5	F=0.6	F=0.7	F=0.8	F=0.9	F=1.0		F=0.1	F=0.2	F=0.3	F=0.4	F=0.5	F=0.6	F=0.7	F=0.8	F=0.9	F=1.0
DEJONG1	CR=0.0	0	0	0	0	0	0	0	0	0	0	CR=0.0	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
	CR=0.1	2	9	10	10	10	10	10	10	10	3	CR=0.1	1150	1442	1435	1748	1802	1953	2193	2387	2645	2845
	CR=0.2	2	8	10	10	10	10	10	10	10	6	CR=0.2	750	889	913	1044	997	1133	1229	1343	1488	1425
	CR=0.3	1	7	9	10	10	10	10	10	10	7	CR=0.3	560	569	648	687	738	870	886	1044	1056	1794
	CR=0.4	1	6	10	9	10	10	10	10	10	8	CR=0.4	520	503	553	617	627	702	771	872	992	1471
	CR=0.5	0	4	9	9	10	10	10	10	10	8	CR=0.5	5000	405	468	520	639	654	753	783	923	1026
	CR=0.6	1	1	8	10	10	10	10	10	10	9	CR=0.6	460	280	484	505	541	607	655	784	821	872
	CR=0.7	0	0	5	9	9	10	10	10	10	8	CR=0.7	5000	5000	388	498	541	567	651	771	870	984
	CR=0.8	0	1	4	6	10	10	10	10	10	9	CR=0.8	5000	430	465	455	477	562	658	699	831	1057
	CR=0.9	0	0	0	4	8	10	10	10	10	8	CR=0.9	5000	5000	5000	480	531	538	647	709	844	1559
	CR=1.0	0	0	0	0	3	5	6	8	10	0	CR=1.0	5000	5000	5000	5000	490	518	583	793	914	5000
		F=0.1	F=0.2	F=0.3	F=0.4	F=0.5	F=0.6	F=0.7	F=0.8	F=0.9	F=1.0		F=0.1	F=0.2	F=0.3	F=0.4	F=0.5	F=0.6	F=0.7	F=0.8	F=0.9	F=1.0
DEJONG2	CR=0.0	0	0	0	0	0	0	0	0	0	0	CR=0.0	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
	CR=0.1	0	0	0	0	0	0	0	0	0	0	CR=0.1	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
	CR=0.2	0	1	1	1	3	0	1	1	0	0	CR=0.2	5000	3420	4140	4080	4223	5000	4640	4200	5000	5000
	CR=0.3	0	0	1	4	8	9	10	9	7	2	CR=0.3	5000	5000	3080	2548	2417	3859	3887	3502	4260	4600
	CR=0.4	0	0	2	4	5	9	10	10	10	2	CR=0.4	5000	5000	1640	1725	1988	1897	2585	2585	2597	3120
	CR=0.5	0	1	1	6	3	8	10	10	10	1	CR=0.5	5000	860	520	1243	1137	1613	1558	2055	1867	2030
	CR=0.6	0	0	0	1	6	9	9	10	10	2	CR=0.6	5000	5000	5000	1650	768	1120	1049	1283	1340	2800
	CR=0.7	0	0	0	2	4	8	10	10	10	2	CR=0.7	5000	5000	5000	645	735	919	1220	1006	1125	2235
	CR=0.8	0	0	1	1	5	6	9	10	10	4	CR=0.8	5000	5000	500	560	646	790	806	978	874	1690
	CR=0.9	0	0	0	2	5	6	7	10	10	2	CR=0.9	5000	5000	5000	410	518	568	1306	745	740	1025
	CR=1.0	0	0	0	1	1	5	3	9	9	0	CR=1.0	5000	5000	5000	330	320	404	513	731.11	748	5000
		F=0.1	F=0.2	F=0.3	F=0.4	F=0.5	F=0.6	F=0.7	F=0.8	F=0.9	F=1.0		F=0.1	F=0.2	F=0.3	F=0.4	F=0.5	F=0.6	F=0.7	F=0.8	F=0.9	F=1.0
ZIMMERMANN	CR=0.0	0	0	0	0	0	0	0	0	0	0	CR=0.0	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
	CR=0.1	0	0	0	0	0	0	0	0	0	0	CR=0.1	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
	CR=0.2	0	1	0	0	0	0	0	0	0	0	CR=0.2	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
	CR=0.3	1	2	8	8	10	9	8	8	6	0	CR=0.3	1920	2225	2936	3284	2708	3553	3705	4293	3960	5000
	CR=0.4	2	2	6	7	9	9	9	10	10	0	CR=0.4	855	1765	1825	2014	1919	2338	2519	2703	3001	5000
	CR=0.5	0	1	5	10	10	10	9	10	10	1	CR=0.5	5000	980	1466	1352	1546	1760	1834	2052	2285	2110
	CR=0.6	0	0	3	8	10	10	9	10	10	1	CR=0.6	5000	5000	1137	1169	1352	1227	1512	1659	1884	1900
	CR=0.7	0	0	3	4	7	9	8	9	10	1	CR=0.7	5000	5000	887	985	1017	1160	1284	1456	1490	1820
	CR=0.8	0	0	0	4	8	9	9	9	10	0	CR=0.8	5000	5000	5000	890	953	1032	1216	1254	1379	5000
	CR=0.9	0	0	0	3	6	7	10	10	8	0	CR=0.9	5000	5000	5000	927	830	1009	2143	1179	1329	5000
	CR=1.0	0	0	0	0	0	3	8	10	10	0	CR=1.0	5000	5000	5000	5000	5000	937	1254	1157	1364	5000

Table 4.2.2 DE1 Best Settings

The best settings are marked in red and it can be seen for the DeJong 1 Function that the best settings are in the low range $F=0.3$ and $CR=0.5$. For Dejong 2, $F=0.8$ and $CR=0.1$ yield the best solution and on the Zimmermann function $F=0.6$ and $CR=0.8$ should be chosen. However, for all 3 functions, high valued settings for both parameters can be seen to yield results with higher convergence frequencies and efficiency. In the testing for Problems 1, we will test at $F=0.3$ and $CR=0.5$, $F=0.85$ and $CR=0.7$[16]and $F=0.9$ and $CR=1.0$ to see whether the results conform to producing the best results for high F and CR values. In Problem 2, DE1 with settings F=0.3 and CR=1.0 is tested in the place of DE1 with F=0.85 and CR=0.7.

---

[15] If, for a certain setting, 7 out of 10 runs converged, the average of the 7 converged runs is given in the table. 5000 is displayed if there were no converged runs at all.

[16] This middle setting was chosen to allow easy comparison with results for similar problems in [5] and provide a mid-range test setting.

It should be noted that on comparing values in Table 4.2.1 with those in Table 4.2.2 for the same F and CR settings, the values in this section are higher. This is solely due to the initial population that was used, which in this section was held constant during all of the tests.

4.3 CMA

Whenever ES-CMA is used in this paper, a (10+10) type ES is employed when no recombination is chosen and a (10/10+10) is used when recombination is selected. The latter involves 2 parents in the recombination process to produce 1 child and the 5 children are each CMA mutated twice to form 10 individuals. These 10, combined with the 10 parents from the previous generation, form the selection population for the next generation and a Best Individuals selection method is used for the environmental selection. A scaling of this is applied in Problem 2 where populations of 30 are used.

4.3.1 CMA Comparison

Method / **Problem**	Method 1 DE1	Method 2 ES-CMA
De Jong 1	766	501
De Jong 2	979	605
Zimmermann	1518	1666*

Table 4.3.1 CMA Comparison with DE1 using 3 Test Functions
* Indicates that convergence only occurred 5 times out of 10.

A comparison of CMA's performance with that of DE1 (using the same settings for DE1 as in Table 4.2.1) was done using the 3 functions listed above and a constant initial population. The values above are once again averaged over 10 runs. The results indicate that CMA has the potential to out-perform DE1 on certain problems and is worth investigating for use on Problems 1 and 2. The ES-CMA used here is without recombination and using the 01 settings described below.

4.3.2 CMA Best Settings

The following tests were done to help determine the best ES-CMA combination to proceed with when tackling Problems 1 &2. A (10/10+10) ES-CMA and a (10+10) ES-CMA, as described in Section 4.3, were used with and without recombination selected, respectively.

The recombination method is a variable here, with either no recombination (NoRecomb), Weighted Average (WA) or Crossover (CR) being the choices. As well as this, suggestions for the Strategy variables c_σ, c_c (the latter two being both represented by c in the table below), c_{cov} and d (see Section 2.4 for definitions) were found in three papers by the authors of CMA in the years 1996 [7], 1997 [26] and 2001 [27]. The suggestions for the sets of Strategy variable values will be referred to by the year of the paper, namely 96, 97 and 01. The values given for n=2 will be used for the two parameter functions De Jong 2 and Zimmermann. The n=3 values will be used for De Jong 1 testing.

				n= 2			n= 3		
	96	**97**	**01**	**96**	**97**	**01**	**96**	**97**	**01**
c	1/(n^0.5)	1/(n^0.5)	4/(n+4)	0.707	0.707	0.667	0.577	0.577	0.5711
ccov	2/n^2	2/(n^2+n)	2/(n+(2^0.5))^2	0.5	0.333	0.17	0.222	0.167	0.103
d	n	n^0.5	(n+8)/4	2	1.414	2.5	3	1.732	2.75

Table 4.3.2 Suggested Strategy Variable Settings by Authors

The results from varying both the recombination method and the Strategy variable settings are shown in the next two figures, with data presented in the same manner as Section 4.2.2. Note the greyed-out 01 result for the Zimmermann function; it did not meet the 90% or above convergence criteria but was the only CMA setting to converge at all.

	CMA+NoRecomb			CMA + WA			CMA + CR		
	96	97	01	96	97	01	96	97	01
DJ1	10	10	10	10	10	10	10	10	10
DJ2	4	10	10	0	0	4	0	0	0
Zimmermann	0	0	2	0	0	0	0	0	0

Figure 4.3.3 CMA best settings no. of times converged out of 10.

	CMA+NoRecomb			CMA + WA			CMA + CR		
	96	97	01	96	97	01	96	97	01
DJ1	662	614	542	289	261	327	480	506	453
DJ2	2023	2123	894	5000	5000	2825	5000	5000	5000
Zimmermann	5000	5000	925	5000	5000	5000	5000	5000	5000

Figure 4.3.4 CMA best settings nfe to converge or meet halting criterion

The overall best setting for CMA is shown to be CMA with no recombination for the 01 values but when they do converge, CMA+WA and CMA+CR show better performance as can be seen for the De Jong 1 function. CMA+NoRecomb, CMA+WA and CMA+CR, all with the 01 settings, will be used for Problem 1.

4.3.3 CMA Settings Suggestions

Through inspection of the three results tables in Section 4.3.2, some trends were observed:

i The damping parameter d for the 01 settings, which showed the best performance, were either the highest or close to the highest relative to the other suggested values. We recall from Section 2.4 that d controls how quickly the step size reduces. Higher values mean slower reduction in step size.

ii The c and c_{cov} values for the 01 settings are always smaller than the other 2 sets of suggested values. We recall that smaller c values means longer accumulation time and more correlation information is incorporated and low values of c_{cov} means slower, increased stability in the solution and more use of correlation information.

These two observations seem to suggest that slower rates, either slower reduction in step size or longer accumulation times before changes are made, help CMA´s performance over the range of test functions. We should also note that WA has a blending effect on the population and CR has a disruptive effect which may disturb the complicated adaptation process of CMA and cause it not to converge on the more difficult De Jong 2 and Zimmermann functions. In the case of the Zimmermann function, the location of the global optimum being near the boundary or the feasible region makes convergence even more unlikely. In the case of De Jong 1 however, the simple and smooth topology seems to benefit from these ´disturbances´ as can be seen from the improved performance when recombination is used.

With this in mind, I propose two strategies and suggest settings accordingly. The suggested settings will be referred to as CMA-A and CMA-B

4.3.3.1 CMA-A – This follows from the above discussion which suggest that the 01 values may have performed better on the whole due to lower c and c_{cov} values and its higher d settings. Therefore a simplistic approach of using even lower c and c_{cov} settings along with an increased d value gives the following trial settings:

	n=2	n=3
c	0.65	0.55
c_{cov}	0.15	0.1
d	2.75	3.2

Table 4.3.5 CMA-A Settings

4.3.3.2 CMA-B – The second strategy comes from considering the parameters c_σ and c_{cov} separately. This strategy postulates that from the above data, we have seen that slower rates of adaptation of the Covariance matrix and more accumulation and use of correlation information seem to benefit CMA and so low values of c_c and c_{cov} are retained (similar to CMA-A reasoning). However, the global step size can be adapted at a much quicker rate since it depends on a lot fewer variables than C, so c_σ should be set higher than c_{cov}. The damping parameter is set to a high value as before so as not to decrease the step size too quickly which impairs the algorithms ability to explore the search space. Following these criteria and other criteria set in Section 2.4, the following settings are proposed:

	n=2	n=3
c_σ	0.72	0.6
c_c	0.65	0.55
c_{cov}	0.15	0.1
d	2.75	3.2

Table 4.3.6 CMA-B Settings

4.3.3.3 CMA-A&B Validation – By carrying out a test in exactly the same manner as in Section 4.3.2, the following results were obtained:

	CMA + No Recomb		CMA + WA		CMA + CR	
	A	B	A	B	A	B
DJ1	10	10	10	10	10	10
DJ2	10	10	6	2	0	0
Zimmermann	5	3	0	0	0	0

Table 4.3.7 CMA-A&B no. of times converged out of ten.

	CMA + No Recomb		CMA + WA		CMA + CR	
	A	B	A	B	A	B
DJ1	562	587	313	312	506	458
DJ2	953	869	2992	3858	5000	5000
Zimmermann	860	850	5000	5000	5000	5000

Table 4.3.8 CMA-A&B nfe to converge or meet halting criterion

The results show that CMA-B performs better out of the two strategies on the whole, in some cases only slightly better. CMA-A achieves better results when combined with WA recombination and even though CMA-B with no recombination converges quicker for the Zimmermann function, we note the equivalent CMA-A scheme converges more times and is therefore more reliable. It is therefore hard to say which strategy is better and when comparing them to the authors' suggested settings. Strategy CMA-B displays the best performance for De Jong 2 and Zimmermann functions when used with no recombination and CMA-A and the 01 values follow close behind. The best combination for De Jong 1, however, is still CMA+WA with the 97 values and CMA-A and B are not expected to show great performance when used with recombination since they were modelled as an extension of the ethos of the 01 values which themselves did not perform well when used with recombination. CMA with the 01 values will from now on be referred to as CMA 01. The latter, along with CMA+WA and CMA+CR will be tested on Problem 2.

5 Inverse & Direct optimisation

A summary of the previous tests carried out with the various algorithms on different problems, and also work to be presented below, are summarised in the work matrix below. Results from testing done in Section 4.2 will be used on both Problems 1 and 2. Work done in Sections 4.3.2 and 4.3.3 will be used exclusively in Problems 1 and 2, respectively. Other data for the two optimisation problems are held in Appendix A and include graphs of the global step size vs generation no. for the CMA tests; m,p and t variation with no. of generations for both optimisation problems; m,p and t summed then normalised percentage difference with target coordinate values at generation 59 for Problem 1 and CL and CD values for Problem 2.

Problem	Algorithm	DE1				ES-CMA NoRecomb					ES-CMA+WA/CR				
		F0.3CR0.5	F0.85CR0.7	F0.9CR1.0	F0.3CR1.0	96	97	1	A	B	96	97	1	A	B
Test Functions	Dejong 1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Dejong 2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Zimmerman	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Problem 1	Inverse Optimization	✓	✓	✓				✓					✓		
Problem 2	Direct Optimization	✓		✓	✓			✓	✓	✓					
✓	Signifies the algorithm settings are used to solve the optimisation problem														

Table 5.1 Work Matrix

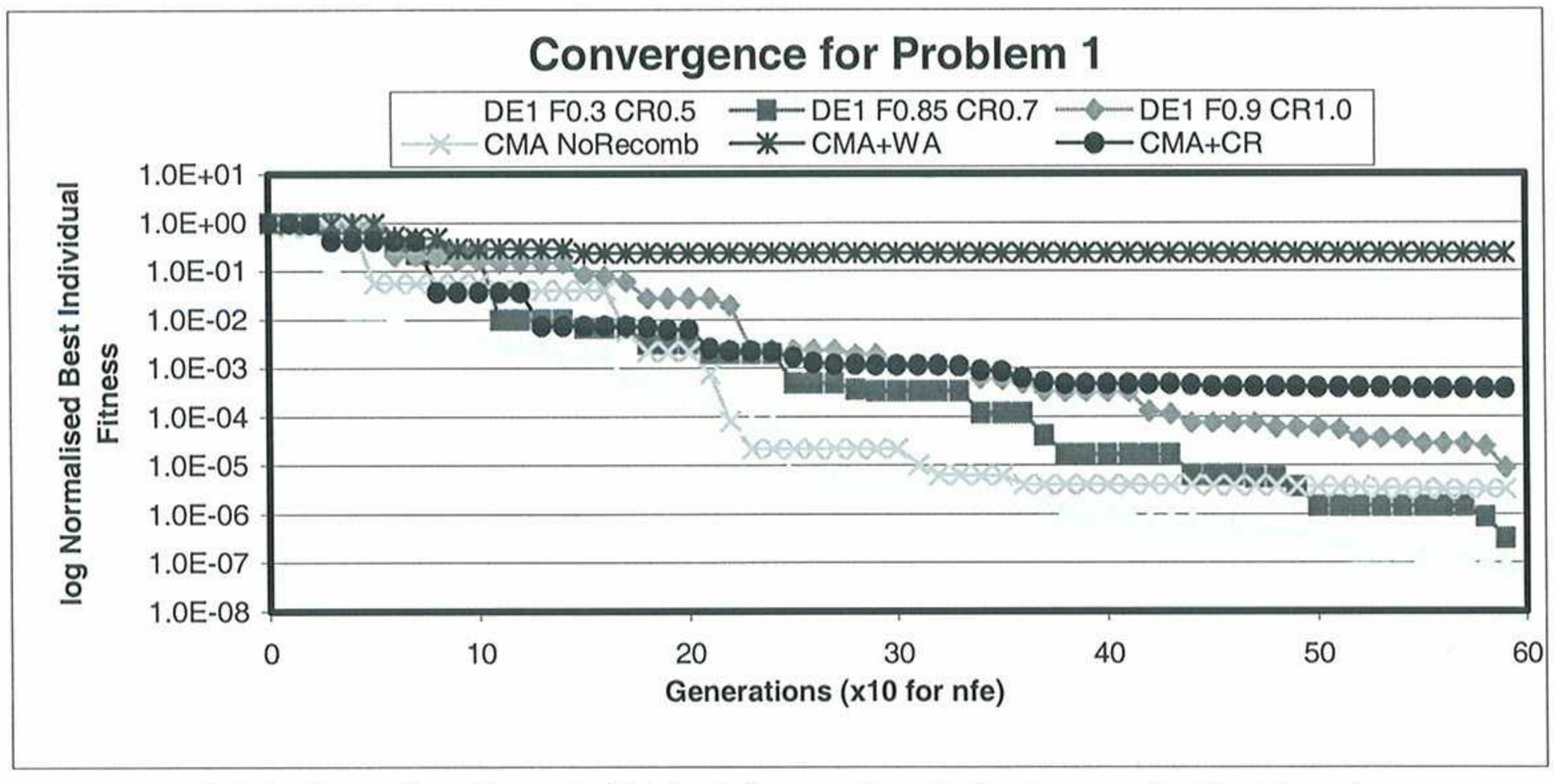
5.1 Problem 1 – Inverse Optimization Results

In Figures 5.1.3 and 5.1.4, the differences in the results between all of the DE1 runs are visually indistinguishable and DE1 with F0.9 and CR1.0 represents all three DE1 settings results. The same applies for CMA with no recombination and CMA+CR and even though the latter does not converge, their pressure and aerofoil profiles are very similar. CMA+WA is also presented in the figures since it produces very different values to the other schemes. The 01 values are used when testing with CMA.

Design Parameter	Target	DE1			ES-CMA 01 Values		
		F0.3CR0.5	F0.85CR0.7	F0.9CR1.0	NoRecomb	WA	CR
m, p, t values at Generation 59	0.02000 0.40000 0.15000	0.02000 0.39994 0.15002	0.01999 0.40018 0.14998	0.02003 0.40020 0.15013	0.02002 0.39892 0.15005	0.01438 0.12276 0.15336	0.01993 0.41274 0.14993
Σ(MOD(CoordsDifference))	0	8.189E-05	1.046E-04	6.411E-04	2.825E-04	7.134E-02	2.330E-03
Generations to Converge* (1e-5)		25	44	59	32	Not Converged	Not Converged
Fitness at 59 Generations		8.71E-08	3.177E-07	8.91E-06	3.29E-06	2.348E-01	3.748E-04

*Value equals number of generations after the initial generation to converge

Table 5.1.1 Comparison between target and results at 59 generations for Problem 1



5.1.2 Normalised best individual fitness for all 6 schemes for Problem 1

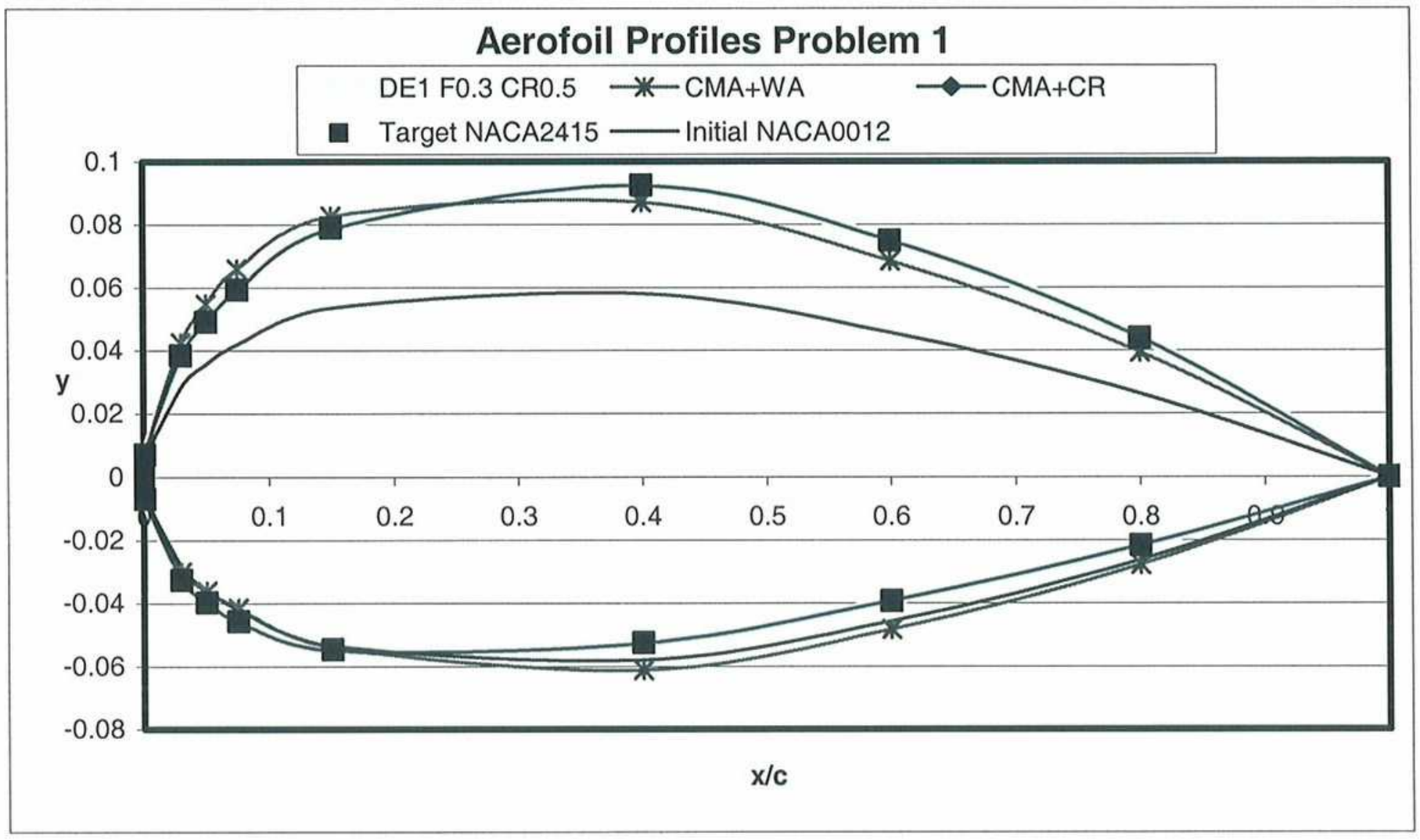


Figure 5.1.3 Aerofoil Profiles at generation 59 for Problem 1

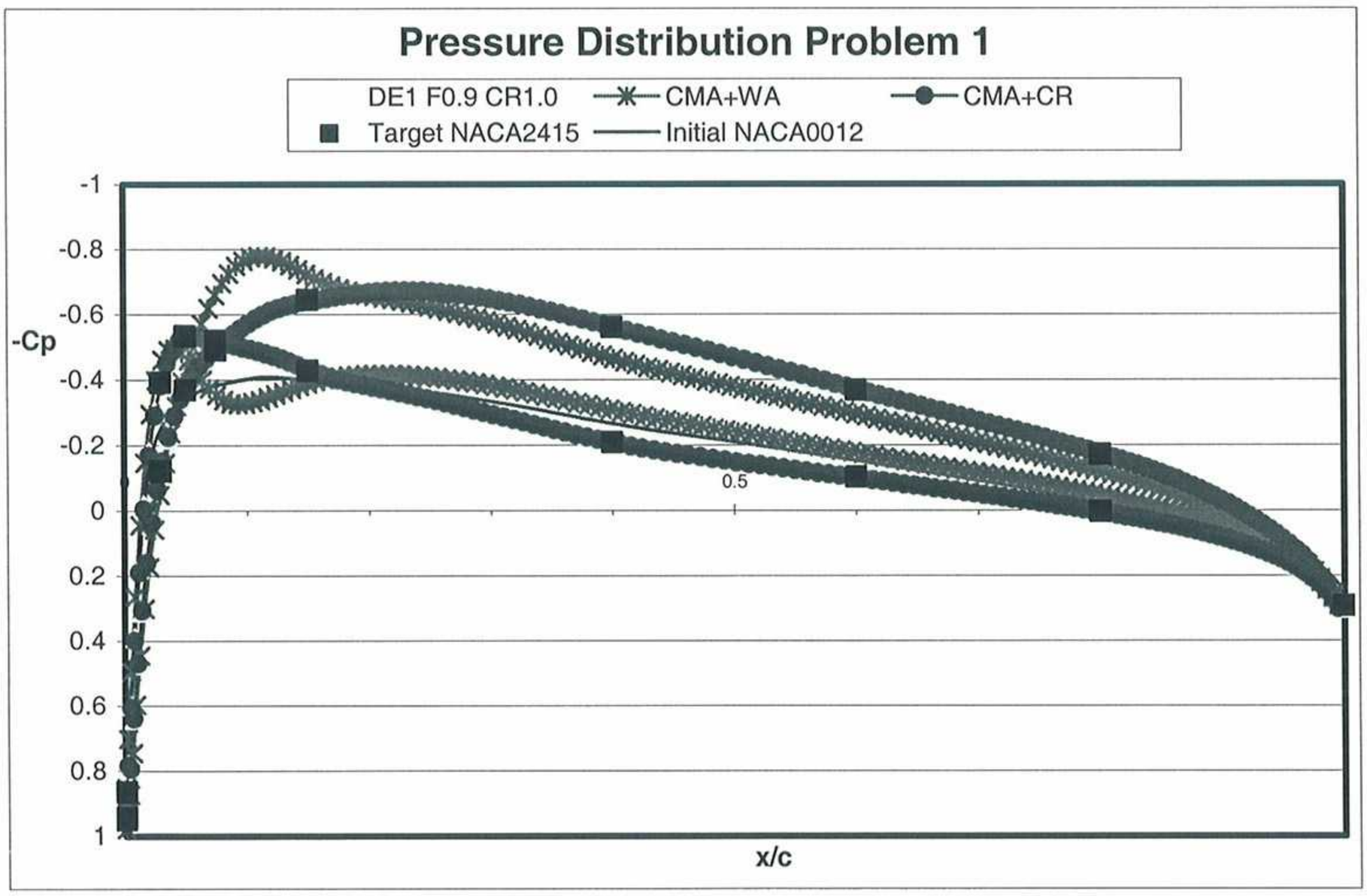


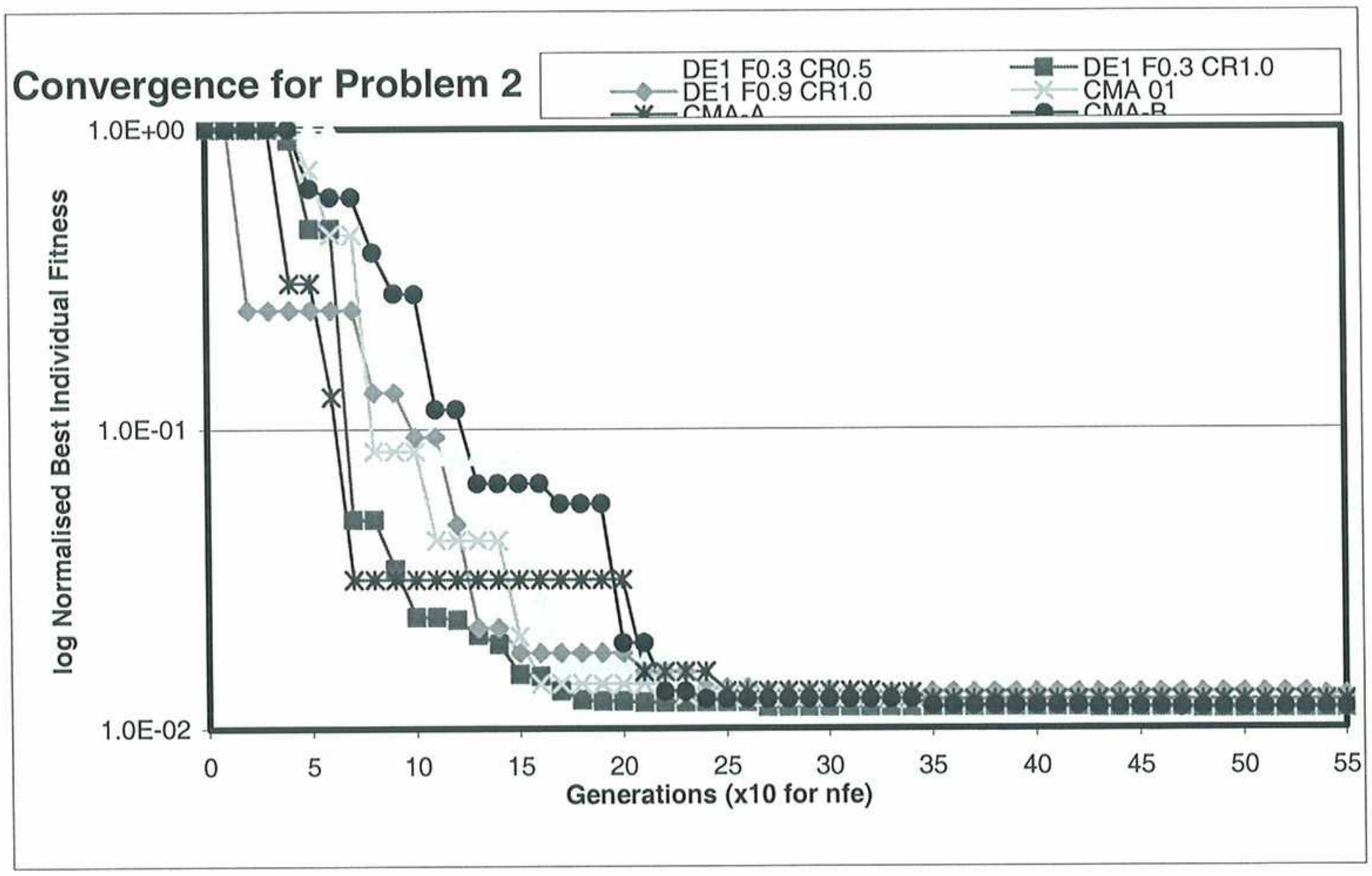
Figure 5.1.4 Pressure distributions at generation 59 for Problem 1

5.2 Problem 2 – Direct Optimisation Results

The aerofoil profiles and pressure distributions from the optimisation process are presented below with results from the DE1 ad CMA schemes on separate graphs for clarity. The results for both schemes have also been presented on a single graph in Appendix A for the aerofoil profile and the pressure distributions, on Figures A.2.5 and A.2.6, respectively. Figures A.2.7 and A.2.8 in Appendix A show the C_L and C_D variations with the number of generations. Testing was done using a population of 30 to increase diversity.

Design Parameter	Target	DE1			ES-CMA 01 Values		
		F0.3CR0.5	F0.9CR1.0	F0.3CR1.0	01	A	B
m	values at generation 55	0.01834	0.01999	0.01811	0.02062	0.02053	0.01783
p		0.42635	0.40018	0.44438	0.25137	0.25455	0.46431
t		0.13422	0.14998	0.10630	0.11436	0.11557	0.10872
CL	0.19	1.90012E-01	1.89970E-01	1.90003E-01	1.90000E-01	1.90000E-01	1.90000E-01
Cl&Cltarget difference		1.1998E-05	2.9531E-05	2.6667E-06	2.98674E-07	2.27826E-07	1.41078E-07
CD		2.36484E-02	2.31919E-02	2.13753E-02	2.29884E-02	2.28864E-02	2.15661E-02
Normalised Fitness at 55 Generations *Value equals number of generations after the initial generation		1.29135E-02	1.28563E-02	1.15838E-02	1.24315E-02	1.23743E-02	1.16596E-02

Table 5.2.1 Comparison of results from different algorithms tested on Problem 2 at Generation 55



5.2.2 Normalised best individual fitness for 6 schemes for Problem 2

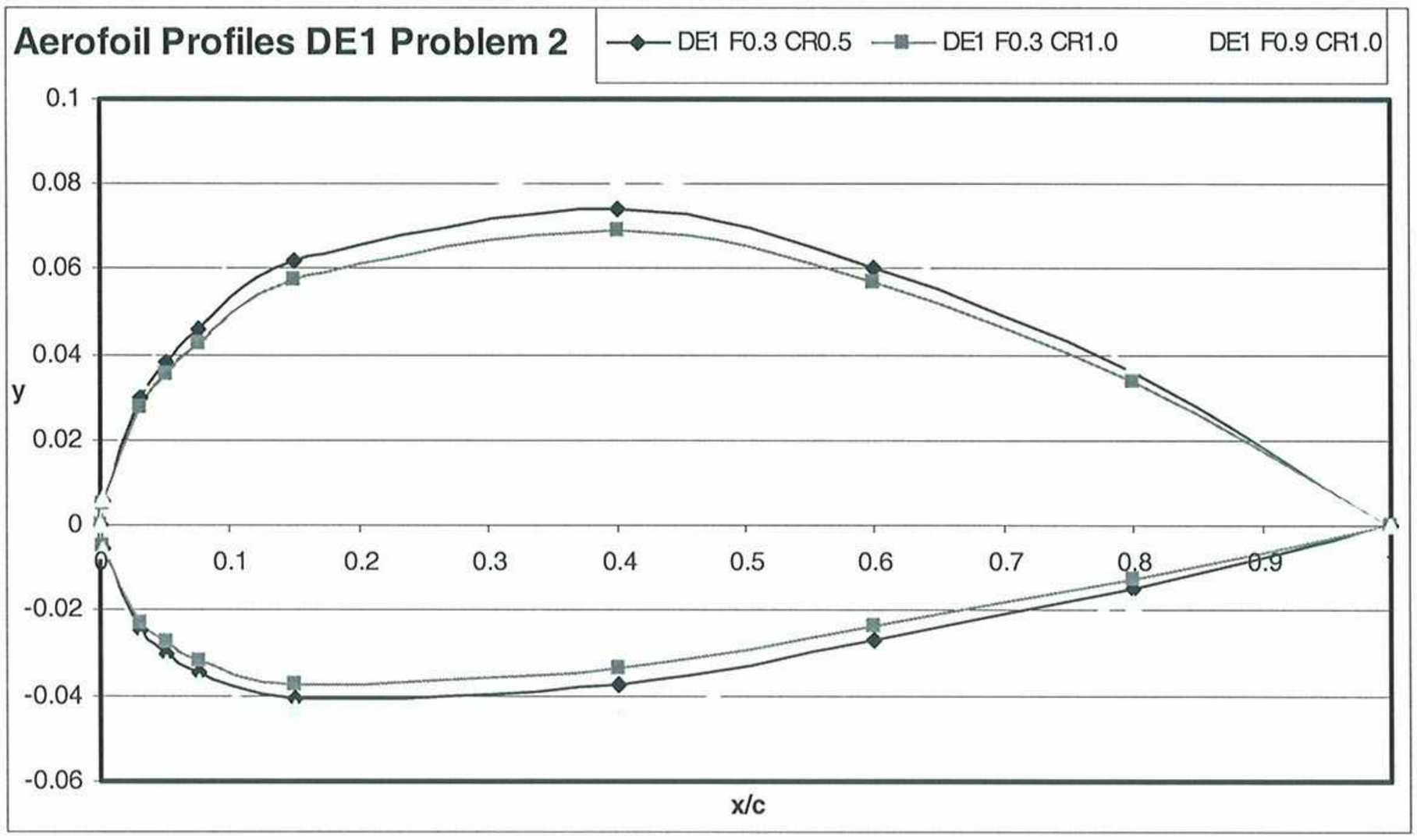


Figure 5.2.3 Aerofoil profiles at generation 59 for DE1 tests on Problem 2

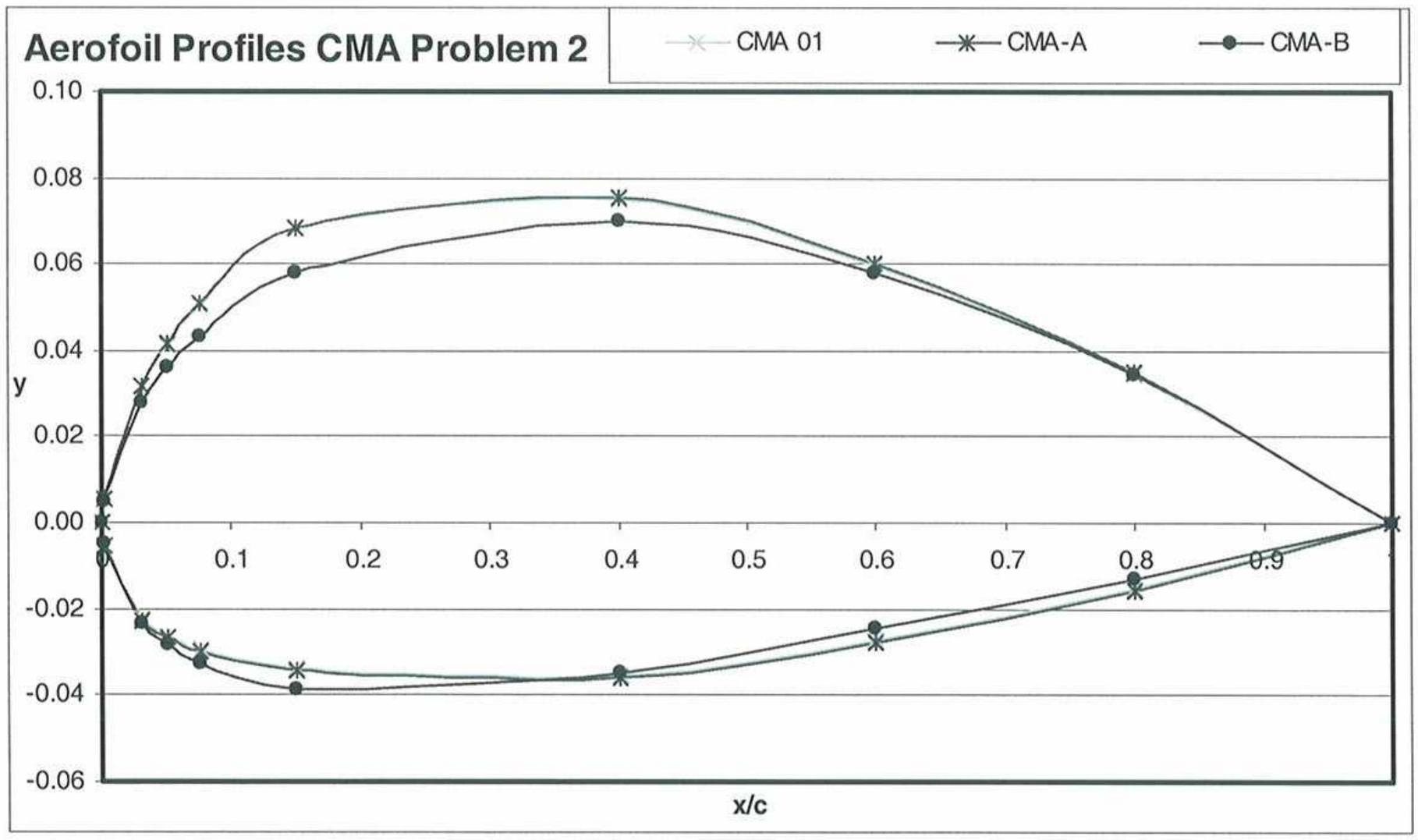


Figure 5.2.4 Aerofoil profiles at generation 59 for CMA tests on Problem 2

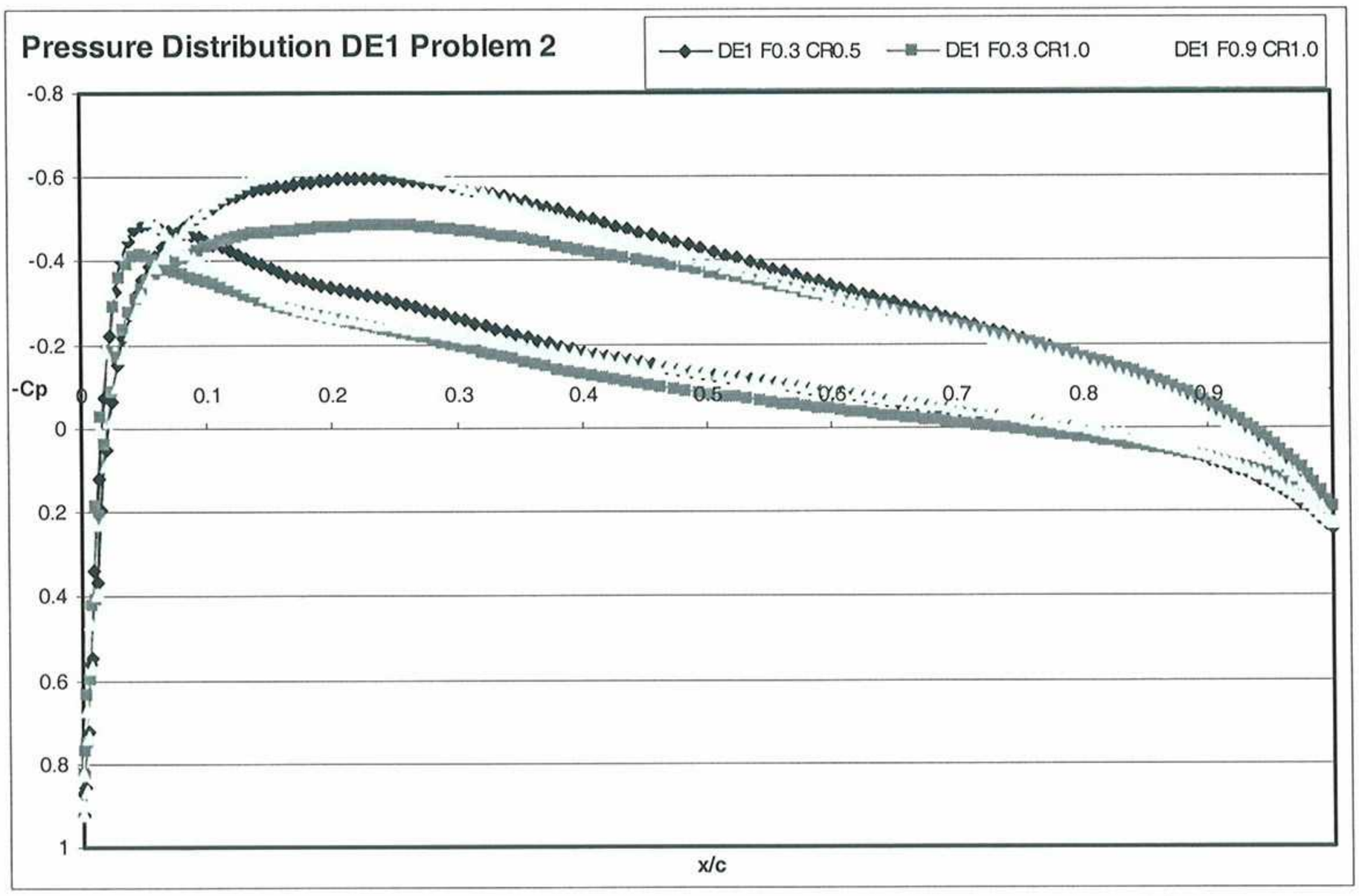


Figure 5.2.5 Pressure distributions at generation 55 for DE1 tests on Problem 2

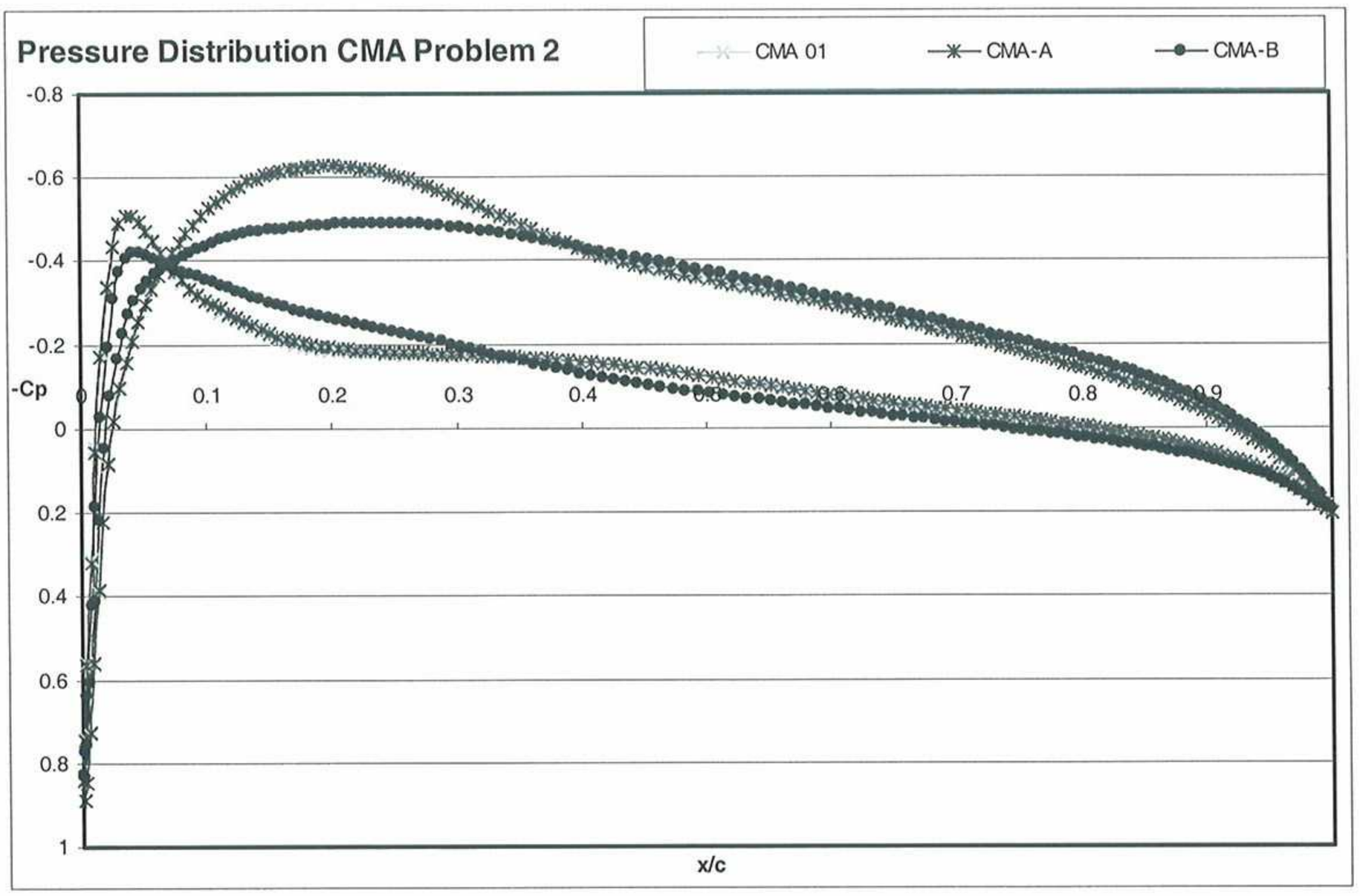


Figure 5.2.6 Pressure distributions at generation 55 for CMA tests on Problem 2

6 Evaluation & Conclusions

6.1 Standard Parametric Optimisation Objective Functions

A summary of the tests carried out on the three parametric optimisation functions in Section 4 is summarised below and the data is presented in the same format as in Chapter 4:

Algorithm / Problem		DE1				ES-CMA NoRecomb					ES-CMA+WA					ES-CMA+CR				
		F0.3CR0.5	F0.85CR0.7	F0.9CR1.0	F0.3CR1.0	96	97	1	A	B	96	97	1	A	B	96	97	1	A	B
Test Functions	De Jong 1	9	-	10	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	De Jong 2	1	-	9	0	4	10	10	10	10	0	0	4	6	2	0	0	0	0	0
	Zimmermann	5	-	10	0	0	0	2	5	3	0	0	0	0	0	0	0	0	0	0

Figure 6.1 No. of times converged out of ten

Algorithm / Problem		DE1				ES-CMA NoRecomb					ES-CMA+WA					ES-CMA+CR				
		F0.3CR0.5	F0.85CR0.7	F0.9CR1.0	F0.3CR1.0	96	97	1	A	B	96	97	1	A	B	96	97	1	A	B
Test Functions	De Jong 1	468	-	314	5000	662	614	542	562	587	289	261	327	313	312	480	506	453	506	458
	De Jong 2	520	-	748	5000	2023	2123	894	953	869	5000	5000	2825	2992	3858	5000	5000	5000	5000	5000
	Zimmermann	1466	-	1364	5000	5000	5000	925	860	850	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000

Figure 6.2 Nfe to converge or meet halting criterion

We can consider the results in two groups since De Jong 1 is a three parameter function and the same settings which are seen to be beneficial when used on it, behave differently when employed on the two parameter functions De Jong 2 and Zimmermann.

For De Jong 1, it can be seen that all of the tested methods and settings meet the 90% or above convergence criteria, with the exception of DE1 using F=0.3 and CR=1.0, which failed to converge at all within 500 generations. The best performing method for this function overall was the ES-CMA using Weighted Average recombination on the 97 settings. The latter settings for ES-CMA use relatively high values for c and c_{cov} and low damping which correspond to quick but less reliable adaptation of the evolution paths and the Covariance matrix. Also there is less use of correlation information and the global step size is reduced at a quicker rate, which therefore reduces the mutation step size more quickly. Alongside these effects, we also have the blending effect of Weighted Average recombination which reduces the diversity in the population. These quick change rates and small mutation steps are favoured for the simple topology of De Jong 1 where the algorithm does not have to explore large parts of the search space before reaching the easy to find optimum. The best performing DE1 settings for De Jong 1 are F=0.3 and CR=0.5 and follow similar ideology in that small mutation step sizes are used with F being small and these small changes are incorporated into approximately half of the parameter $\underline{u}$. On the other hand, the worst performing settings, from DE1 with F=0.3 and CR=1.0, also uses small mutation steps but since it uses a CR=1.0, the vector $\underline{u}$ becomes identical to $\underline{v}$ and the process basically amounts to mutation only. This means that if the initial population is not diverse and individuals lie far from the global optimum, the small mutation steps are not sufficient to help the algorithm converge.

For De Jong 2 and Zimmermann, the most effective and efficient algorithm was DE1 with F=0.9 and CR=1.0. This once again, amounts to a mutation only process, but with large mutation steps which allows a more expansive search of the feasible region in order to arrive at the harder to find global optima. With smaller mutation steps, the solution may become stuck and never reach the optimum. The best performing ES-CMA settings were the CMA-B values, used without recombination. This method uses slower but more reliable adaptation with more use of correlation information. The global step size is also decreased at a slower rate so that a larger amount of the search space for this more difficult optimisation problem function can be explored. The different time scale at which the C matrix, the global step size and their evolution paths can be adapted is taken into account and this consideration is shown to be important, with

these settings proving most efficient. ES-CMA used with recombination, which fared well in the last test, is shown to have major convergence problems with this more complicated topology. This algorithm was not designed to be used with such recombination methods and whereas their disruptive effects happened to have a beneficial affect on performance of the last function, for the more difficult test function, they disturb the intricate procedures used by CMA and hinder it from finding the optimum. It is in fact meaningless to use CMA with recombination since the evolution path information on which CMA depends so much, is greatly disturbed. When we then try to de-correlate the evolution path, we can only do so with respect to the last step since the evolution path from the parents can not be incorporated in a meaningful way. This may be the reason why it is the 97 values, with its quick adaptation procedure, that performs the best for De Jong 1 since, through its high c and c_{cov} values, it intends to use less correlation information in the first place.

We remember also that we are using the CMA two parameter values here which are tuned for quicker adaptation than the three parameter values since there is one less dimension or permutation to consider. We cannot, however, automatically assume that two parameter functions are easier to solve and automatically speed up the adaptation process to enhance efficiency. It is seen here that for the De Jong 2 and Zimmermann functions that the settings that achieve the best performance, are those that are closest to the three parameter settings, in particular CMA-B, which preserve the large step sizes, the slow adaptation rate and the higher use of correlation information, in order to reach the optimum.

6.2 Problem 1

For the inverse optimisation problem, DE1 with F=0.3 and CR=0.5 converges the quickest to the normalised fitness of 1e-05 and also to the lowest fitness after 59 generations, as can be seen by Figure 5.1.2. CMA with no recombination, using the 01 values, follows close behind in terms of efficiency. Figures 5.1.3 and 5.1.4 show only the DE1 F=0.3 and CR=0.5 results since the outcome of all the DE1 tests are very similar after 55 generations, with the total difference between their 19 coordinates being a maximum of only 5.6e-4. Similarly, the CMA+CR result is also representative of the CMA+NoRecomb run. CMA+WA however, proves to be the least robust and the most inefficient setting and its results differ significantly from the others. This is also shown very clearly on Figures A.1.1 to A.1.7 of the *m,p,t* and coordinate points variations in Appendix A. None of the CMA runs converge when used with recombination and this has been expected from the discussion in Section 6.1. The disruptive effect of the recombination process on CMA can be seen in Figure A.1.8, showing the global step size evolution: Crossover recombination disturbs the adaptation, hampering the necessary decrease in step size, so that the algorithm will never reach the optimum unless completely by chance. The CMA+WA step sizes increase significantly on more than one occasion but luckily manages to decrease, but does so too soon, as it reaches a local optima and its solution stagnates, as seen on Figure 5.1.2. Repeating the run is not expected to improve the result.

6.3 Problem 2

In Problem 2, the population size was increased to 30 to increase diversity without which the algorithms fail to converge. For this direct optimisation problem, it is once again a Differential Evolution algorithm that performs the best in terms of the normalised fitness reached at the halting criterion of 55 generations. The strategy CMA-B follows close behind with its corresponding normalised fitness being only 7.6e-3 larger than the most efficient setting of DE1, which is F=0.3 and CR=1.0, and both can be seen to converge quickly to their low fitness values on Figure 5.2.2. The latter also achieves the lowest drag whilst the former attains a C_L value closest to the target 0.19 value. The worst setting in terms of normalised fitness is DE1 with F=0.3 and CR=0.5. This may be due to the lower crossover probability setting which reduces the contribution from the trial vector in the individual $\underline{u}$ and means that there is slower

progression in the solutions from the initial population. The introduction of test setting F=0.3 and CR=0.1 was due to the limitation of values used in previous tests where either very low or very high mutation and crossover values were used. Using a mixed scheme of low mutation and high crossover provided results for a new hybrid which would be neither too slow in evolving, nor too extreme in its mutation process. Coincidentally, it proved the most efficient.

In Problem 2, many more optima, both local and global, are expected than in Problem 1, since although more constraints are applied in the form of a penalty function, the optimisation requirement is not to reach one specific aerofoil, and the problem is a lot less restrictive. The minimal optimum found for Problem 2 is reached by CMA-B and DE1 F=0.3 and CR=1.0 and is shown in Figure 6.3.1, although we can not be absolutely certain that it is the global optimum.

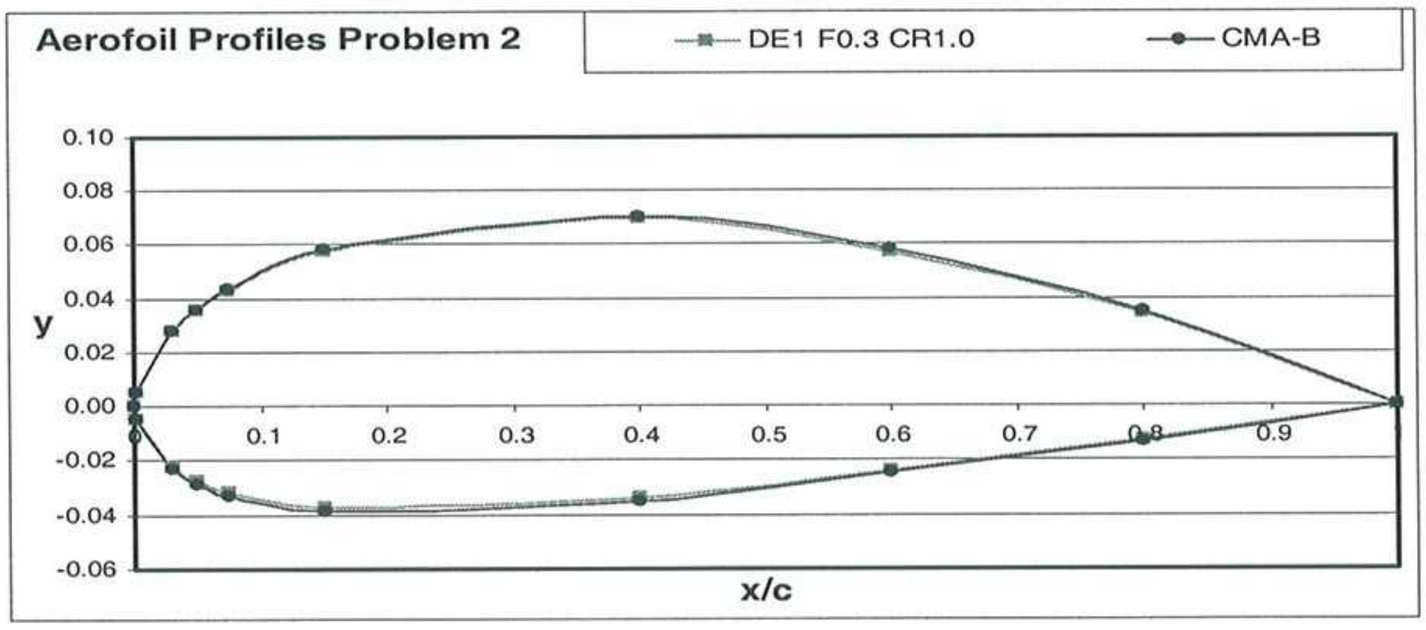


Figure 6.3.1 Problem 2, Best DE1 and ES-CMA results produce similar profile

We recall that the equation for generating trial parameters in DE1 consists of using three randomly selected vectors from the population. This therefore suggests that this method depends greatly on the convergence of the majority of the population towards an optimum, in order for the mutation step size to decrease sufficiently to obtain it. Otherwise, a random selection of an individual far away from the target optimum would pull the solution away from it. This could be the reason why the F=0.9 and CR=1.0 DE1 setting does not perform as well as DE1 with F=0.3 and CR=1.0 within the 55 generations. The former magnifies the importance of a large number of the population to be in the vicinity of an optimum, due to its large value increasing the sensitivity to the differential variation between two randomly chosen vectors. The low *F* value setting, which performed the best, is not as sensitive to the deviation of the population. However, it requires a large population and for the population to contain good starting individuals, placed not too far from the optimum, since it relies only on mutation with a small step size to escape the many local optima and reach the global ones. Whereas the DE1 settings produced three very different aerofoils and pressure distributions as shown in Figures 5.2.3 and 5.2.5, CMA 01 and CMA-A produced very similar *m, p* and *t* values, aerofoil profiles and pressure distributions, indicating the presence of a local optimum. The two strategies suggested in this paper yield lower normalised fitnesses than the author suggested 01 values, although the CMA-A result is only slightly lower. The slower adaptation of the strategy variables with the CMA-B setting once again proves to be more robust and efficient than its two CMA counterparts. Looking at the global step size evolution on Figure A.2.4, we can see that for the three CMA settings, the rate of decrease is similar after 16 generations, although CMA 01 which performed the worst out of the three, does not show any increases in step size to try to escape the local optima during its evolution.

It is also interesting to note on Figures A.2.1 to A.2.3, that the two best performing algorithms (CMA-B and DE1 F=0.3 and CR=1.0) evolved to give *m* and *p* values which are very similar to the worst performing algorithm (DE1 F=0.3 CR=0.5). It is only the *t* value of the latter that differs significantly from the other two schemes, doing so to the order of 3e-02. Figure A.2.3 shows that for all six schemes used in Problem 2, the *m, p* and *t* values start to fluctuate much less eratically after approximately 30 runs. A contribution to the poor performance of DE1 using

F=0.3 and CR=0.5, could be that the scheme jumps to a high t value on generation 23, and is not able to reduce it over the following 22 generations to reach the global optimum, due to its combination of a small mutation step and a low crossover probability.

6.4 Overall Summary of Optimisation Test Results

CMA provided close competition for DE1 and CMA-B consistently followed the best performing algorithm, both being DE1 but with different settings, closely on the two inverse problems. CMA-B was seen to be the most robust out of all of the settings tested in this paper and it is through simple consideration of each of the parameters separately that we draw on its benefits, so further consideration and investigation of the parameters for CMA has been shown to be very worthwhile. Recombination has been proven to have a detrimental effect on the robustness of the CMA schemes and unless certain information is known about the topology of the function to indicate their use is beneficial, this evolutionary operation disrupts the delicate and complicated CMA process and should not be used.

It has been confirmed through the two aerofoil optimisation problems that although high values for F and CR used in DE1 are not the most efficient, they are the most robust. This is because the high F value, when the problem is first started, allows the exploration of a large part of the feasible search space which improves its chance of finding the global optimum for more difficult problems, but decreases it efficiency for simple topologies. The DE1 algorithm, in accepting only the better of the two individuals $\underline{x}_{i,G}$ and $\underline{u}$, can be considered very 'greedy'. However, when using large F values, the increased sensitivity to differential variation between the vectors contained in the population means that for a decent sized population (10 or above), if even a small number of solutions are located near the global optimum and the majority of individuals lie near a local minimum, a scheme with a large F value is likely to increase the step size until the majority of the population converge on the global optimum and this behaviour helps the greedy DE1 algorithm from becoming trapped in local optima. In contrast, the low valued DE1 settings, whilst they might be the most efficient for certain problems, the results prove that they do not demonstrate these robust qualities.

Looking from the trends in the results and considering the theory, although it has not been possible to show this, it can be deduced that, for the majority of problems, if a test is run with no halting criterion on the number of function evaluations, the high F valued DE1 schemes would in general produce more accurate results than low F valued schemes due to its higher sensitivity as formulated in the differential operator for DE1.

In all problems, we have to establish either a priority between speed and robustness in an algorithm. It has been shown in the above discussion that whereas there is no single algorithm with a certain setting that proves most efficient, irrespective of the problem to be solved, there are schemes which display great robustness and they are typically methods which retain high mutation step sizes, meaning less efficiency when searching for an easy to find optimum, but greater endurance when searching for a hard to find optimum and helps the algorithm to resist being trapped in local optima.

One thing that has to be bared in mind when interpreting the results presented here, is that the time limitations of a four month project did not allow several test runs for Problems 1 and 2. The stochastic nature of the Evolutionary Methods requires further tests before definite trends can be observed. However, the results presented here can be interpreted as an indication of the behaviour of the schemes. Moreover, it has been shown that Evolutionary Algorithms are an effective method for solving optimisation problems but more runs should be done in order to eliminate stochastic variations and allow full interpretation of the results. A comparison should also be done between Evolutionary Methods and other approaches to determine the efficiency of these schemes and also to motivate enhancements to make them competitive when compared with other methods.

6.5 Model Simulation

The problem simulation using GiD and Tdyn requires further testing to improve accuracy as seen from the test results in Section 2.9.1. The best direction to take is either to further investigate structured mesh generation in GiD or alternatively, use another meshing program which can be easily automated. The use of a structured mesh will aid the turbulence model simulation by making it easier to meet the requirement of placing the first node inside the Tdyn defined log law region. However, CFD programs are very sensitive to the mesh used and mesh sensitivity analyses may have to be considered in order to evaluate the best type of meshing for aerofoil problems [28]. As it stands, the solution process of the turbulent aerofoil model takes roughly one day to solve and refined models are expected to take even longer. The alternative would therefore be to use a program such as XFOIL if only aerofoil problems are to be used.

6.6 Further Work & Suggestions

- The range of aerofoils that could be explored in this report was limited to the NACA four digit profiles. This was to reduce the degrees of freedom in the problem since the calculation process of the aerofoil simulation would take too long. More variety in the types of aerofoil that can be investigated could be achieved by using different representation of the aerofoils such as by using Breziér curves.

- As mentioned in Section 6.4, the different parameters in CMA should be further investigated to realise the full potential of the method.

- Section 6.5 concludes that the Tdyn model of the turbulent aerofoil should either be studied further in order to increase the accuracy of the calculation, or another method should be used to represent and simulate the model. One benefit of the operational setup with GiD and Tdyn however, is the option to solve coupled problems by activating the thermal or structural solvers in Tdyn. Another obvious benefit of using Tdyn over a program such as XFOIL is the opportunity to generate more complex aerodynamic geometry or even other shapes for optimisation. A brief study on the 3D simulation of a swept wing was carried out towards the end of this project using the same turbulence model and wall function as previously used in this report. A batch file is contained in Appendix C.7, which was written to set up the wing geometry and control volume, shown in Figure 6.6.1. The pressure, velocity and eddy viscosity results produced from these tests are shown in Appendix E. A suggestion for the objective function of the 3D problem would be to use the sweep or the planform shape as a variable in an inverse optimisation test and use the Collocation Method to approximate the necessary analytical results.

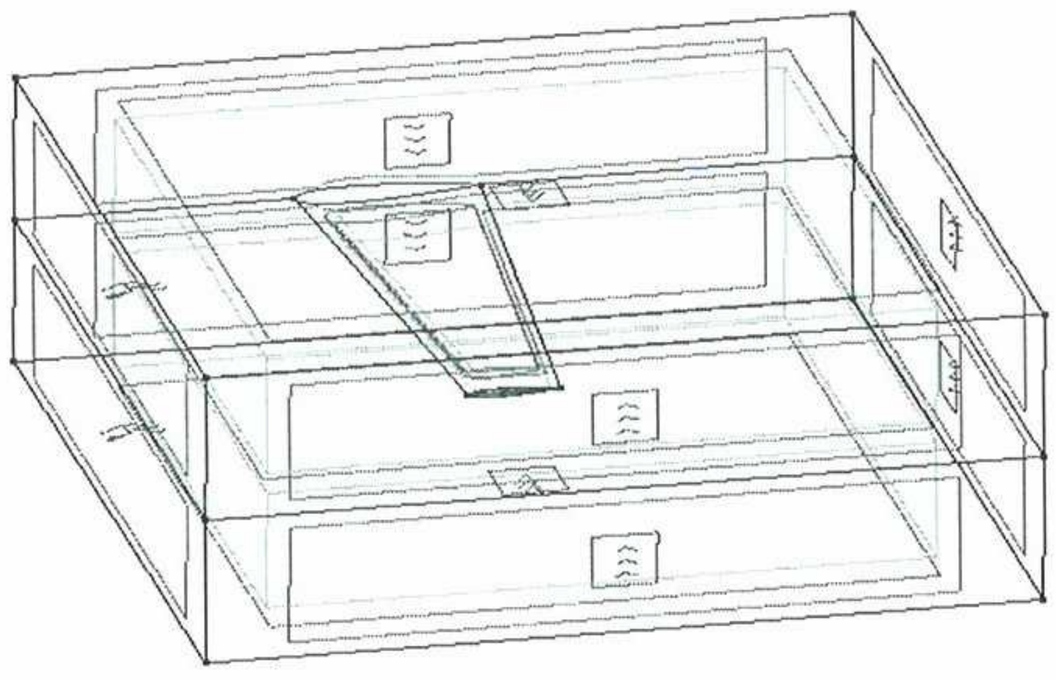


Figure 6.6.1 3D model of a wing with fluid conditions assigned

References

1 http://cs.felk.cvut.cz/~xobitko/ga/

2 Bio-Inspired Optimization Algorithms for Engineering Applications – Sibylle D. Müller

3 Optimization: An attempt at describing the State of the Art – Elke Pahl

4 www.geatbx.com

5 Aerodynamic Shape Optimization using Probabilistic-Stochastic Search Methodologies – Craig P. Thamotheram

6 "Evolutionsstrategie: Optimierung technischer System nach Prinzipien der biologischen Evolution" – Fromman-Holzboog, Stuggart, 1973 - Rechenberg, I

7 Adapting Arbitrary Normal Mutation Distributions in Evolutionary Strategies: The Covariance Matrix Adaptation – Nikolaus Hansen & Andreas Ostermeier, 1996

8 http://www.inf.ethz.ch/personal/hansenn/CMAES2.pdf

9 Reducing the Time Complexity of the Derandomised Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) - Nikolaus Hansen, Sibylle D. Müller, Petros Koumoutsakos

10 http://e-collection.ethbib.ethz.ch/ecol-pool/diss/fulltext/eth14719.pdf

11 http://wanda.fh-aargau.ch/staff/bueche/download/opt_cost526/

12 www.mathworld.wolfram.com/Hessian.html

13 Differential Evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces – Rainer Storn and Kenneth Price

14 http://raphael.mit.edu/xfoil/

15 Tdyn theory

16 Tdyn Reference Manual

17 A stabilized finite point method for fluid mechanics, Comput. Methods Appl. Mech. Engrg. 39 (1996) 315-346 - E. Onate, S. Idelsohn, O. C. Zienkiewicz, R. L. Taylor and C. Sacco.

18 Finite Calculus Formulation for Finite Elements analysis of incompressible flows. (Eulerian, ALE and Lagrangian approaches, Comput. Meth. Appl. Mech. Engng) (2004) - E. Onate, A. Valls, J. Garcia

19 NACA Technical Memorandum 110446, April 1997, Turbulence Modelling Validation, Testing, and Development - J. E. Bardina, P. G. Huang, and T.J. Coakley.

20 Turbulence and Transition Modelling – ERCOFTAC Summerschool, Stockholm June 1995

21 http://www.stanford.edu/class/me469b/handouts/turbulence.pdf

22 http://www.pagendarm.de/trapp/programming/java/profiles/NACA4.html

23 http://www.mech.upatras.gr/~flow2002/papers/pe06.pdf

24 http://www-waterloo.ansys.com/cfx/PDF/PDF0035.pdf

25 http://www.aerospaceweb.org/question/airfoils/q0041.shtml

26 Convergence Properties of Evolution Strategies with the Derandomised Covariance Matrix Adapation: The $(\mu/\mu_I,\lambda)$-CMA-ES – Nikolaus Hansen & Andreas Ostermeier, 1997

27 Completely Derandomised Self-Adaptation in Evolutionary Strategies – Nikolaus Hansen & Andreas Ostermeier, 2001

28 Optimum Aerodynamic Shape Design for Fluid Flow Problems Including Mesh Adaptivity – G. Bugeda, E. Oñate

29 http://www.waterloo.ansys.com/cfx/PDF/PDF0035.pdf

30 http://www.ams.org.mcom/2004-73-248/ S0025-5718-04-01660-6/home.html

Appendix A: Additional Results

A.1 Problem 1 Results:

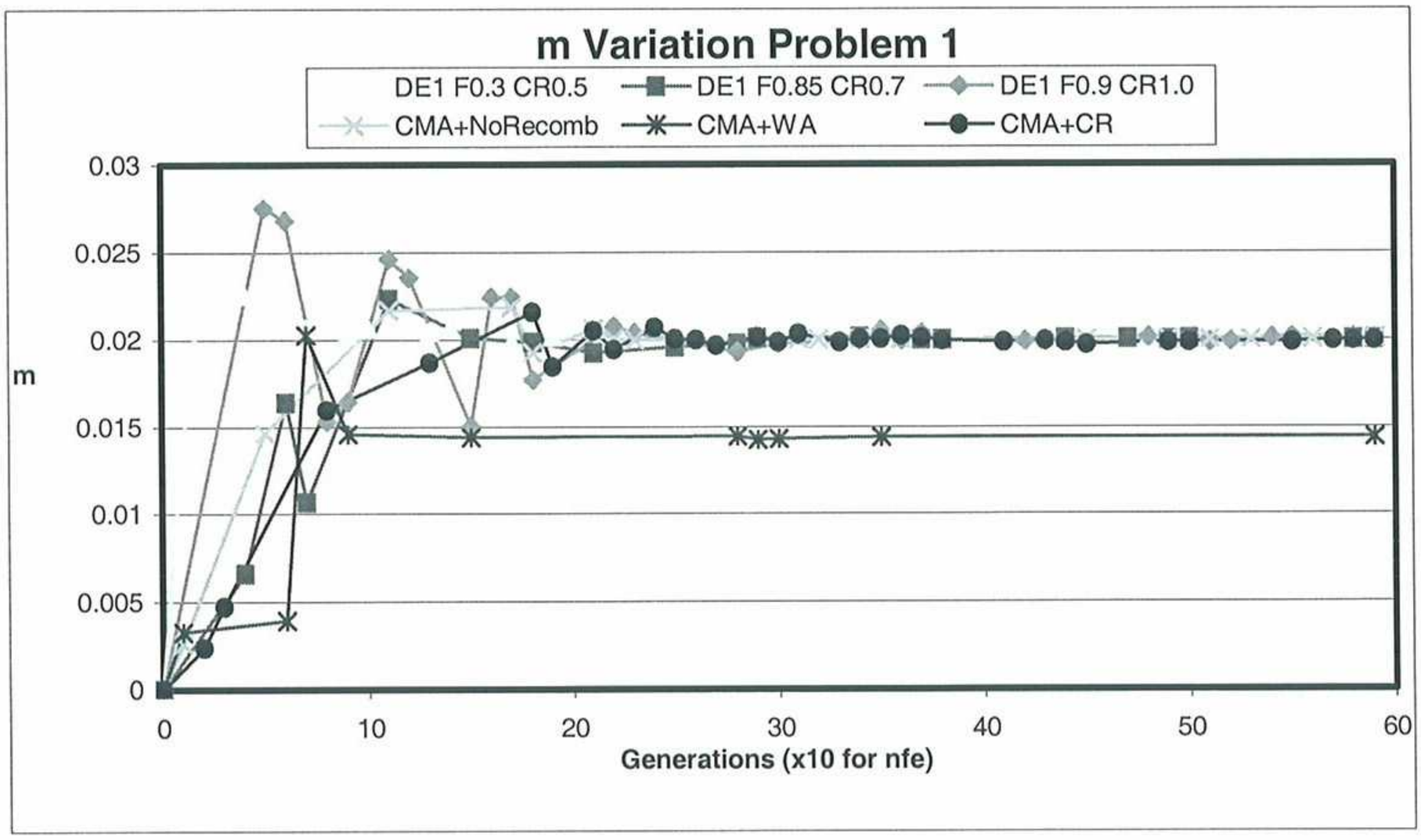


Figure A.1.1 Problem 1 m variation vs number of Generations

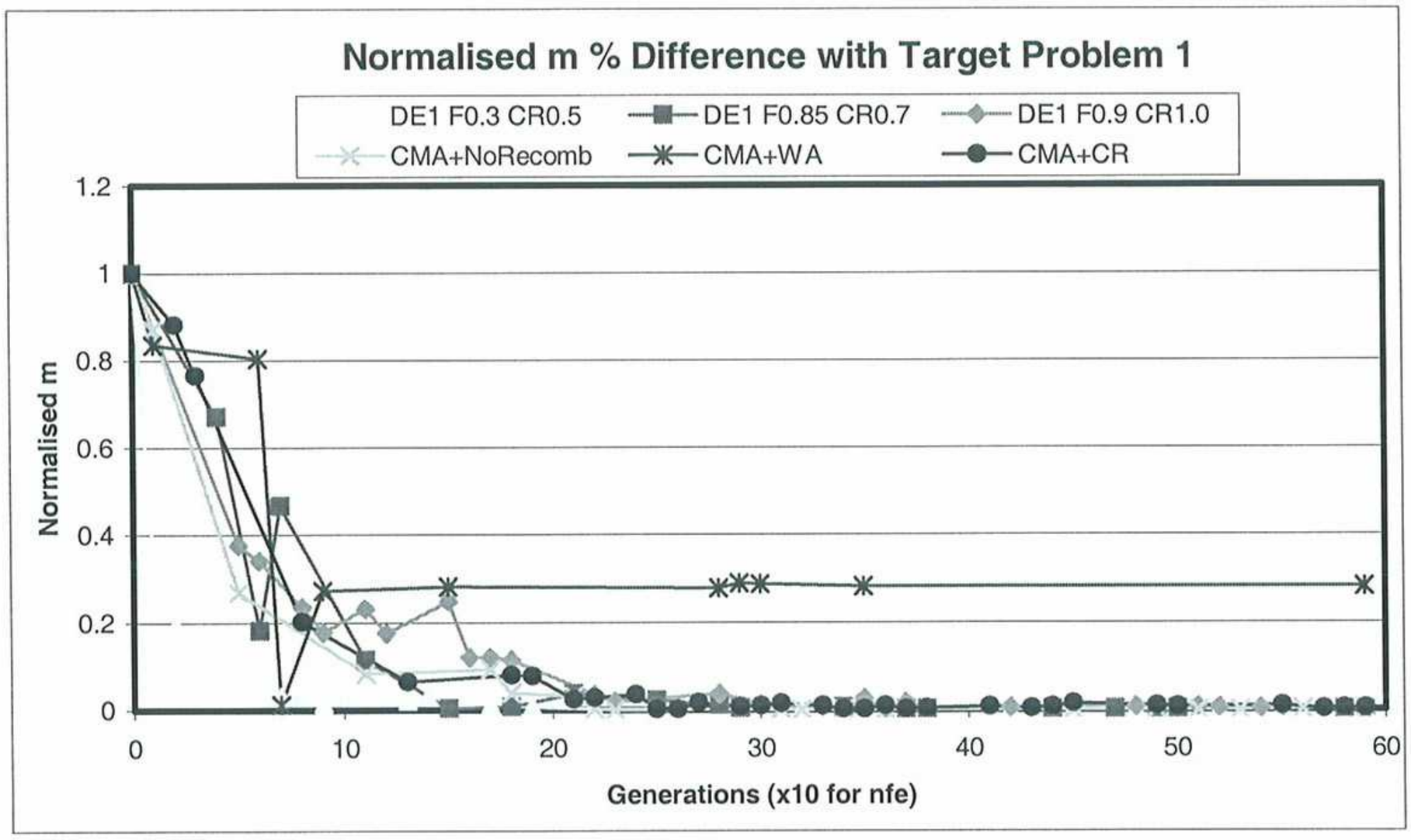


Figure A.1.2 Problem 1 Normalised m % difference with Target NACA2415 values vs number of Generations

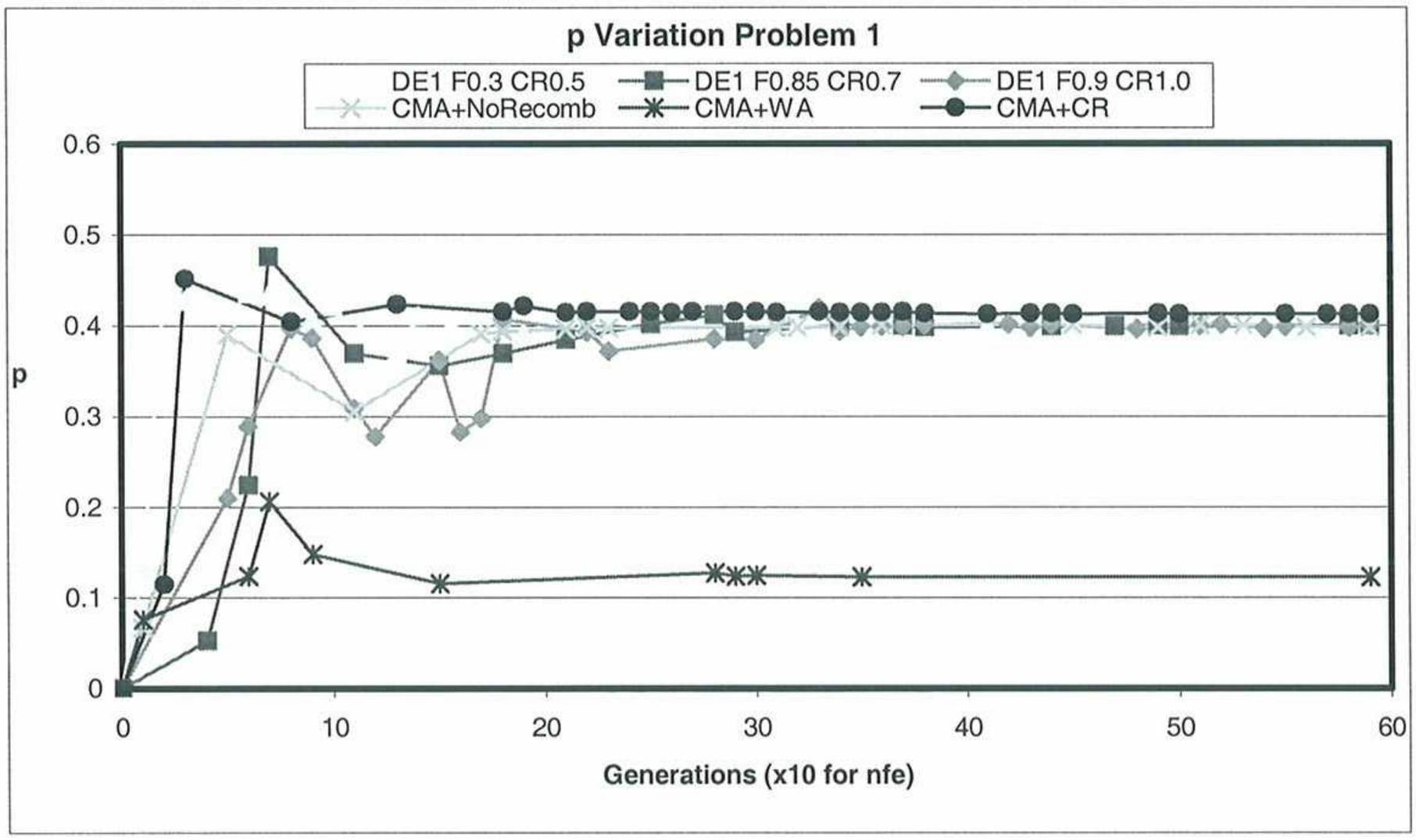


Figure A.1.3 Problem 1 p variation vs number of Generations

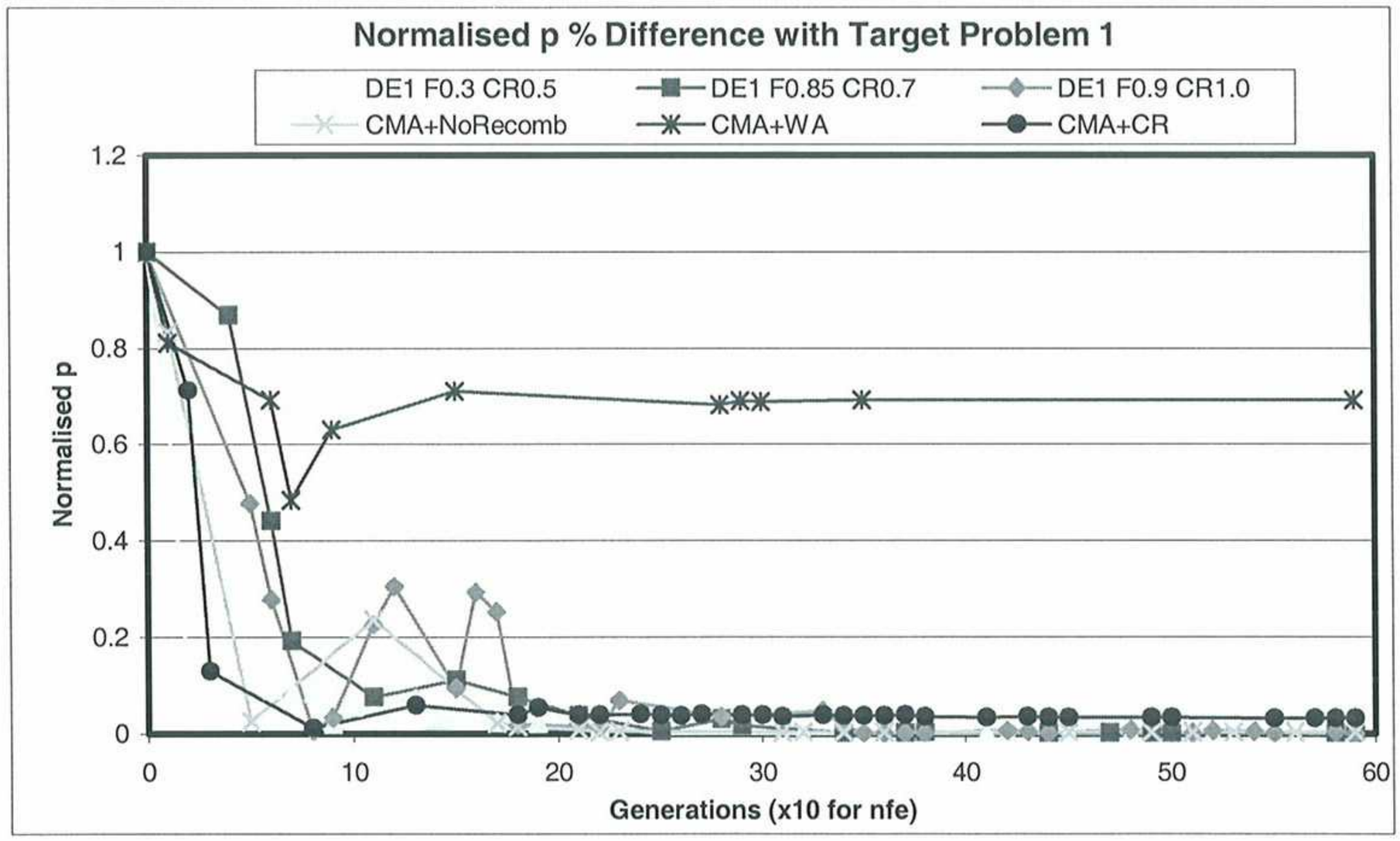


Figure A.1.4 Problem 1 Normalised p % difference with Target NACA2415 values vs number of Generations

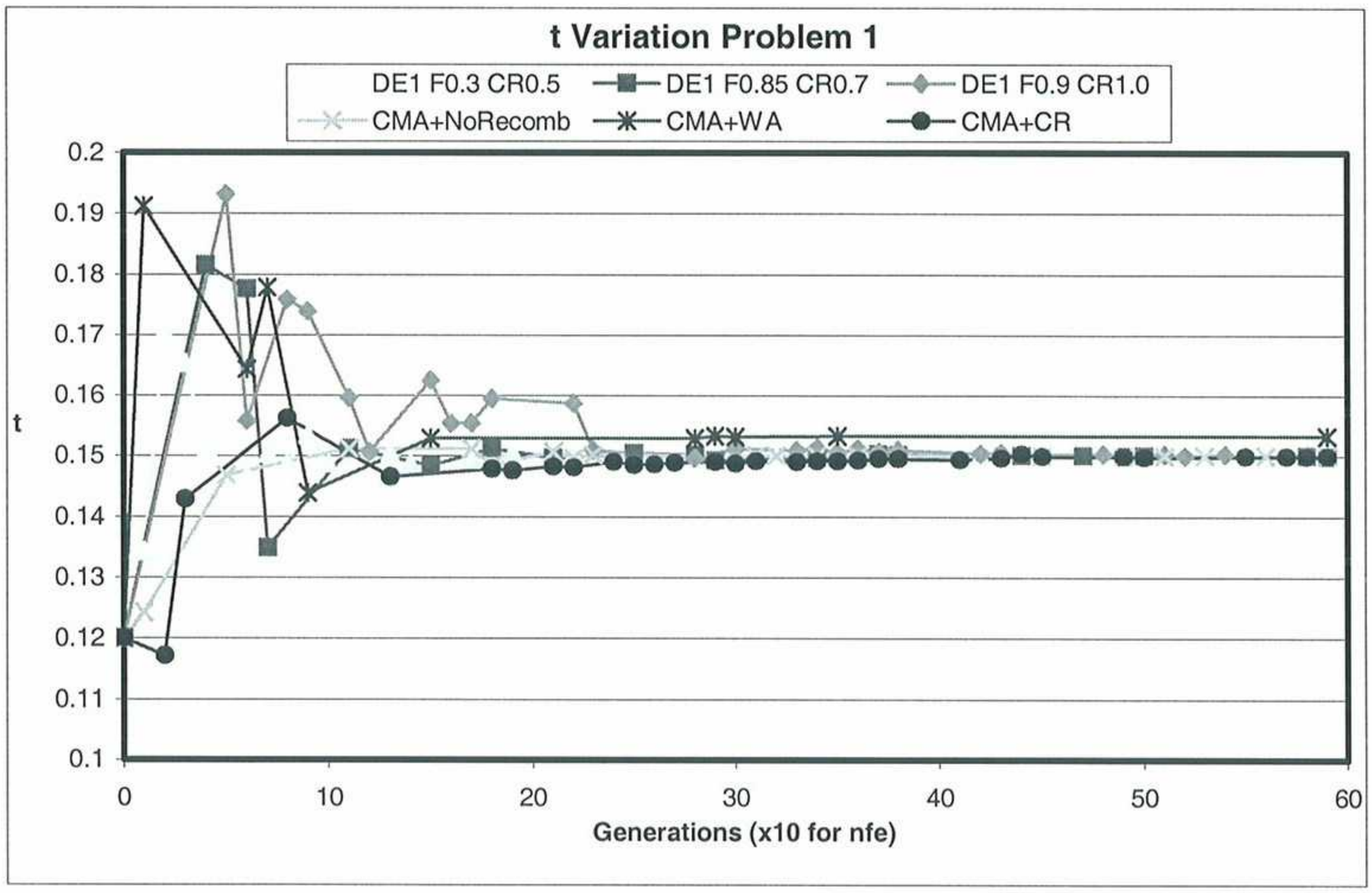


Figure A.1.5 Problem 1 t variation vs number of Generations

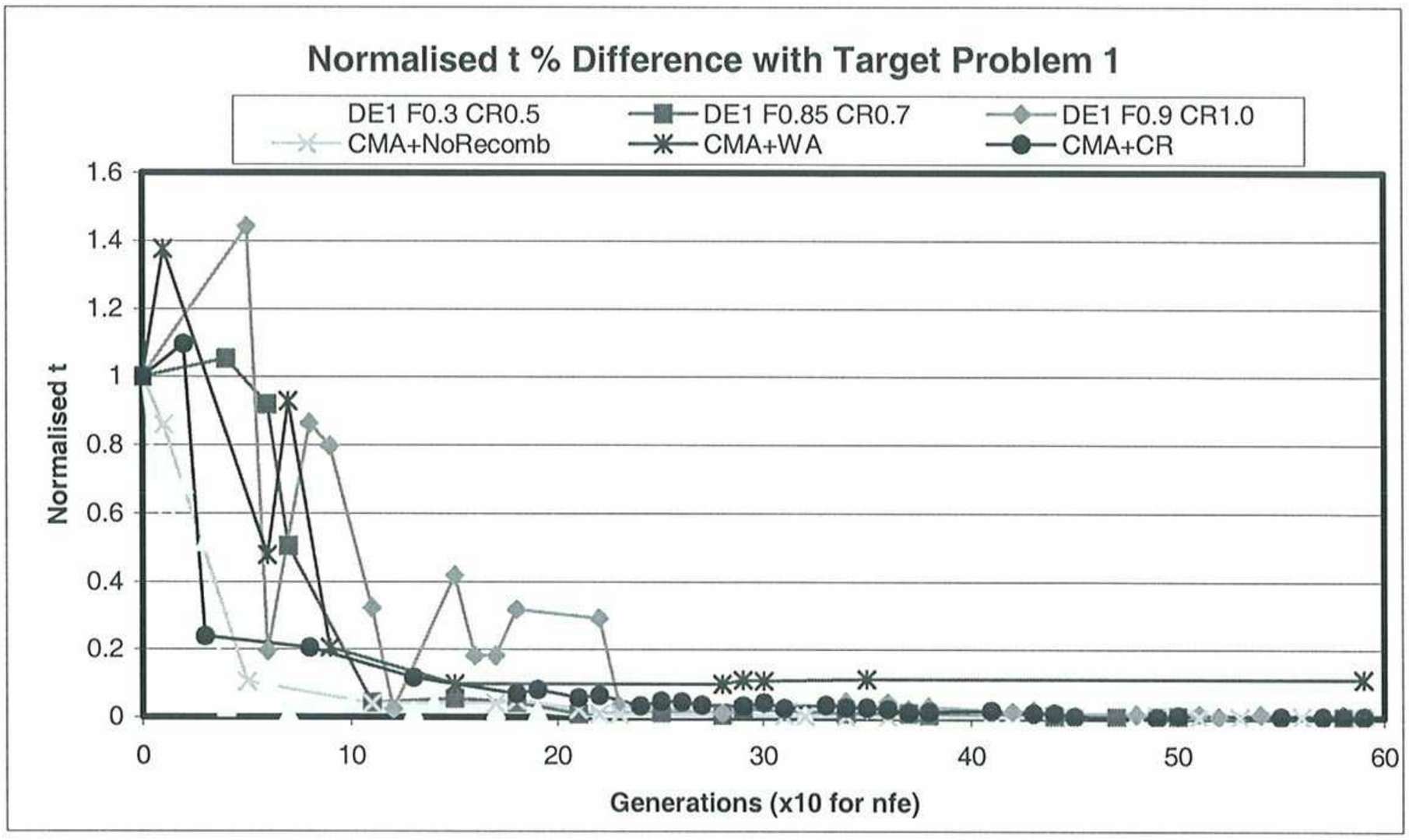


Figure A.1.6 Problem 1 Normalised t % difference with Target NACA2415 values vs number of Generations

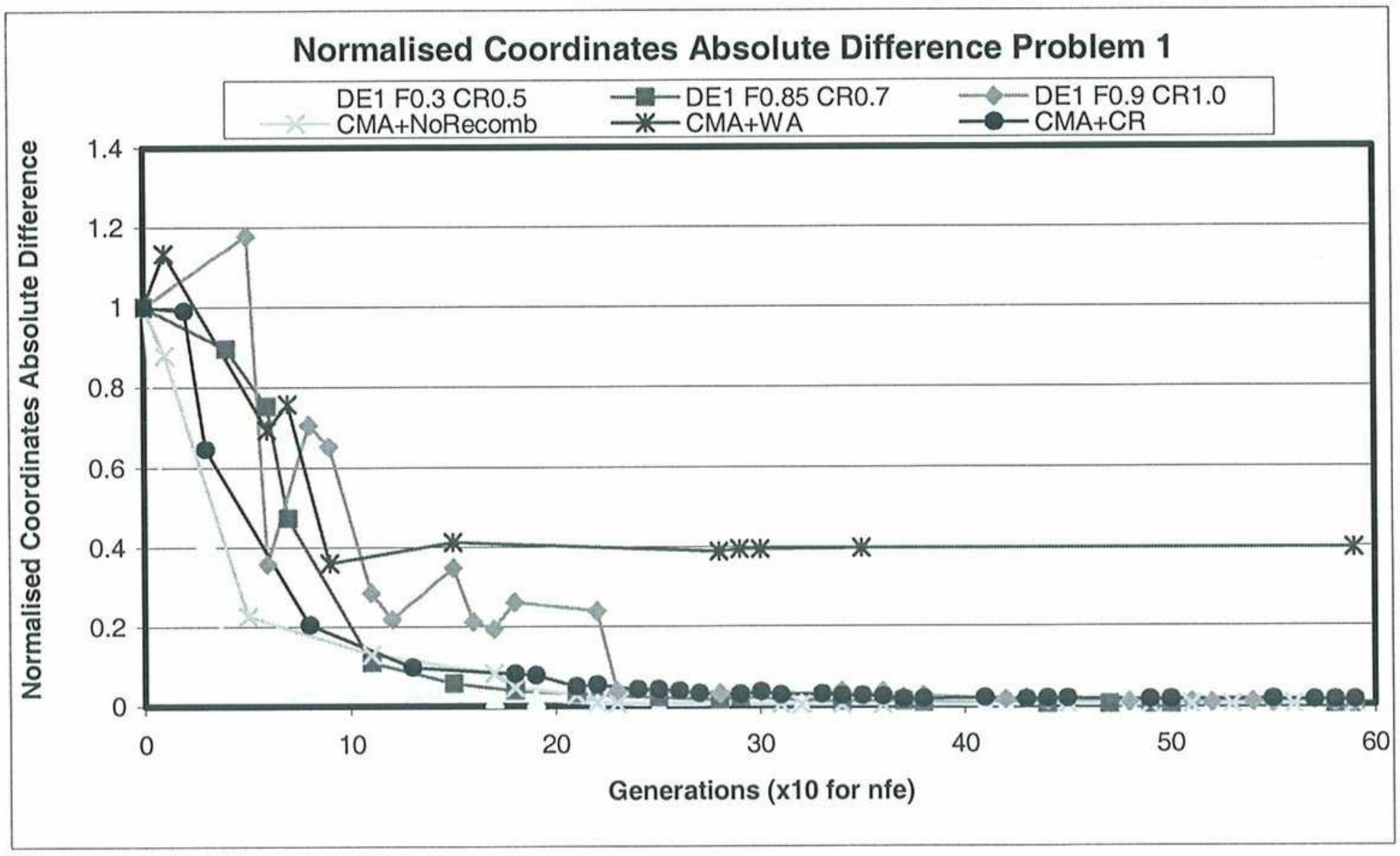


Figure A.1.7 Problem 1 Normalised coordinates difference with Target NACA2415 values (sum of magnitude of differences between the aerofoil and the target for the 19 coordinates defining the aerofoil) vs number of Generations

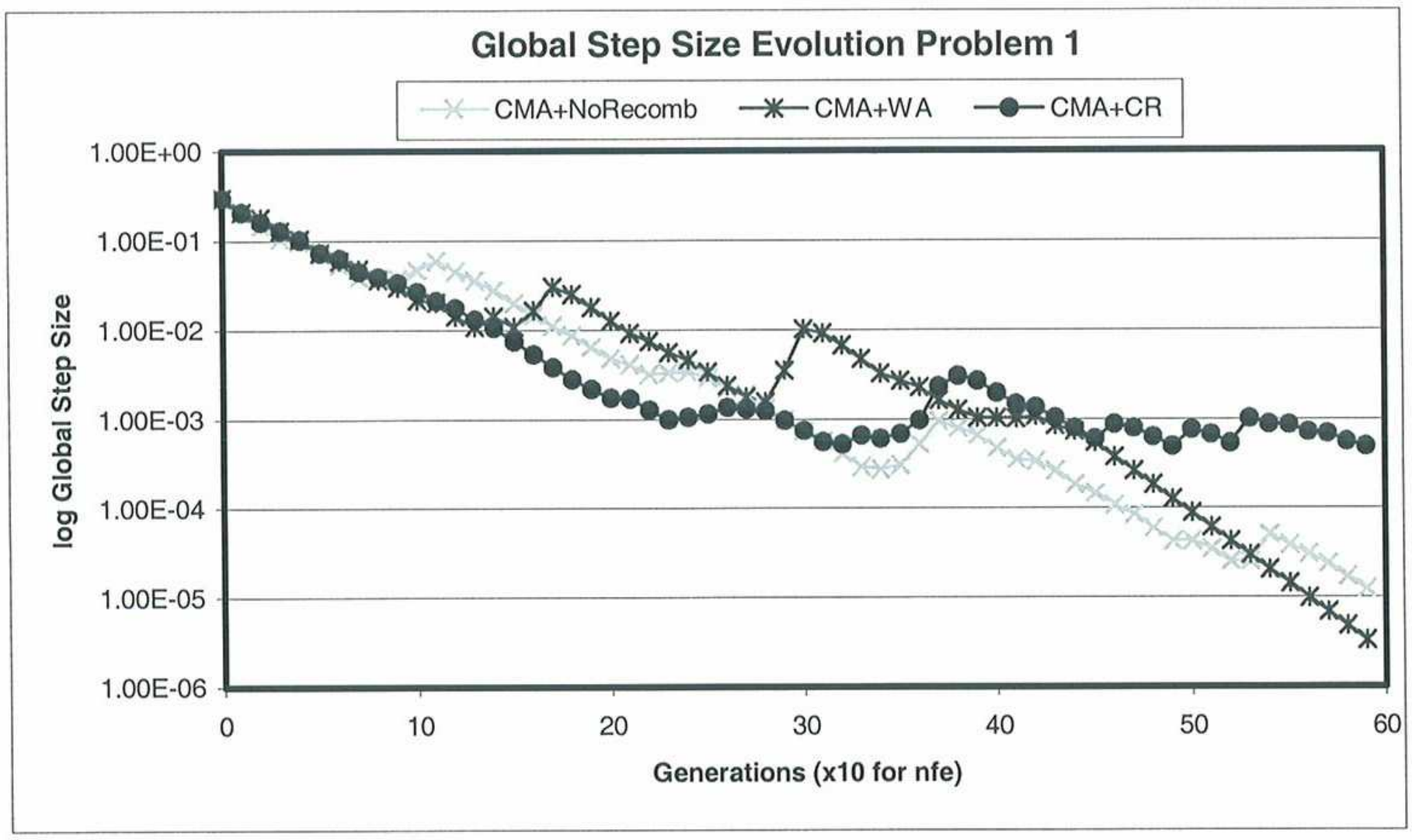


Figure A.1.8 Problem 1 Global step size evolution vs number of Generations

A.2 Problem 2 Results

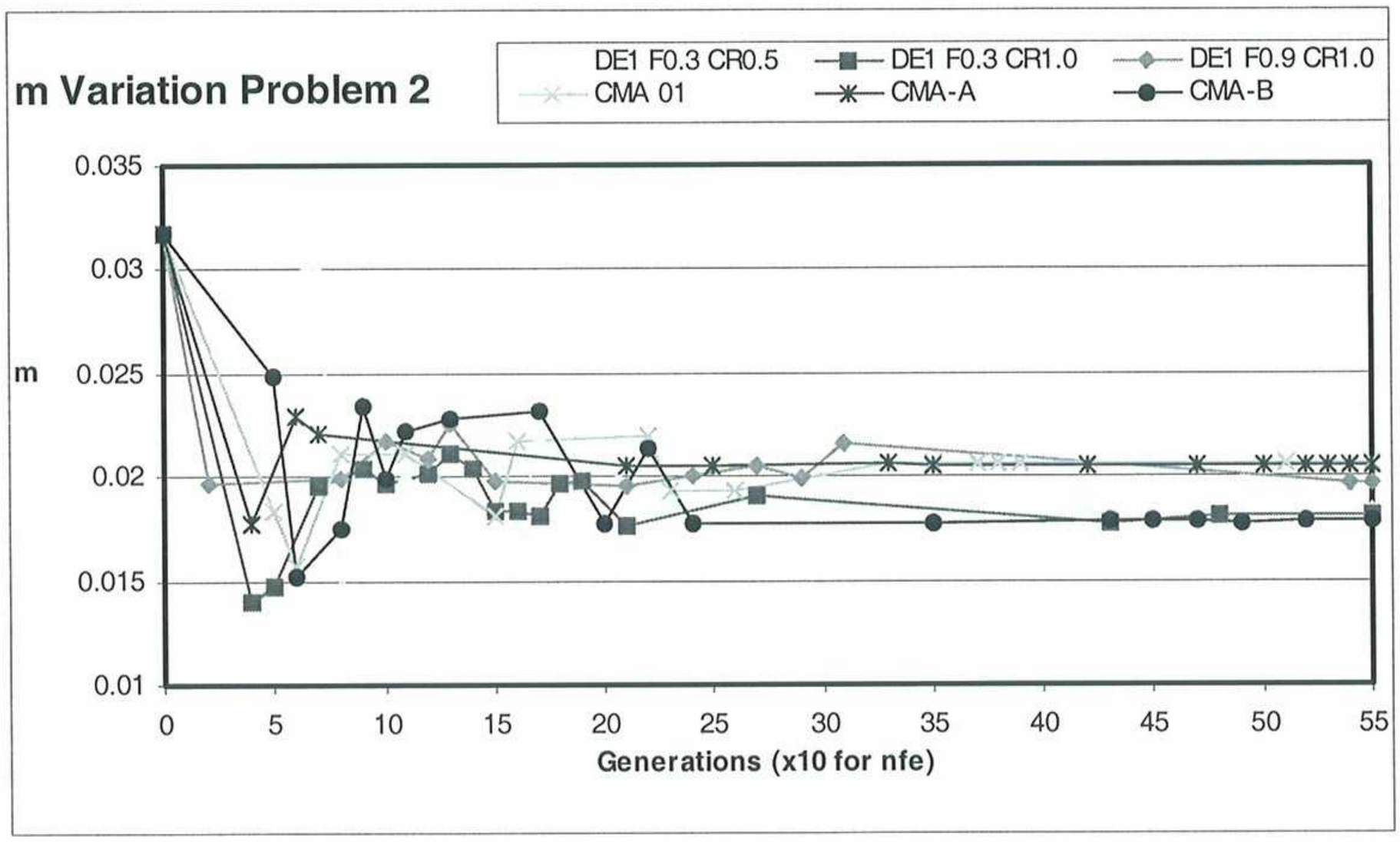


Figure A.2.1 Problem 2 m variation vs number of Generations

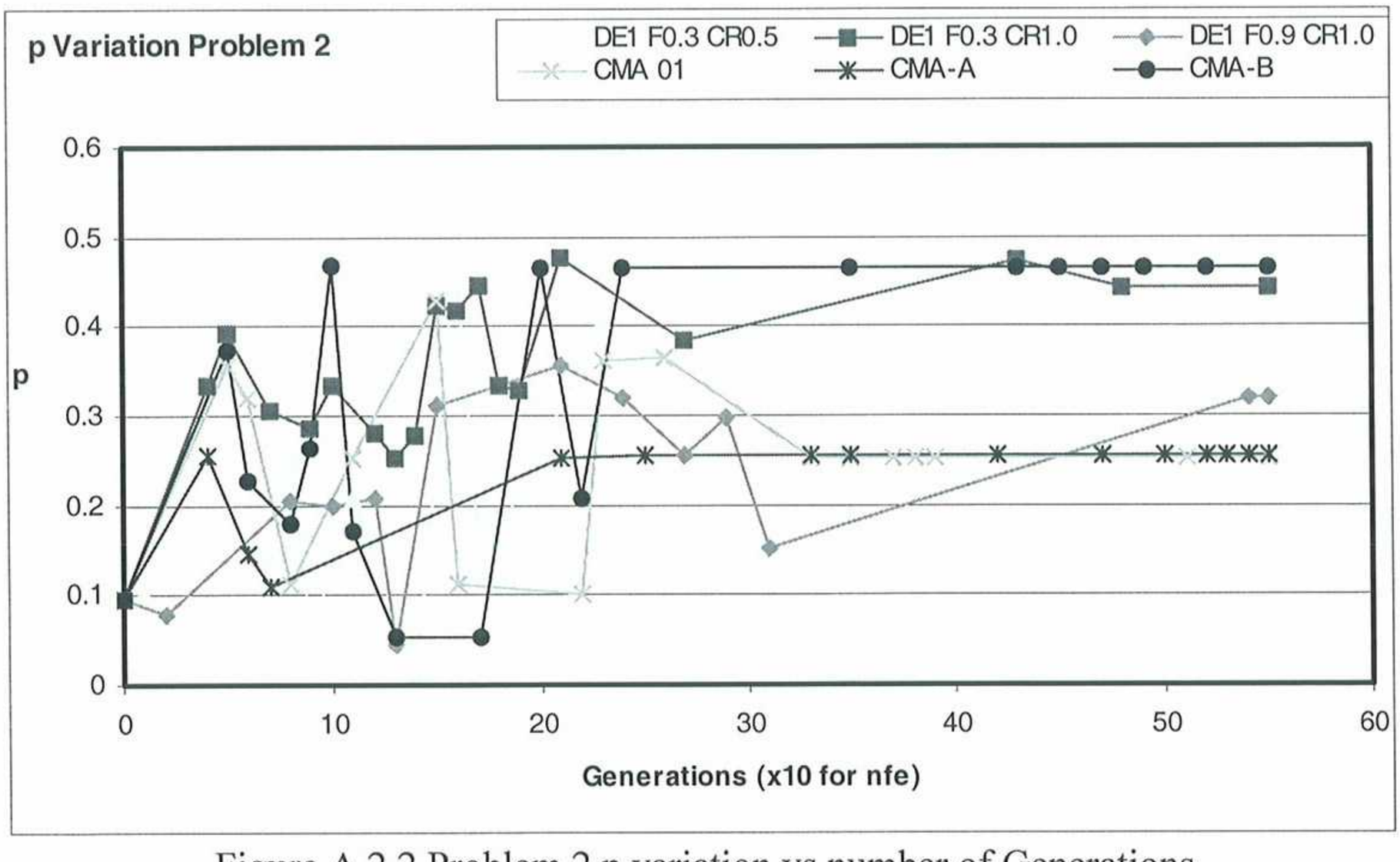


Figure A.2.2 Problem 2 p variation vs number of Generations

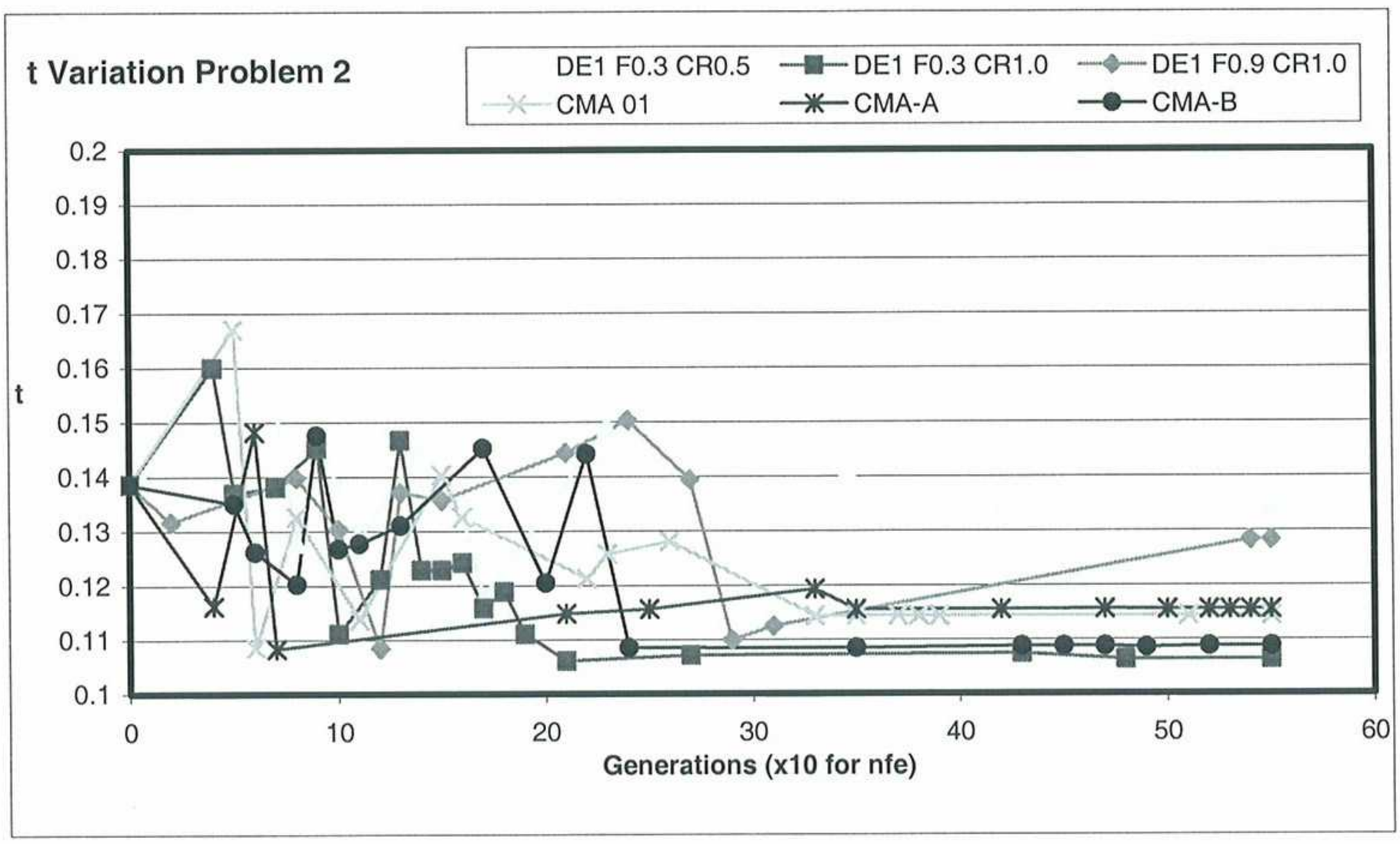


Figure A.2.3 Problem 2 t variation vs number of Generations

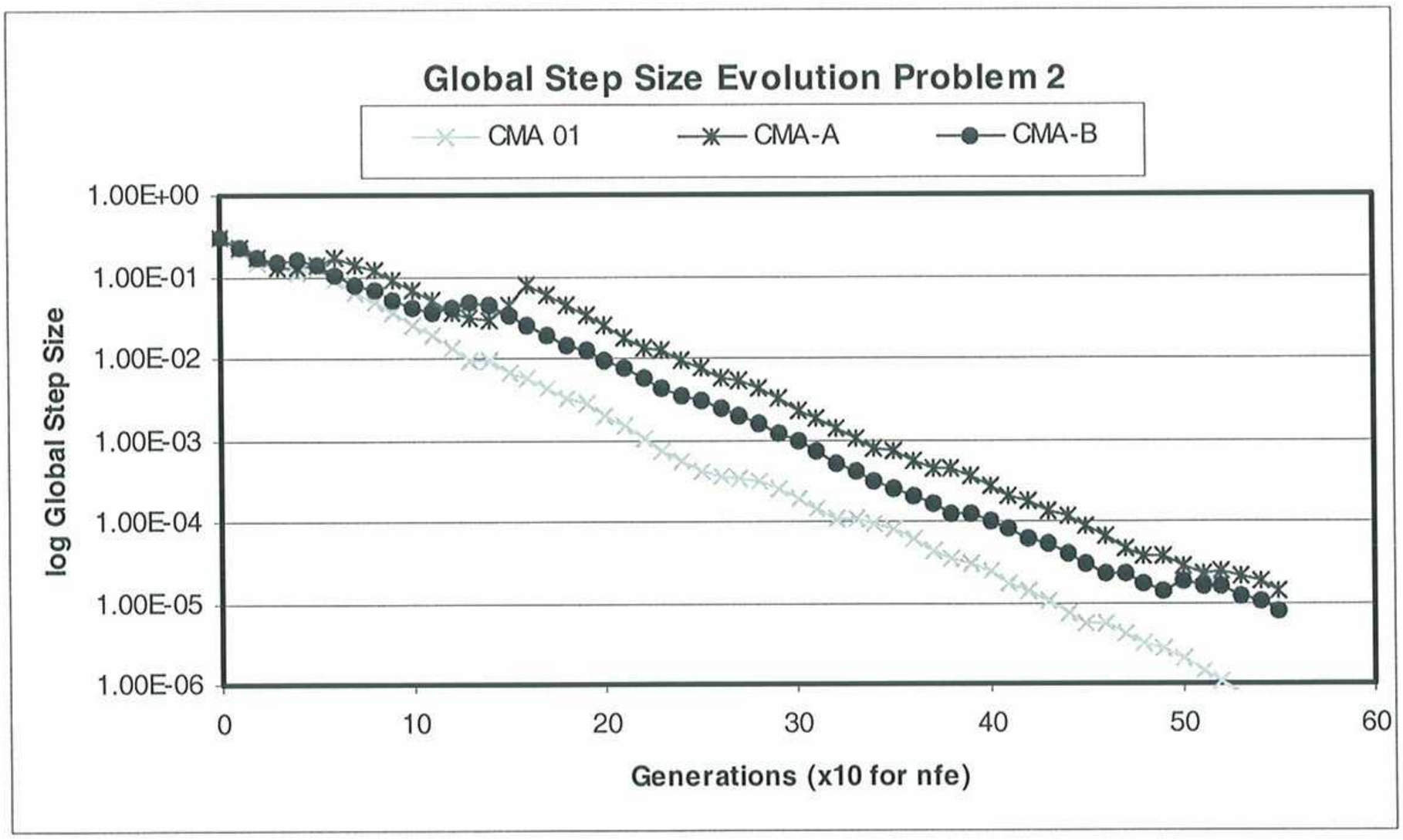


Figure A.2.4 Problem 2 Global step size evolution vs number of Generations

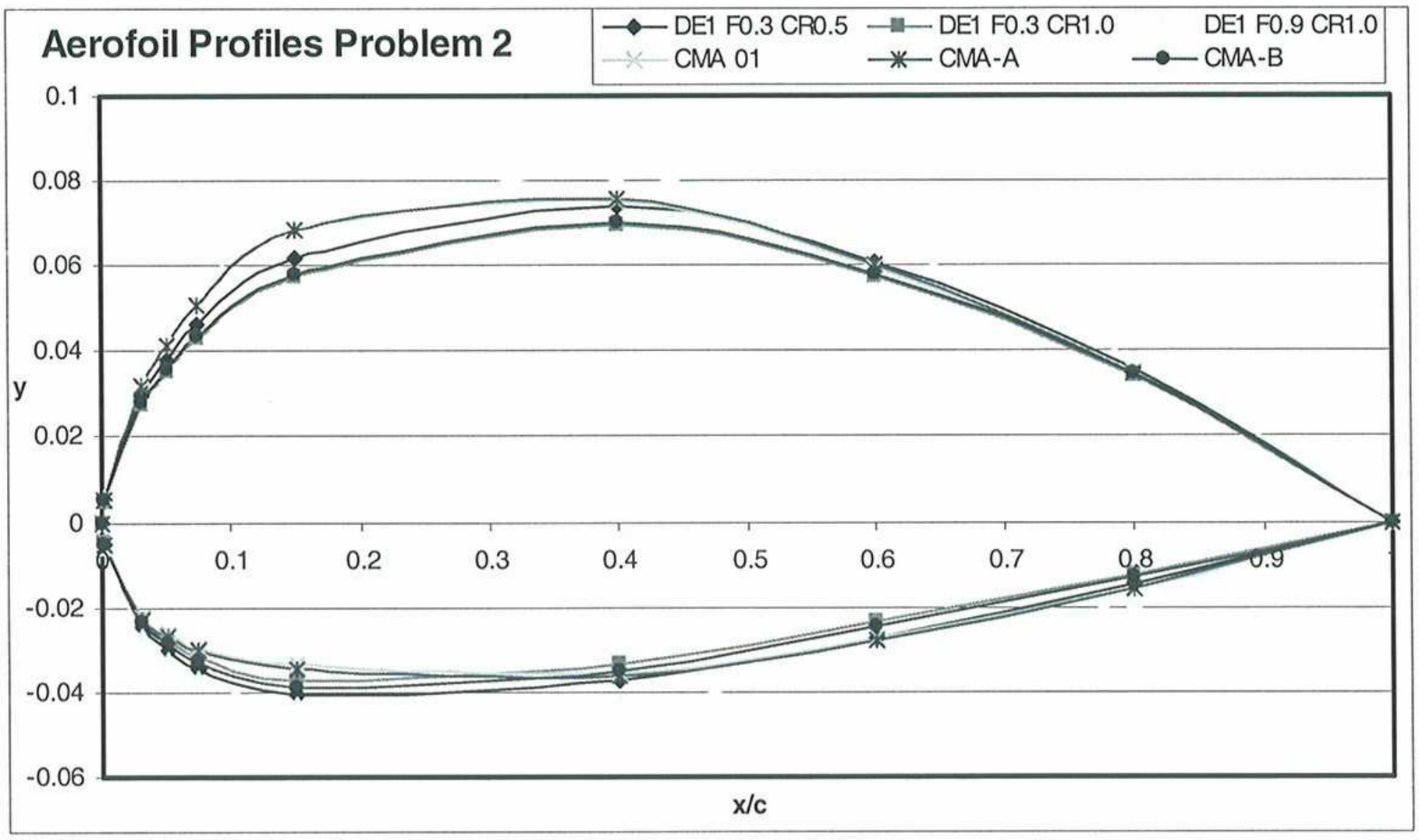


Figure A.2.5 Problem 2 Coordinates profile vs number of Generations for DE1 &CMA schemes

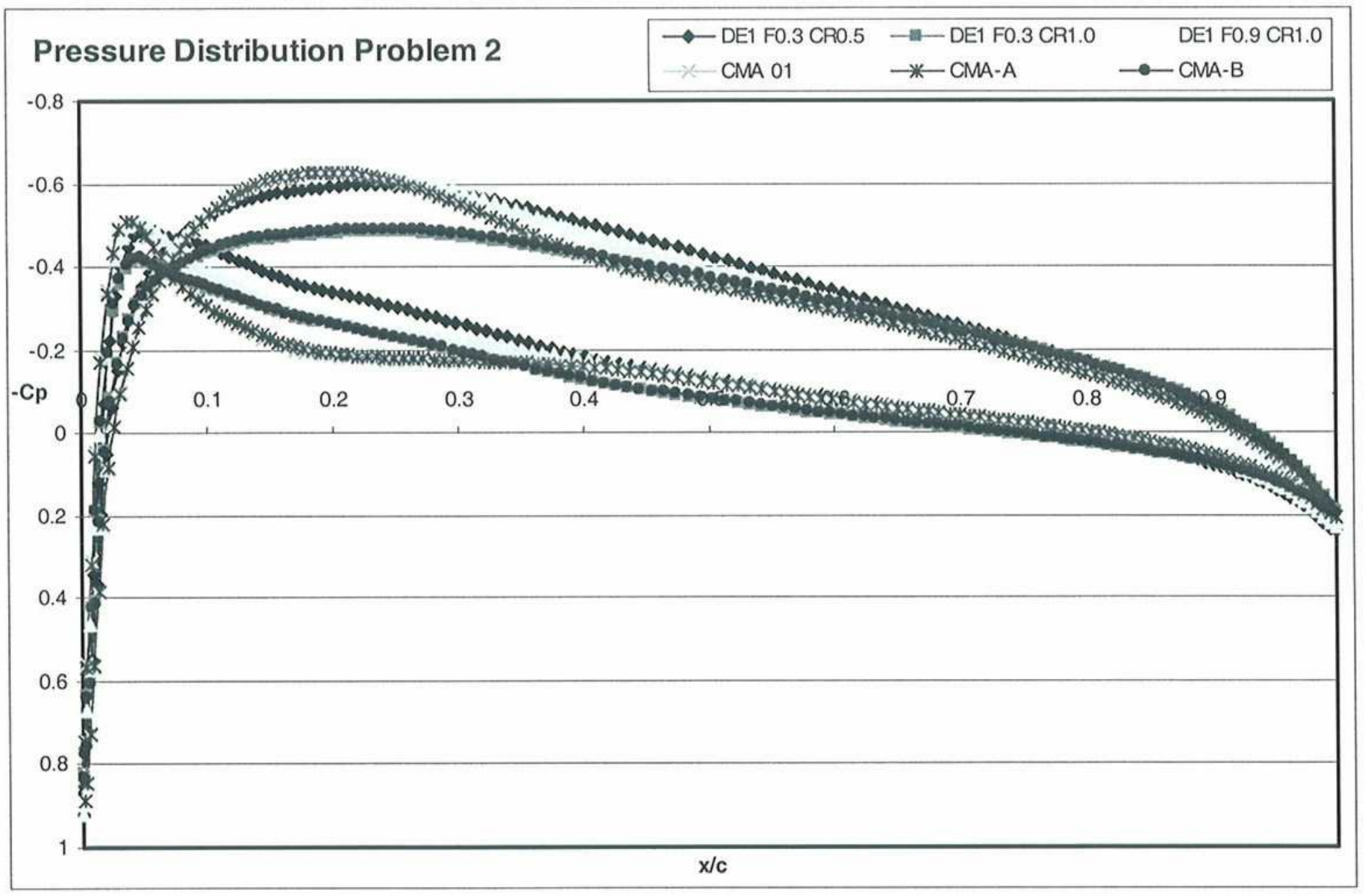


Figure A.2.6 Problem 2 Pressure Distribution vs number of Generations for DE1 & CMA schemes

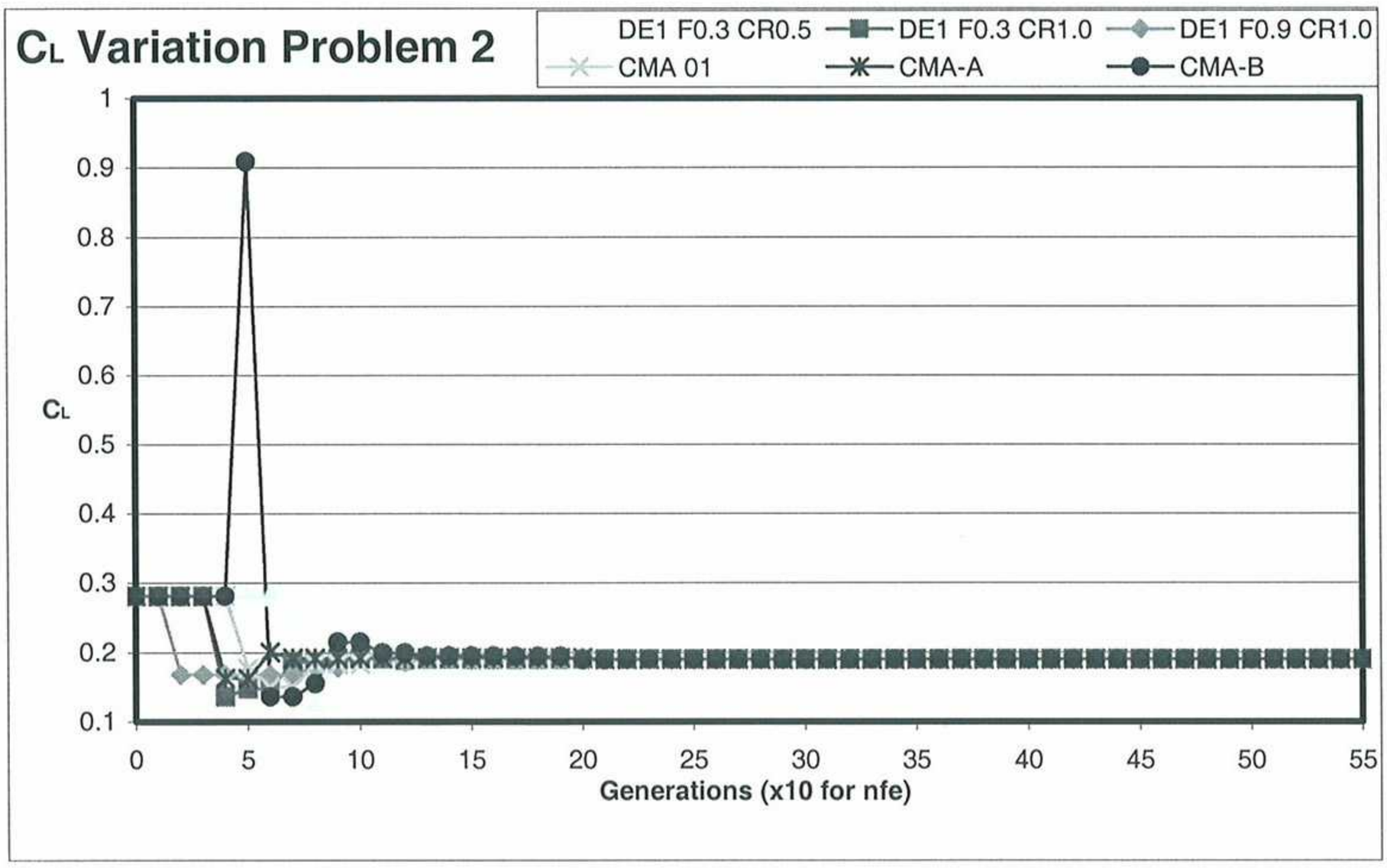


Figure A.2.7 Problem 2 C_L variation vs number of Generations

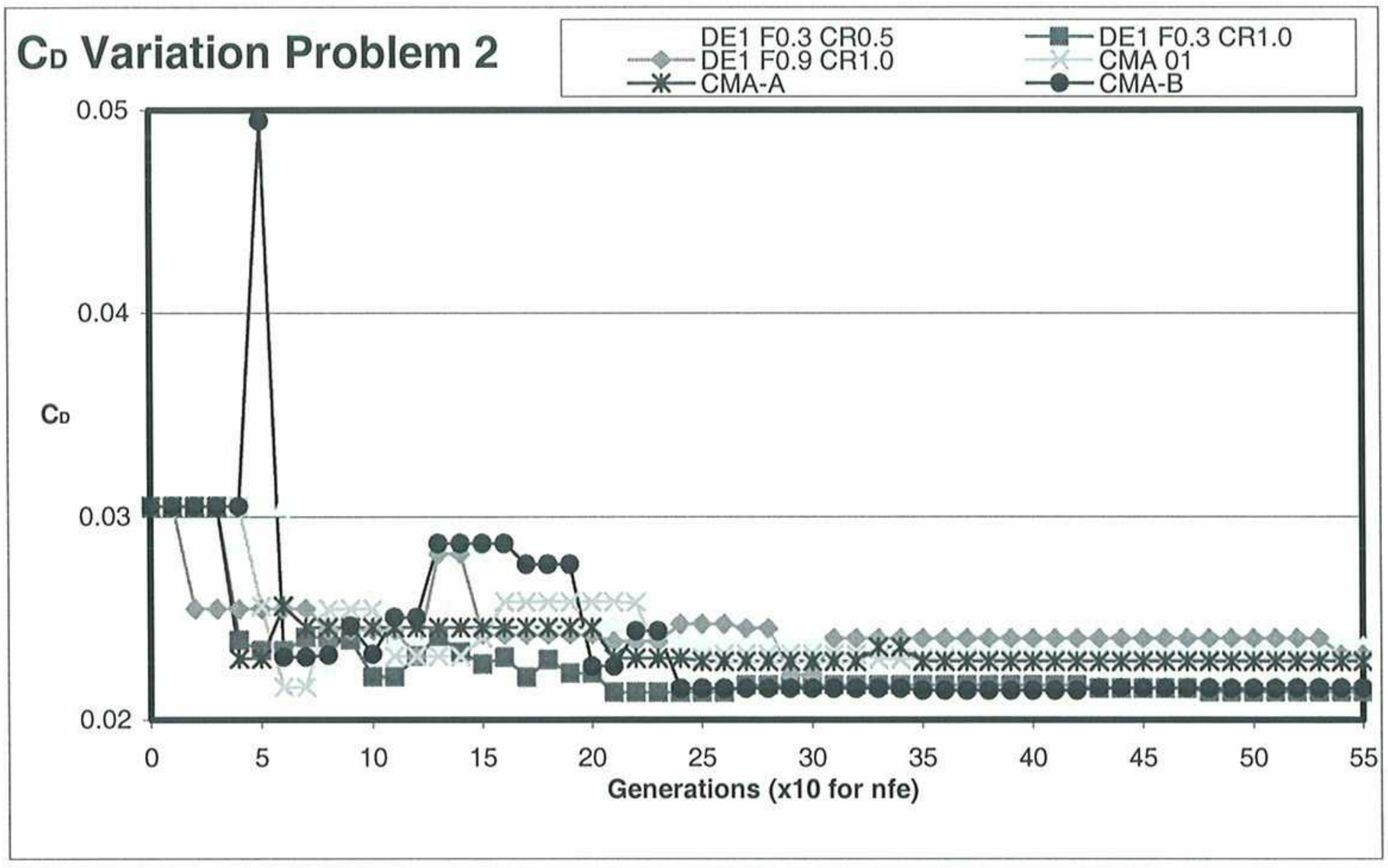


Figure A.2.8 Problem 2 C_D variation vs number of Generations

Appendix B: Tdyn and Turbulence Modelling Theory

Tdyn is a finite element method fluid dynamics simulator and is a very important part of the optimisation process since it will be relied on to model a problem and produce the required results. The following is a summary of the fundamentals of Tdyn, provided to allow a better understanding of how the program works and any limitations it may have.

B.1 Tdyn Principles

Tdyn solves the three dimensional, incompressible and slightly compressible Navier Stokes Equations in a given domain Ω and time interval (0,t):

$$\left.\begin{array}{lll} \text{(A)} & \rho\left(\dfrac{\partial u}{\partial t}+(u\cdot\nabla)u\right)+\nabla p-\nabla\cdot(\mu\nabla u)=\rho f \ \text{ in } & \Omega\times(0,t) \\ \text{(B)} & \nabla\cdot u=0 \qquad\qquad \text{in} & \Omega\times(0,t) \end{array}\right\} \quad \text{Equation B.1}$$

where $u=u(x,t)$ is the velocity vector, $p=p(x,t)$ is the pressure field, ρ is the (constant) density, μ, the fluid dynamic viscosity and f is the volumetric acceleration.

The following boundary conditions are then applied:

$$\left.\begin{array}{ll} \text{(A)} & u=u_c \text{ in } \Gamma_D\times(0,t) \\ \text{(B)} & p=p_c \text{ in } \Gamma_P\times(0,t) \\ \text{(C)} & n.\sigma.g_1=0,\ n.\sigma.g_2=0,\ n.u=u_M \text{ in } \Gamma_M\times(0,T) \\ \text{(D)} & u(x,0)=u_0(x) \text{ in } \Omega_D\times\{0\} \\ \text{(E)} & p(x,0)=p_0(x) \text{ in } \Omega_D\times\{0\} \end{array}\right\} \quad \text{Equation B.2}$$

In Equation B.2, Γ represents the boundary to the domain Ω, n is the normal unit vector, g_1 and g_2 the tangent vectors of the boundary surface, u_c is the velocity field on Γ_D (the Dirichlet type boundary), p_c is the prescribed pressure on Γ_P (the pressure boundary), σ is the stress field, u_M is the value of normal velocity and u_0 and p_0 are the initial velocity and pressure fields respectively. Together, Γ_P, Γ_D and Γ_M form the boundary Γ but a point existing in one of the boundaries can only be of one type unless it is part of the border between two boundaries.

In Tdyn, the Finite Element Method (FEM) is used to space discretise the Navier-Stokes equations while an iterative implicit two steps Fractional Step type method is used for time discretisation so as to avoid the order reduction caused by non-constant boundary conditions in the methods [30]. Stabilisation of convection dominated problems is done by the Finite Calculus method which seeks to change the formulation of the problem from the outset by considering the balance of flux over a finite sized domain. By doing so, this method hopes to attain higher order terms, previously absent in other standard methods, which supply the necessary stability for a classical Galerkin finite element discretisation to now be used with equal order velocity and pressure terms. The final stabilised momentum balance equation is presented in the following section. It is through a standard time discretization of this equation that we arrive at the velocity and pressure equations, solved iteratively by Tdyn, necessary for the optimisation problems' simulation and analysis.

B.2 Turbulence Modelling in Tdyn [16, 19, 20, 21]

Turbulent flow plays a part in many engineering problems and it is characterised by a multitude of scales in time and space. The associated mixing and diffusion may be orders of magnitudes larger than in the laminar case and can dominate the flow behaviour. It is therefore crucial to understand these to accurately model the behaviour of the flow.

B.2.1 Reynolds Averaging – It is undesirable and unfeasible to consider all of the small scale fluctuations that occur in the presence of turbulence. Therefore, the turbulent velocity u_i is decomposed into a fluctuating component u_i' and a steady component $\overline{u}_i$ by a process called Reynolds Averaging.

$$u_i = \overline{u}_i + u_i' \qquad \text{Equation B.3}$$

where

$$\overline{u}_i(x,t) = \frac{1}{T}\int_{-T/2}^{T/2} u_i(x,t+\tau)d\tau \qquad \text{Equation B.4}$$

The substitution and averaging of these quantities in the governing equations give rise to the Reynolds-Averaged Navier-Stokes equations (RANSE). The RANSE now contain extra terms that can be written in as a function of the Reynolds stresses tensor in Cartesian coordinates as:

$$\tau_{ij}^R = -\rho\overline{u_i' u_j'} \qquad \text{Equation B.5}$$

These extra terms in the governing equations now necessitate extra information in order to solve the system. This is done by defining the Reynolds stresses in terms of known (averaged) quantities through turbulence models.

B.2.2 Turbulence Models – Tdyn employs turbulence models which introduce an eddy viscosity μ_T to relate the unknown τ^R to mean flow variables in accordance with Boussinesq's postulation which says that, in analogy to molecular diffusion, the Reynolds stresses depend on the deformation rates of the mean flow. The models are categorised according to the number of partial differential transport equations that, for most cases, describe the turbulence velocity and length scale, and need to be solved to find the eddy viscosity in addition to the RANSE and ultimately solve for the unknown Reynolds stresses.

Zero equation models through to four equation models exist, however, the increased accuracy of higher number transport equation models is accompanied with added complexity and computing cost. For this reason, only zero, one and two equation models (being also the only ones used in TDyn), which have experienced the most success on balance of accuracy and cost, are going to be considered. The final form of the RANSE using these models with Finite Calculus stabilisation formulation (see [17,18] for derivation) is given below:

$$r_{mi} - \frac{h_{mj}}{2}\frac{\partial r_{mi}}{\partial x_j} = 0 \qquad \text{on } \Omega,\ i,j = 1,2,3.\ \text{No sum in } i. \qquad \text{Equation B.6}$$

$$r_d - \frac{h_{dj}}{2}\frac{\partial r_d}{\partial x_j} = 0 \qquad \text{on } \Omega,\ j = 1,2,3. \qquad \text{Equation B.7}$$

Equations 2.9.5 and 2.9.6 are the Stabilised Momentum balance equation and the Stabilised Mass equation, respectively, where

$$r_{mi} = \rho\left[\frac{\partial u_i}{\partial t} + \frac{\partial\left(u_i u_j\right)}{\partial x_j}\right] + \frac{\partial p}{\partial x_i} - \frac{\partial}{\partial x_j}\left[\left(\mu + \mu_T\right)\frac{\partial u_i}{\partial x_j}\right] - \rho f_i$$

$$r_d = \rho\frac{\partial u_i}{\partial x_i} \quad \text{with } \mu_T \text{ in the above equation being the eddy viscosity.}$$

B.2.3 Tdyn Models

i. Zero Equation models: Smagorinski model, Mixing-Length model

These employ no transport equations and, working now in relation to the kinematic eddy viscosity υ_T (also called the turbulence viscosity), this is modelled by experimental or empirical functions of the local mean flow variables so that turbulence is considered to be generated and dissipated in the same place, i.e. diffusion and convection of the turbulence are ignored.

ii. One Equation models: k model, Spallart-Allmaras model

These equations model turbulent transport through only one transport differential equation while the other turbulent quantity is specified from empirical formulation. For the k model, the turbulence viscosity is equated to a velocity scale $k^{1/2}$ and also an empirically specified length scale L. Following manipulation of the Reynolds and momentum equations, the ´k equation´, a partial differential equation to solve k, is obtained. The introduction of inaccuracy through experimental values for L is shared with the zero equation models and has been shown to limit the usefulness and generality of this type of model. Modern one equation models, such as Spalart-Allmaras, abandon the k equation and are based on a ad-hoc transport equation for the eddy viscosity directly.

iii. Two equation models: k-ε model, k-ω model, k-ω SST model, k-kτ model

It is with two equation models that flow history effects can be accounted for to some reasonable degree. Here, the transport equations for velocity and length scales, or alternatively two other quantities that make up the eddy viscosity, are developed. All of the models investigated here employ the k equation formulation for the velocity scale. The second required formula may be in terms of a variety of turbulence quatitites. Tdyn presents models based on a time scale τ, a dissipation rate ε and an inverse time scale ω.

The model chosen for problem modelling in this report is the k-ω Shear Stress Trransport (SST) model. This is a hybrid 2 equation model and is based on k-ω formulation but assimilates desired attributes from both the k-ε and k-ω models with the use of a blending function. The blending function activates the Wilcox k-ω model close to the surface boundary, whilst the k-ε is used on the outer flow. Such a model allows the high performance of the Wilcox model, which is good with flow separation simulation, to be used near the wall without the appearance of its sensitivity to free stream conditions. A detailed investigation into the k-ω SST model and its formulation is presented in [29] and the full formulation as employed in Tdyn is presented in [19].

B.2.4 Boundary Layer Modelling

For wall-attached turbulent boundary layers, the normal gradients in some flow variables become very large as we approach the surface. Some of the turbulence models mentioned in Section B.2.3 start to become unable to model the viscosity dominated region close to the surface known as the viscous sub-layer and would also need to be modified to account for low

Reynolds number effects[1]. Moreover, even if the models were to be accurate in this region, a high mesh element density is required and makes the calculation possible but infeasible. Tdyn uses Law of the Wall functions to deal with these problems by using a logarithmic relationship between U^+ (the non-dimensional time averaged velocity parallel to the wall) and y^+ (the non-dimensional normal distance to the wall) in the log law region (see Figure B.2.1 below). The logarithmic relationship is given by:

$$U^+ = \frac{1}{\kappa}\ln y^+ + B \qquad \text{Equation B.8}$$

where κ is the von Kármán constant and B is a dimensionless constant. Correlation of experimental data by Coles and Hirst (1969) set the values of these two parameters to be $\kappa \approx 0.41$ and $B \approx 5.0$.

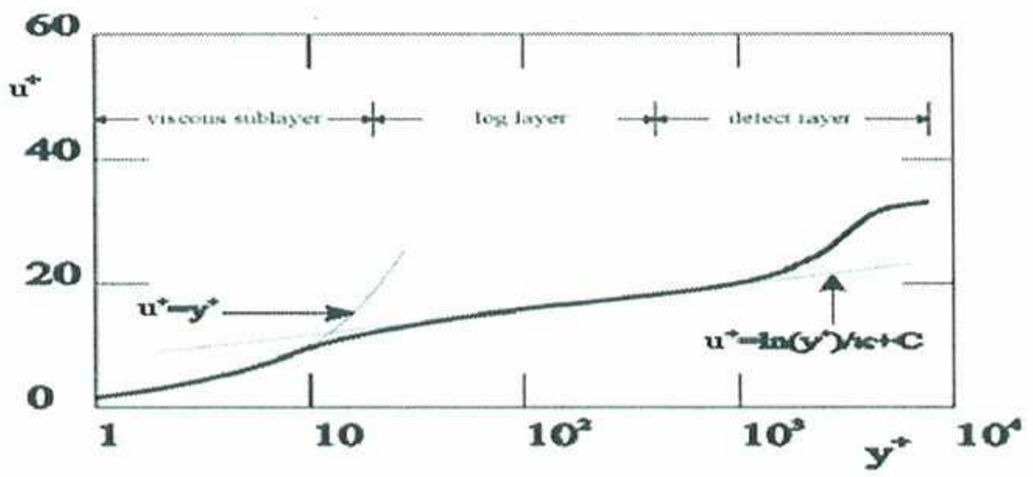


Figure B.2.1 Boundary Layer Regimes

The cross-over point between the wall function and the turbulence model is the first node of the mesh outside of the surface. At this point, the y+ value of the node is used to find U^+, through the use of Equation B.8, which is then used as a Dirichlet boundary value to find the turbulence kinetic energy and turbulence dissipation required for the turbulence model calculation of the RANSE equations. For this to produce meaningful results, the y+ value of the first node must lie inside the log law range. This range, when using the YplusWall law of the wall model in Tdyn, is given by the user specified y+ value, from which the wall function is activated, up to an upper limit of approximately $y^+ < 100$. A fine mesh was used to try to ensure that the first node lies inside this region. Also, by starting the calculation above the viscous sub-layer, we avoid the large number of mesh elements required to resolve it (see Figure B.2.2).

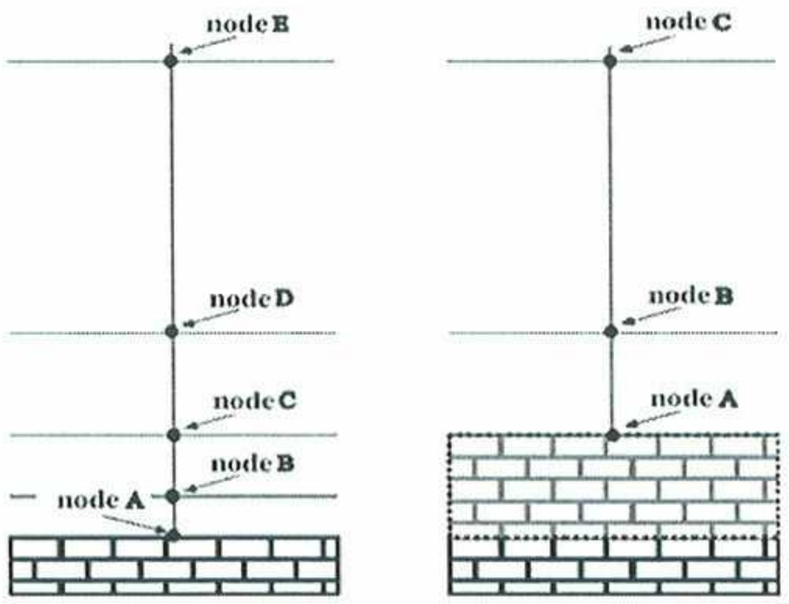


Figure B.2.2 Tdyn law of the Wall modelling on the right of the figure.
Meshing starts level with node A which is inside log law region
and elements to the order of 10^4 may be saved for a 2D aerofoil problem.

[1] Models such as the Spallart-Allmaras and k-ω SST models should ideally be accurate in the viscous sublayer but models such as the k-ε model fail to accurately model separating flows and the high viscous stresses present in this region.

Appendix C: TCL, GiD and Tdyn Modified Files

Appendix C.1: TCL File for Optimisation problem type in GiD

```
############################
#
# is executed when user presses "Run Optima"
# first, Optima is executed in a different thread
# then we keep testing to see if new coords file
has been generated yet
# when it exists, execute loop.bch
#
#
#####################################################
############################

proc Optima { dir } {

    exec $dir/start.sh $dir &
    set counter 2
    set filegenerated [file exists
"$dir/coords$counter.bch"]

    while {$counter<9999999} {
      while {$filegenerated<1} {
          set filegenerated [file exists
"$dir/coords$counter.bch"]
          if {[file exists
"$dir/optimalGeometry.dat"]>0} { return }
      }

      set filefinished [file size
"$dir/coords$counter.bch"]
      while {$filefinished=="0"} {
          set filefinished [file size
"$dir/coords$counter.bch"]
      }
      loop $dir $counter
      incr counter
      set filegenerated [file exists
"$dir/oords$counter.bch"]
    }
}
```

Appendix C.2: Coords.bch – Batch file for aerofoil geometry

escape escape escape geometry create
nurbsline

1	0.00000
0.8	0.02623
0.6	0.04563
0.4	0.05803
0.15	0.05345
0.075	0.04200
0.05	0.03555
0.03	0.02840
0.001	0.00556
0	0.00000

escape escape escape escape

escape escape escape geometry create
nurbsline

0	0.00000
0.001	-0.00556
0.03	-0.02840
0.05	-0.03555
0.075	-0.04200
0.15	-0.05345
0.4	-0.05803
0.6	-0.04563
0.8	-0.02623
1	0.00000

escape escape escape escape

Appendix C.3: Loop.bch – Batch file for aerofoil geometry, meshing, saving, running

escape escape escape escape utilities variables UseMoreWindows 0 escape escape

```
escape escape escape escape utilities variables UseMoreWindows 0 escape escape
redraw
redraw
escape escape escape escape Files BatchFile
"C:/Documents and Settings/anthony/Escritorio/ant new/optima/coords.bch"
escape escape escape escape

escape escape escape geometry create
line
0           0
-3          0
-3          5
10          5
10          0
1           0
escape escape escape escape

escape escape escape geometry create
line
-3          0
-3          -5
10          -5
10          0
escape escape escape escape

escape escape escape geometry create
IntMultLines
AddToSelection
6
7
10
escape
Yes
AddToSelection
4
3
8
escape
Yes
1
2
3
escape
Yes
1
2
7
escape
Yes
escape
escape escape escape geometry create
NurbsSurface
4
5
6
3
7
1
escape
2
3
7
8
9
10
escape escape escape escape escape

escape escape escape escape data cond assign Line_Fix_Pressure change 0.0
5
6
9
10
escape escape
escape escape
escape escape escape escape
escape escape escape escape
escape escape escape escape data cond assign Line_Fix_Velocity change 1 68.0 1 0.0
4
8
escape escape
escape escape
escape escape escape escape
escape escape escape escape
escape escape escape escape data mat assign Air_0oC Surfaces
2
1
escape escape

escape escape escape escape data mat assign Fluid_Body Lines
1 2
escape escape
```

```
escape escape escape escape Meshing AssignSizes Lines
0.01
1 2
escape
escape
escape escape escape escape Meshing AssignSizes Points
0.001
1 3
escape
escape


escape escape escape escape utilities variables SizeTransitionsFactor 0.5 escape escape
escape escape escape escape Meshing MeshCriteria Mesh Surfaces 1: escape
escape escape escape escape Meshing MeshCriteria Mesh Lines 1: escape
escape escape escape escape Meshing Generate 2.5
redraw
escape escape escape escape Meshing MeshView


escape escape escape escape Files SaveAs
"C:/Documents and Settings/anthony/Escritorio/run2.gid"

escape

escape escape escape escape
escape escape escape escape Utilities Calculate escape
```

Appendix C.4: Tdyn2D-2.bas – Execution file to generate Projectname-1.dat file on every run

```
*set elems(linear)
*loop elems
*if(elemsmat>0)
*if(operation(strcasecmp(ElemsMatProp(0),"Fluid_Body")==0))
*elemsconec(1) *NodesCoord(1)
*endif
*endif
*end elems
```

Appendix C.5: Projectname.prb – Problem definition file for each project

Appendix C.5.1 Projectname.prb – Module "General" - File for problem data such as time increment, output step, run time, results output format

```
TITLE: General
#----------#
QUESTION: Number_of_Steps
VALUE: 200
HELP: Number of steps of the simulation.\nTotal physical time will be Number_of_Steps X
Time_increment.
QUESTION: Time_increment
VALUE: 0.001
HELP: Time increment per time step.\nRecommended value dt = C·L/V \n(being L a
characteristic length and 0.1 < C < 0.01).\nMay be a constant or a funtion of time ( i.e.
interpolate(#0.0,0.1,1.0,0.2#t), see Function editor in the bottom-right corner of the window ).
QUESTION: Time_increment_Units#CB#([s],[ds],[cs],[ms])
VALUE: [s]
QUESTION: Max_Iterations
VALUE: 3
HELP: Maximum number of iterations of the non linear fluid scheme (recommended values 3 -
10).
QUESTION: Output_Step
VALUE: 200
HELP: Each Output_Step time steps the results will be written.
QUESTION: Output_Start
VALUE: 1
HELP: The results will be written each "Output_Step" time steps after "Output_Start" steps.
QUESTION: Results_File
VALUE: Ascii
QUESTION: Initialise_Flow_Field
VALUE: None
QUESTION: Initial_Steps
VALUE: 0
HELP: During first "Initial_Steps" steps some controls are carried out.
QUESTION: Start_Up_Control#CB#(None,Time,Speed,Both)
VALUE: Time
HELP: if marked, during first "Initial_Steps" steps the start up process is smoothed.\nThis can
be done by creating an acceleration on the flow (Speed) or by gradually incrementing the time
step size.
QUESTION: Restart#CB#(On,Off)
```

VALUE: Off
HELP: If marked Yes the restart file is used to define the initial data.\n The Restart file taken is "ProblemName.flavia.rst"
TITLE: Write_Results

Appendix C.5.2 Projectname.prb – Module "Turbulence" - File for turbulence model and data

#----------#
TITLE: TURBULENCE
#----------#
QUESTION:
Turbulence_Model#CB#(Laminar,Mixing_Length,Smagorinsky,Kinetic_Energy,K_Energy_Two_Layers,K_E_High_Reynolds,K_E_Two_Layers,K_E_Lam_Bremhorst,K_E_Launder_Sharma,K_Omega,K_Omega_SST,K_KT,Spalart_Allmaras)
VALUE: K_Omega_SST
HELP: Select the turbulence model to be used in the fluid problem
QUESTION: Fix_Turbulence_On_Bodies#CB#(Yes,No)
VALUE: No
HELP: Mark if you want the value of Epsilon or Omega on Bodies to be fixed automatically
QUESTION: TVisco_Min_Ratio
VALUE: 0.001
HELP: Eddy viscosity ratio taken used to calculate the minimum admissible value./nReference value is the maximum physical viscosity of the problem.
QUESTION: TVisco_Max_Ratio
VALUE: 1.0e5
HELP: Eddy viscosity ratio taken used to calculate the maximum admissible value/nReference value is the maximum physical viscosity of the problem.
QUESTION: KEnergy_Min_Ratio
VALUE: 1.0
HELP: Eddy Kinetic Energy ratio taken used to calculate the minimum admissible value./nReference value is the maximum initial kinetic energy of the problem.
QUESTION: KEnergy_Max_Ratio
VALUE: 1.0e5
HELP: Eddy Kinetic Energy ratio taken used to calculate the maximum admissible value/nReference value is the maximum initial kinetic energy of the problem.
QUESTION: Epsilon_Min_Ratio
VALUE: 1.0
HELP: Epsilon ratio taken used to calculate the minimum admissible value./nReference value is the maximum initial epsilon of the problem.
QUESTION: Epsilon_Max_Ratio
VALUE: 1.0e5
HELP: Epsilon ratio taken used to calculate the maximum admissible value./nReference value is the maximum initial epsilon of the problem.
QUESTION: Omega_Min_Ratio
VALUE: 1.0
HELP: Omega ratio taken used to calculate the minimum admissible value./nReference value is the maximum initial omega of the problem.
QUESTION: Omega_Max_Ratio
VALUE: 1.0e3
HELP: Omega ratio taken used to calculate the maximum admissible value./nReference value is the maximum initial omega of the problem.
QUESTION: K_Tau_Min_Ratio
VALUE: 1.0
HELP: K_Tau ratio taken used to calculate the minimum admissible value./nReference value is the maximum initial ktau of the problem.
QUESTION: K_Tau_Max_Ratio
VALUE: 1.0e5
HELP: K_Tau ratio taken used to calculate the maximum admissible value./nReference value is the maximum initial ktau of the problem.
QUESTION: Advanced#CB#(More,Less)
HELP: Hide or Show more options.
DEPENDENCIES:(Less,HIDE,EddyKEnergy_Production_Limit,#CURRENT#,HIDE,Epsilon_Production_Limit,#CURRENT#,HIDE,Epsilon_Reaction_Limit,#CURRENT#,HIDE,Omega_Production_Limit,#CURRENT#,HIDE,Omega_Reaction_Limit,#CURRENT#,HIDE,EddyViscoT_Production_Limit,#CURRENT#,HIDE,EddyViscoT_Reaction_Limit,#CURRENT#)(More,RESTORE,EddyKEnergy_Production_Limit,#CURRENT#,RESTORE,Epsilon_Production_Limit,#CURRENT#,RESTORE,Epsilon_Reaction_Limit,#CURRENT#,RESTORE,Omega_Production_Limit,#CURRENT#,RESTORE,Omega_Reaction_Limit,#CURRENT#,RESTORE,EddyViscoT_Production_Limit,#CURRENT#,RESTORE,EddyViscoT_Reaction_Limit,#CURRENT#)
VALUE: Less
QUESTION: EddyKEnergy_Production_Limit
VALUE: 20.0
HELP: Maximum ratio between the Eddy kinetic energy production and reaction term. This limiter may prevent the unrealistic buildup of eddy viscosity in the stagnation region of the bodies. Recommended value is 20.0.
QUESTION: Epsilon_Production_Limit
VALUE: 2000.0
HELP: Maximum ratio between the Epsilon production and reaction term.
QUESTION: Epsilon_Reaction_Limit
VALUE: 2000.0
HELP: Maximum ratio between the Epsilon reaction and production term.
QUESTION: Omega_Production_Limit
VALUE: 2000.0
HELP: Maximum ratio between the Omega production and reaction term.
QUESTION: Omega_Reaction_Limit
VALUE: 2000.0
HELP: Maximum ratio between the Omega reaction and production term.
QUESTION: EddyViscoT_Production_Limit

VALUE: 2000.0
HELP: Maximum ratio between the Spallart-Almarax model production and reaction term.
QUESTION: EddyViscoT_Reaction_Limit
VALUE: 2000.0
HELP: Maximum ratio between the Spallart-Almarax model reaction and production term.

Appendix C.5.3 Projectname.prb - Book "Initial" - File for initial data

BOOK: INITIAL
TITLE: RANSOL
QUESTION: Initial_Pressure
VALUE: 0.0
QUESTION: Pressure_FuncCond
VALUE: 0
QUESTION: Initial_VelocityX
VALUE: 68.0
QUESTION: VelocityX_FuncCond
VALUE: 0
QUESTION: Initial_VelocityY
VALUE: 0.0
QUESTION: VelocityY_FuncCond
VALUE: 0
QUESTION: Initial_EddyKEner
VALUE: 0.01
QUESTION: Initial_EddyLength
VALUE: 0.01
Appendix 4 Mat file for material properties and fluib boundary conditions

Appendix C.6: Projectname.mat – Material default settings for fluid (Antair) and aerofoil surface fluid boundary/ wall function

Appendix C.6.1 Projectname.mat – Book "Materials(Fluid)" - File for fluid material properties

BOOK:MATERIALS_(FLUID)
NUMBER: 1 MATERIAL Antair
TITLE: RANSOL
QUESTION: Type#CB#(Fluid)
VALUE: Fluid
QUESTION: Density
VALUE: 1.225
QUESTION: Density_Units
VALUE: [Kg/m3]
QUESTION: Viscosity
VALUE: 1.736e-05
QUESTION: Viscosity_Units
VALUE: [Kg/m.s]
QUESTION: Compressibility
VALUE: 0.0
QUESTION: Compressibility_Units
VALUE: [s2/m2]
QUESTION: AccOX_Field
VALUE: 0.0
QUESTION: AccOY_Field
VALUE: 0.0
QUESTION: Acceleration_Units
VALUE: [m/s2]

Appendix C.6.2 Projectname.mat – Module "Ransol" - File for fluid boundary/ wall function properties

TITLE: RANSOL
QUESTION: Type#CB#(Fluid_Body)
VALUE: Fluid_Body
STATE: HIDDEN
QUESTION: SWITCH#CB#(0,1)
VALUE: 1
QUESTION: BoundType
VALUE: YplusWall
QUESTION: Yplus
VALUE: 32
QUESTION: Delta
VALUE: 1.0
QUESTION: Delta_Units
VALUE: [cm]
QUESTION: Cw
VALUE: 0.04
QUESTION: Roughness
VALUE: 100
QUESTION: Roughness_Units
VALUE: [um]
QUESTION: FTau_Field
VALUE: 0.0

QUESTION: Kenr_Field
VALUE: 0.0
QUESTION: Elen_Field
VALUE: 0.0
QUESTION: VelX_Field
VALUE: 0.0
QUESTION: VelY_Field
VALUE: 0.0
QUESTION: VelZ_Field
VALUE: 0.0
QUESTION: VelN_Field
VALUE: 0.0
QUESTION: Sharp_Angle
VALUE: 20.0
HELP: Maximum angle to apply (automatically) Kutta-Jukowsky condition (recommended 20o)
QUESTION: Fix_Angle
VALUE: 100.0
HELP: Maximum angle to fix (automatically) the velocity (recommended 100o)
QUESTION: SternC_Angle
VALUE: 60.0
HELP: Maximum angle between velocity and hull in the floating line.\nIt is used to control the stern flow conditions (recommended 60º)
DEPENDENCIES:(0.0,HIDE,SternC_Angle,0.0)

Appendix C.7: 3D.bch – Batch file to create 3D geometry for Problem 3

```
escape escape escape escape utilities variables UseMoreWindows 0 escape escape
redraw
escape escape escape escape Files BatchFile
"C:/Documents and Settings/anthony/Escritorio/Anthony/ant new/optima3D2/naca0012.bch"
escape escape escape escape

escape escape escape geometry create
line
0          0
-1.5       0
-1.5       1
3          1
3          0
1          0
escape escape escape escape

escape escape escape geometry create
line
-1.5       0
-1.5       -1
3          -1
3          0
escape escape escape escape

escape escape escape geometry create
line
-1.5       1          0
-1.5       1          4.5
-1.5       0          4.5
-1.5       0          0
escape escape escape escape

escape escape escape geometry create
line
-1.5       0          4.5
-1.5       -1         4.5
-1.5       -1         0
escape escape escape escape

escape escape escape geometry create
line
3          1          0
3          1          4.5
3          0          4.5
3          0          0
escape escape escape escape

escape escape escape geometry create
line
3          0          4.5
3          -1         4.5
3          -1         0
escape escape escape escape

escape escape escape geometry create
line
-1.5       1          0
-1.5       1          4.5
-1.5       0          4.5
-1.5       0          0
escape escape escape escape
```

```
escape escape escape geometry create
line
-1.5        1        4.5
3           1        4.5
escape
-1.5        0        4.5
3           0        4.5
escape
-1.5        -1       4.5
3           -1       4.5
escape escape escape escape

escape escape escape escape utilities Copy AllTypes Duplicate MaintainLayers Translation
FNoJoin 0.0,0.0,0.0 FNoJoin 0.0,0.0,3
points 1 3 lines 1 2 surfaces volumes dimensions
escape escape
escape escape

escape escape escape escape utilities Move AllTypes MaintainLayers Scale FNoJoin 0.0,0.0,3
FNoJoin 0.5,0.5,0.0
points 39 40 lines 27 28 surfaces volumes dimensions
escape escape
escape escape

escape escape escape escape utilities Move AllTypes MaintainLayers Translation FNoJoin
0.0,0.0,3 FNoJoin -1.0,0.0,3
points 39 40 lines 27 28 surfaces volumes dimensions
escape escape
escape escape

escape escape escape escape utilities id
1 3 39 40
escape escape escape geometry create
line
FJoin 39
FJoin 1
escape
FJoin 39
FJoin 40
escape
FJoin 3
FJoin 40
escape escape escape escape
escape escape escape escape geometry create IntMultLines
4 5 11 21
escape
Yes
5 6 16
escape
Yes
6 7 10 18
escape
Yes
9 10 20
escape
Yes
8 9 15
escape
Yes
3 4 8 13 23
escape
Yes
1 2 3 31
escape
Yes
1 2 7 29
escape
Yes
27 28 30 31
escape
Yes
escape
Yes
27 28 29 30
escape
Yes
11 12 21 22 24
escape
Yes
16 17 24
escape
Yes
17 18 19 25
escape
Yes
19 20 26
escape
Yes
14 15 26
escape
Yes
12 13 14 22 23 25
escape
Yes
```

escape escape escape escape

escape escape escape escape geometry create NurbsSurface
1 3 4 5 6 7
escape
2 3 7 8 9 10
escape
6 16 17 18
escape
10 18 19 20
escape
12 17 24 25
escape
14 19 25 26
escape
4 11 12 13
escape
8 13 14 15
escape
5 11 16 24
escape
9 15 20 26
escape
3 7 13 18 25 29 30 31
escape
27 30
escape
28 30
escape
1 27 29 31
escape
2 28 29 31
escape escape escape escape

escape escape escape escape geometry create volume
2 4 6 8 10 11 13 15
escape
1 3 5 7 9 11 12 14
escape
escape escape escape escape

escape escape escape escape data cond assign Surface_Fix_Pressure change 0.0
1 2 3 4 5 6 9 10
escape escape
escape escape
escape escape escape escape

escape escape escape escape
escape escape escape escape data cond assign Surface_Fix_Velocity change 1 68.0 1 0.0 1 0.0
7 8
escape escape
escape escape
escape escape escape escape
escape escape escape escape
escape escape escape escape data mat assign Fluid Volumes
1 2
escape escape
escape escape
escape escape escape
escape escape escape
escape escape escape escape data mat assign Fluid_Body Surfaces
12 13 14 15
escape escape
escape escape
escape escape escape
escape escape escape

layer new Layer1 escape
layer changename Layer1 wing escape
layer changename Layer0 cv escape
layer on wing escape
layer entities wing LowerEntities All
points 1 2 40 44 lines 1 2 27 28 29 30 31 surfaces 12 13 14 15 volumes dimensions
escape escape
escape escape
layer off cv escape
escape escape escape escape

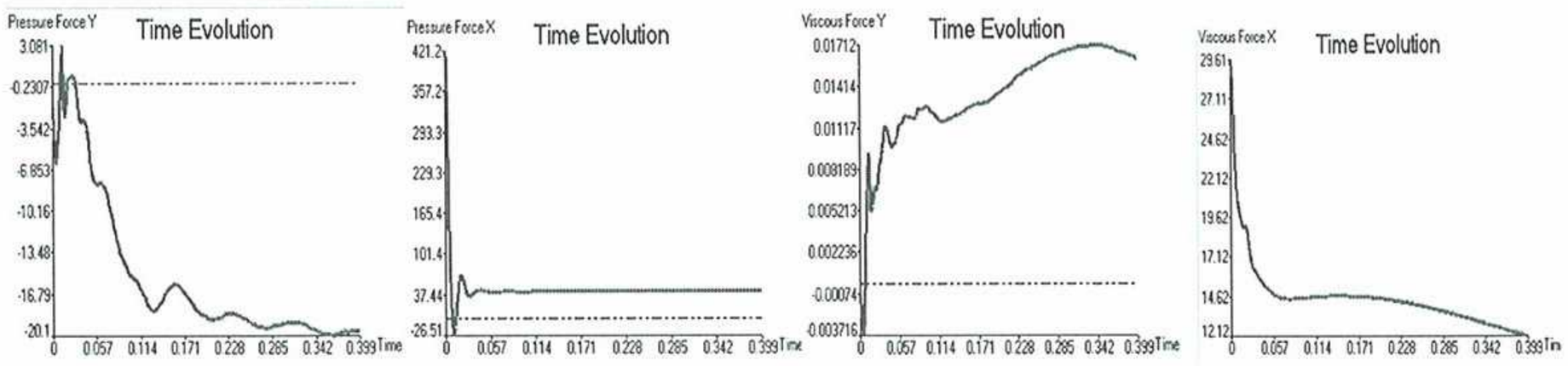
escape escape escape escape Meshing AssignSizes Lines
0.008
27 28 29 30 31
escape escape
escape escape escape escape Meshing AssignSizes Surfaces
0.008
12 13
escape
0.1
14 15
escape
escape
escape escape escape escape

```
escape escape escape escape utilities variables SizeTransitionsFactor 0.2 escape escape
escape escape escape escape Meshing MeshCriteria Mesh Surfaces 1: escape
escape escape escape escape Meshing MeshCriteria Mesh Volumes 1: escape
escape escape escape escape Meshing Generate 0.5
redraw
escape escape escape escape Meshing MeshView
```

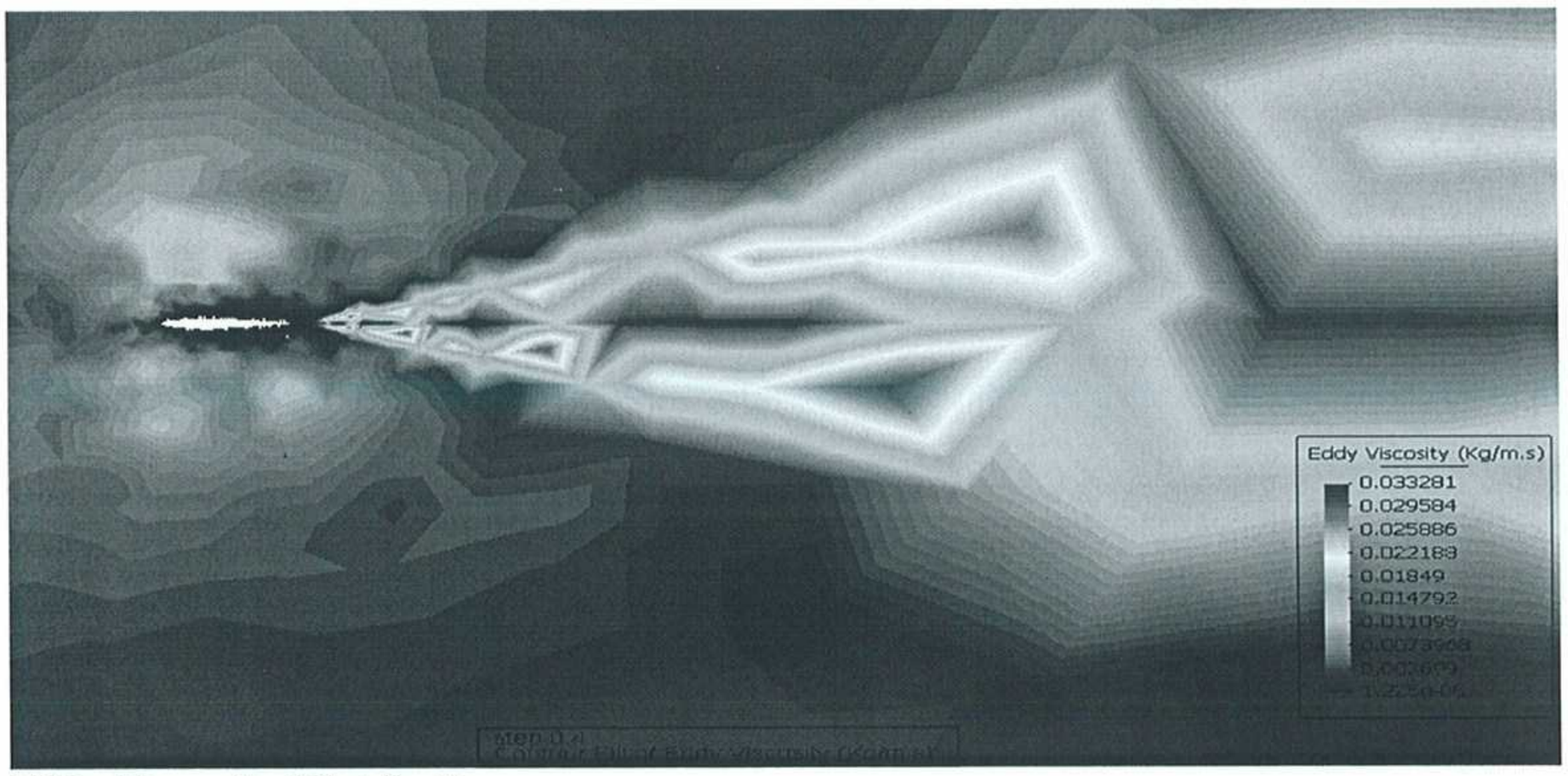
Appendix D: Tdyn Turbulence Models Test Results

The following are the turbulence model test results from Section 2.9.1. Included are the Viscous Force and Pressure Force time responses in the X and Y direction along with Eddy Viscosity and Velocity Distributions around the aerofoil.

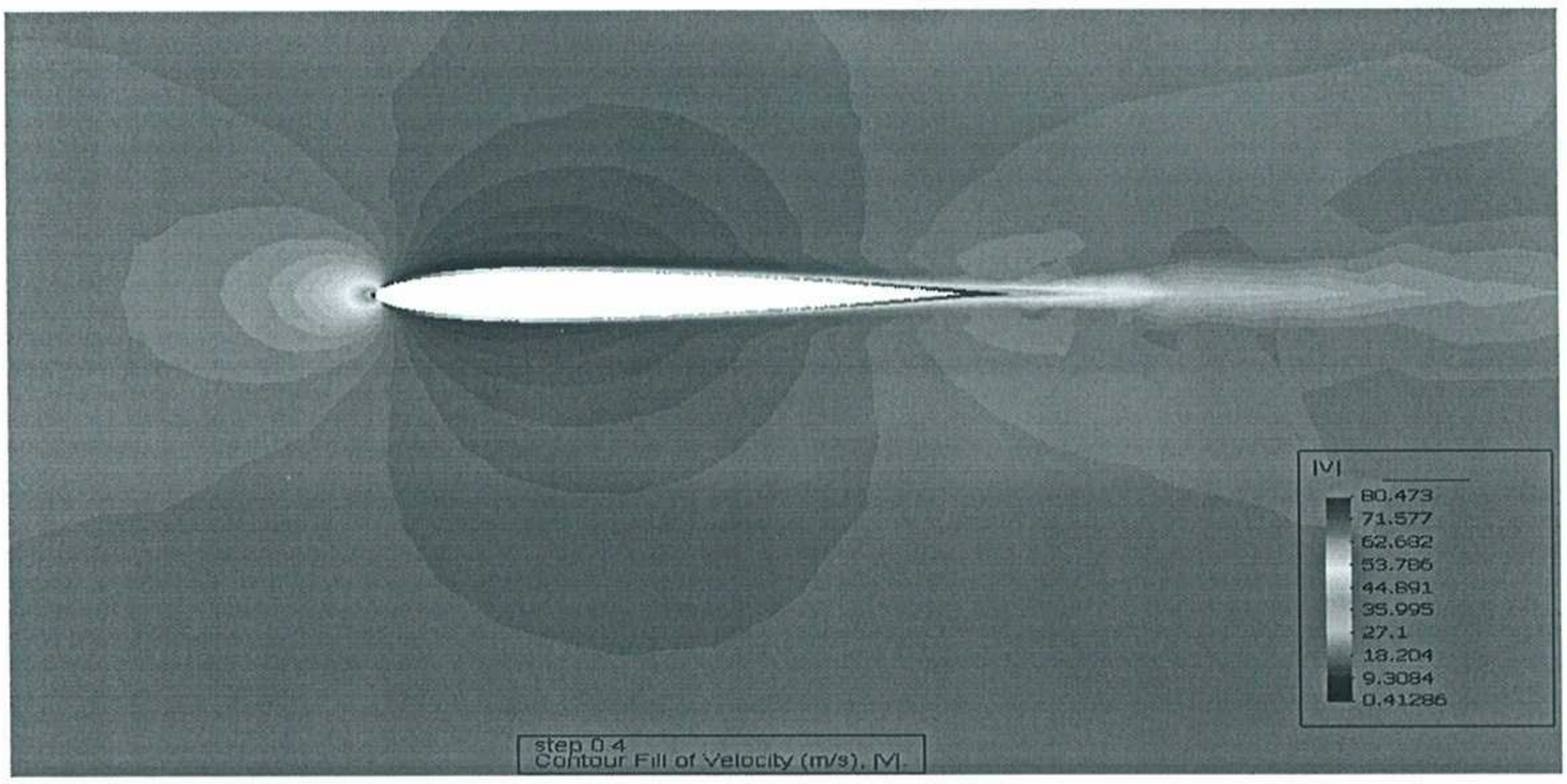
D.1 Smagorinski Model (Smag)



Y and X direction Total Pressure Force response and Y and X direction Viscous Force Response

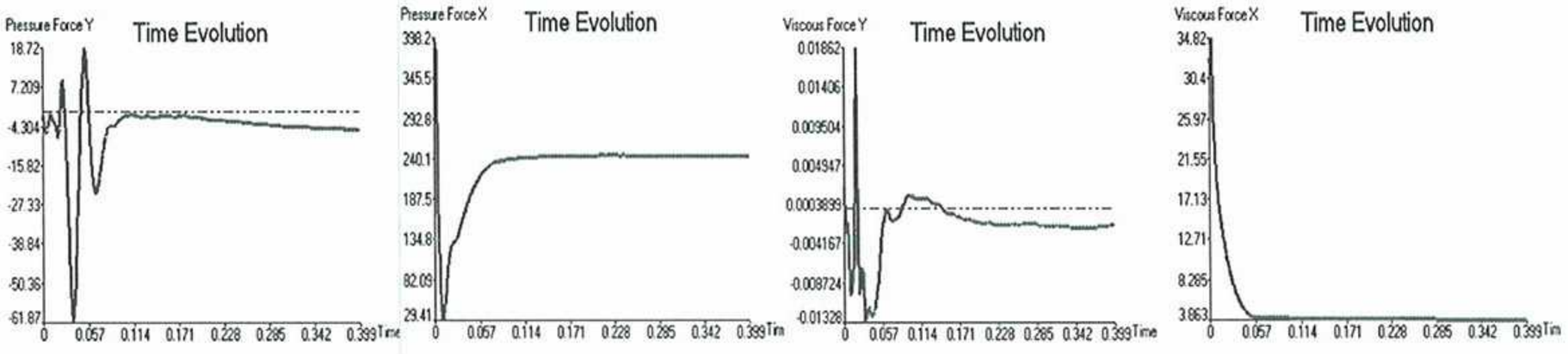


Eddy Viscosity Distribution

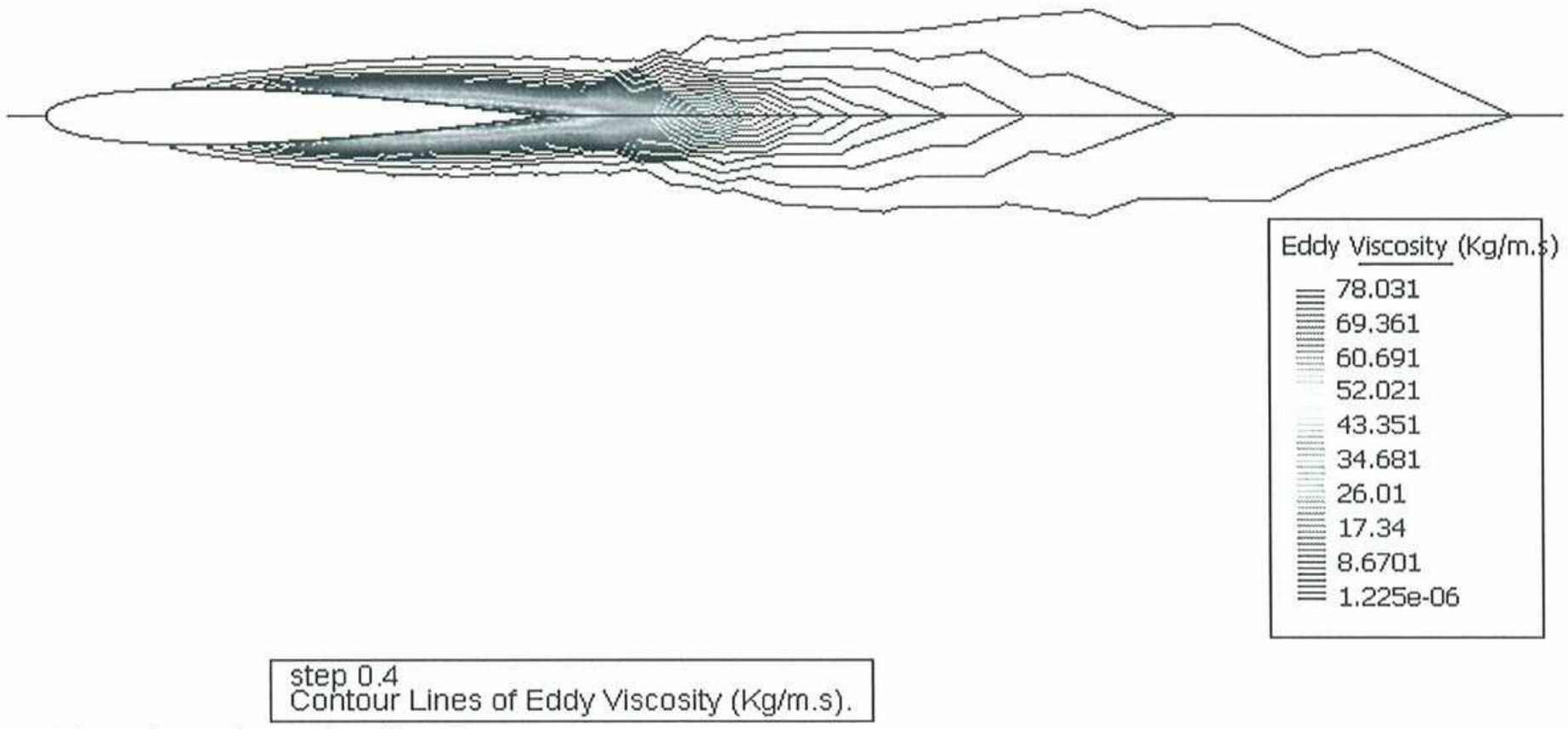


Velocity Distribution

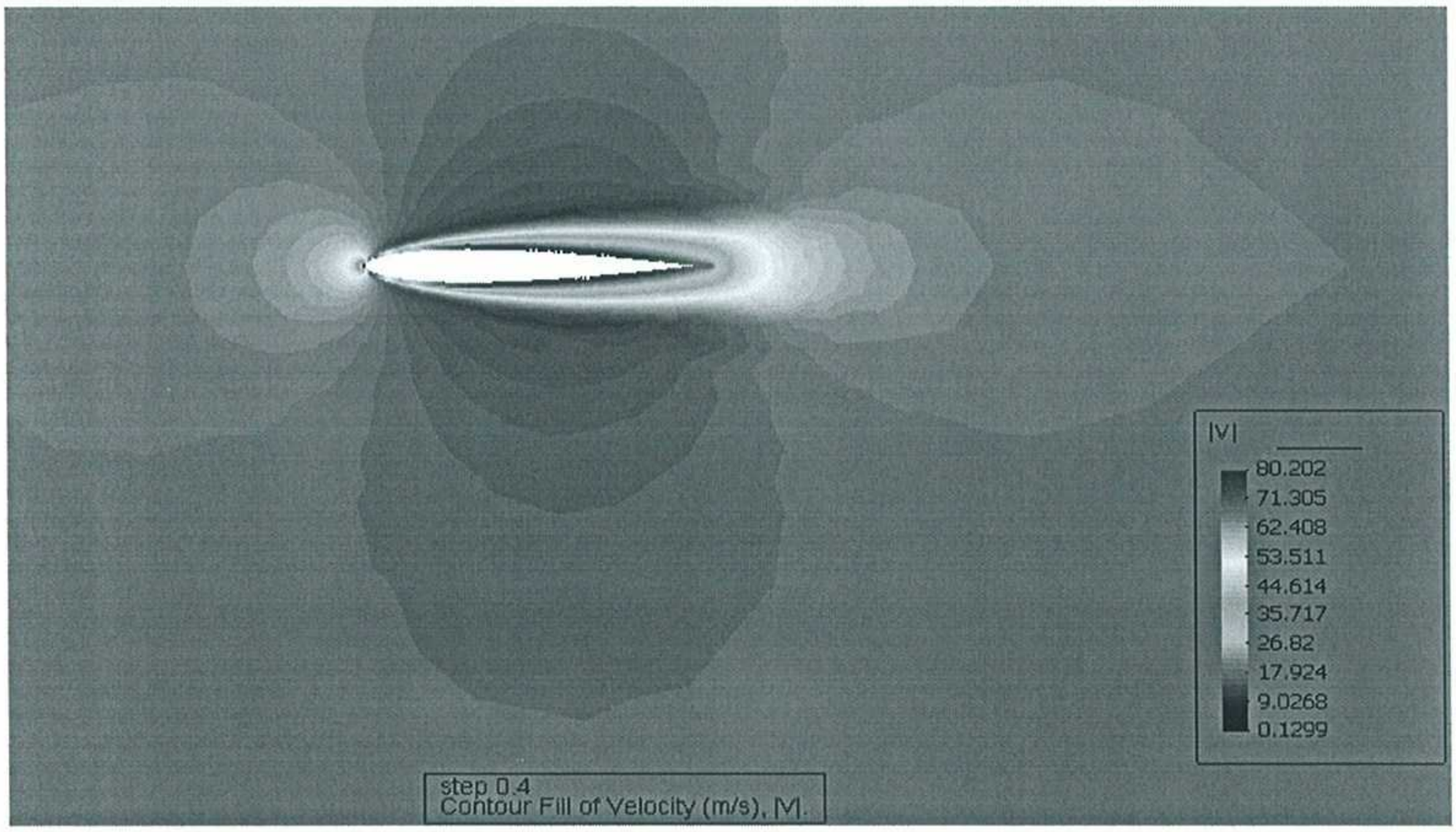
D.2 Kinetic Energy Two Layers (KE2L)



Y and X direction Total Pressure Force response and Y and X direction Viscous Force Response

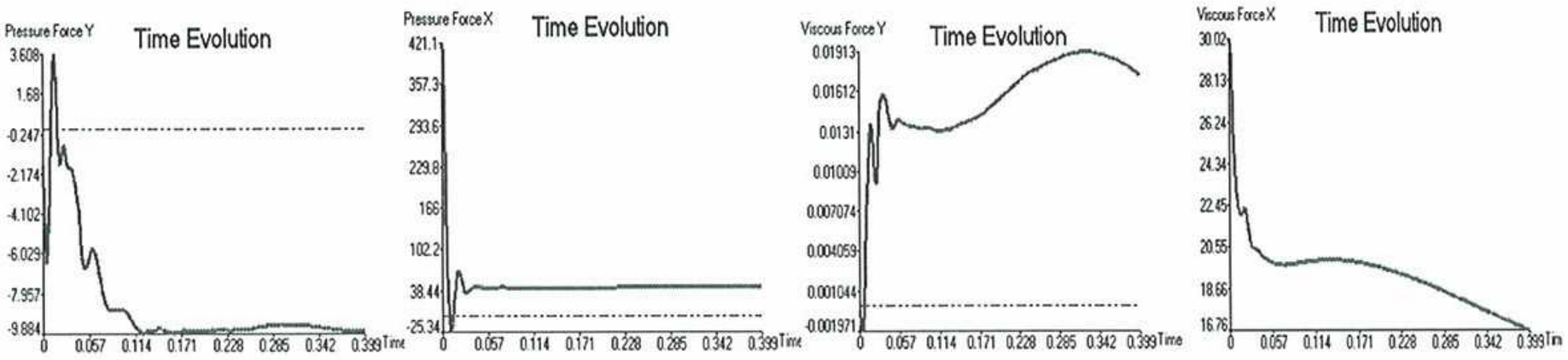


Eddy Viscosity Distribution

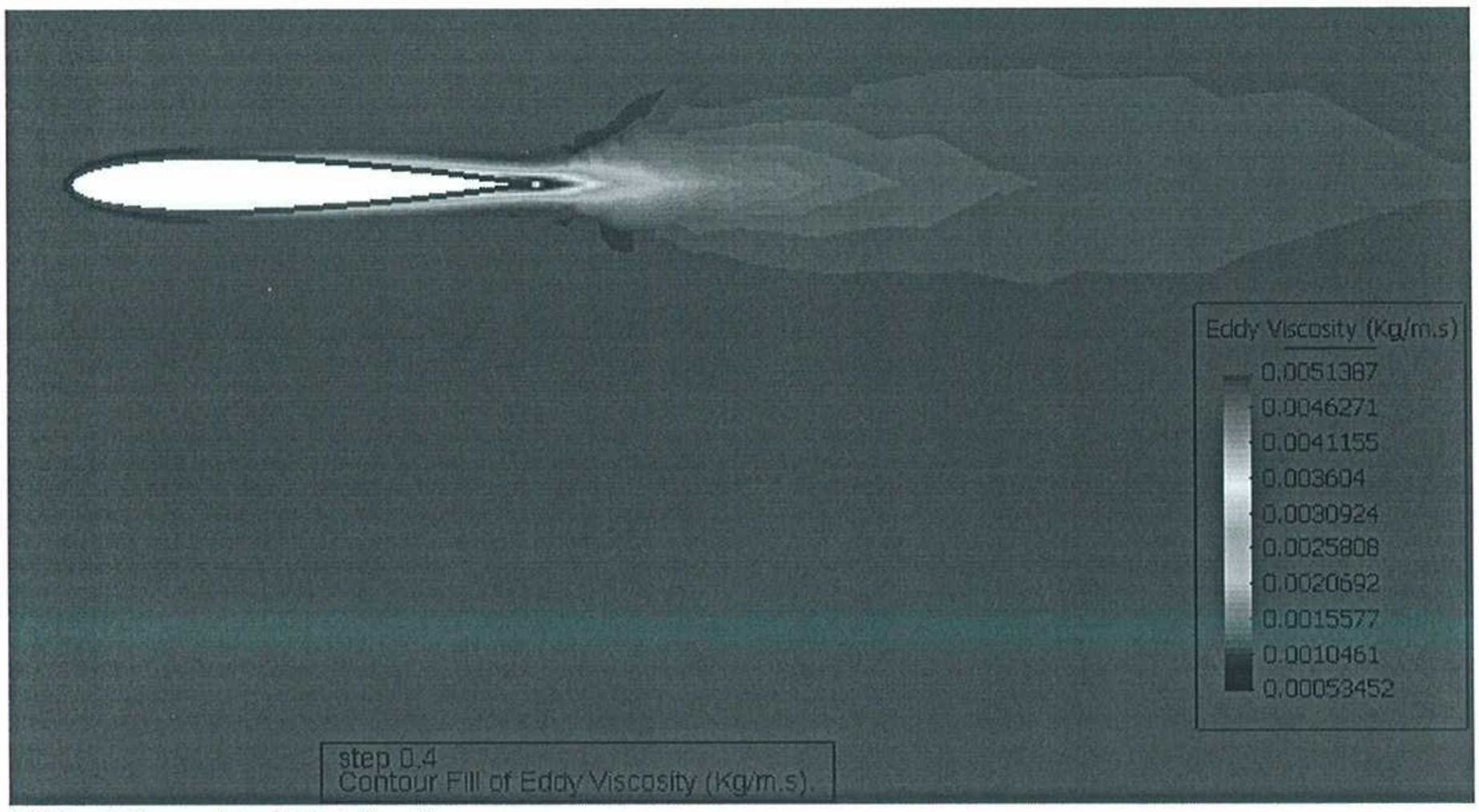


Velocity Distribution

D.3 Spalart-Allmaras Model (SpAl)



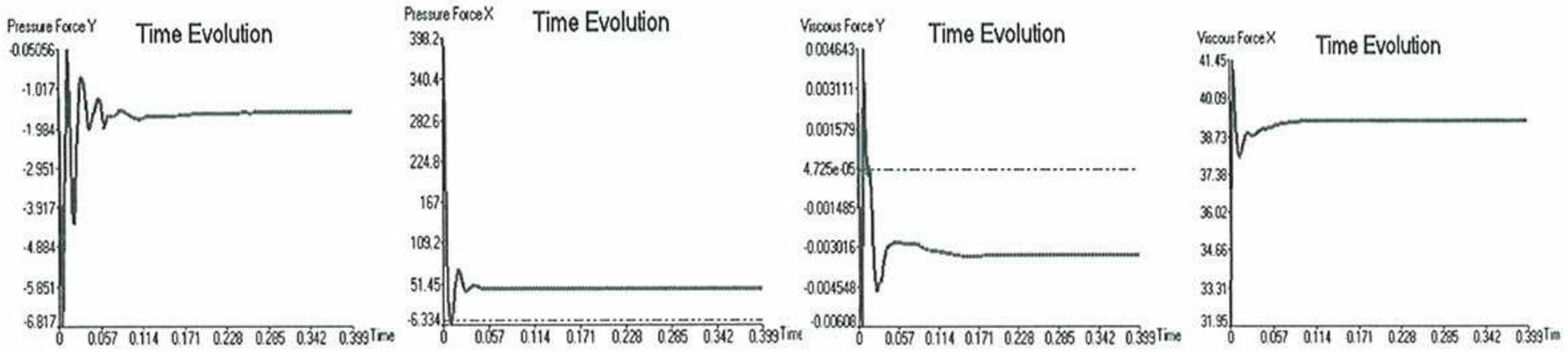
Y and X direction Total Pressure Force response and Y and X direction Viscous Force Response



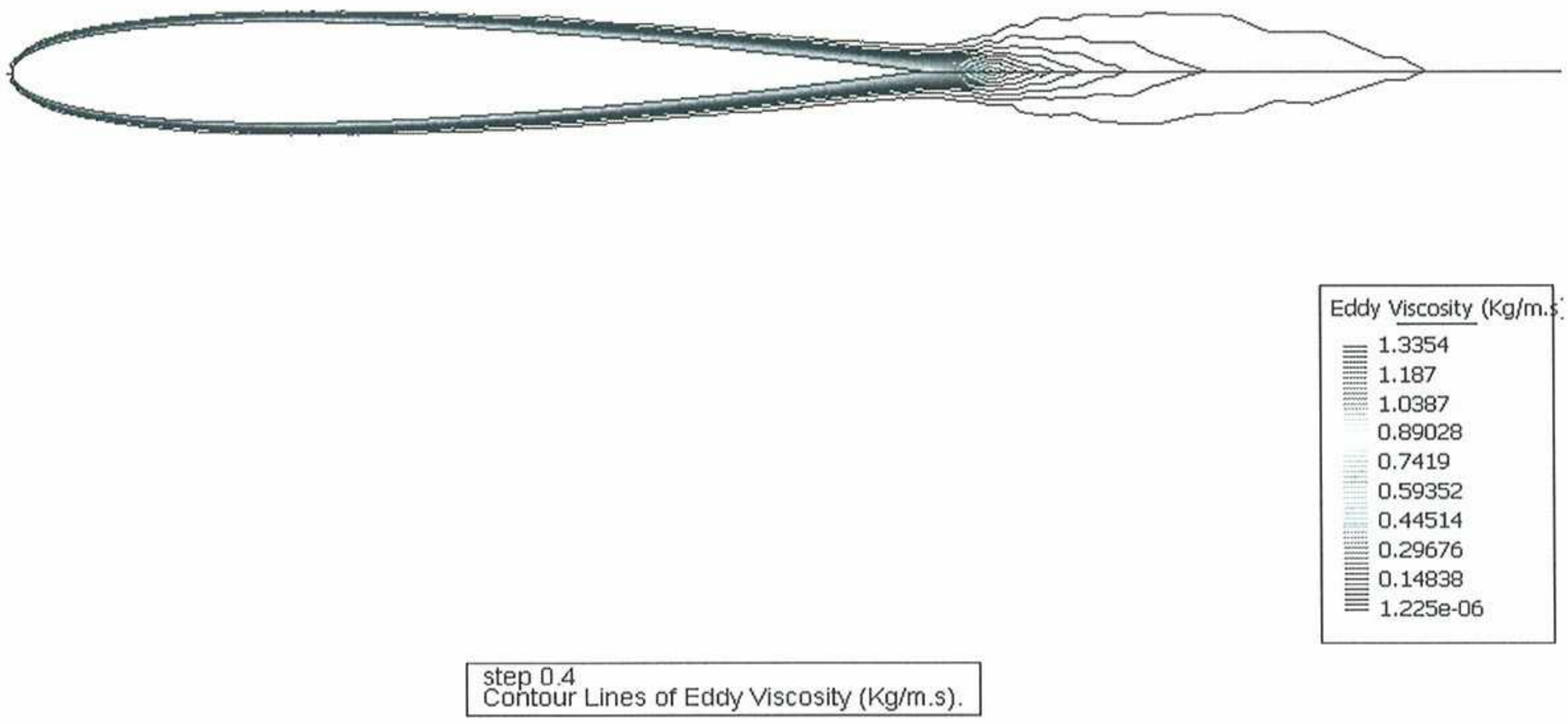
Eddy Viscosity Distribution

Velocity Distribution

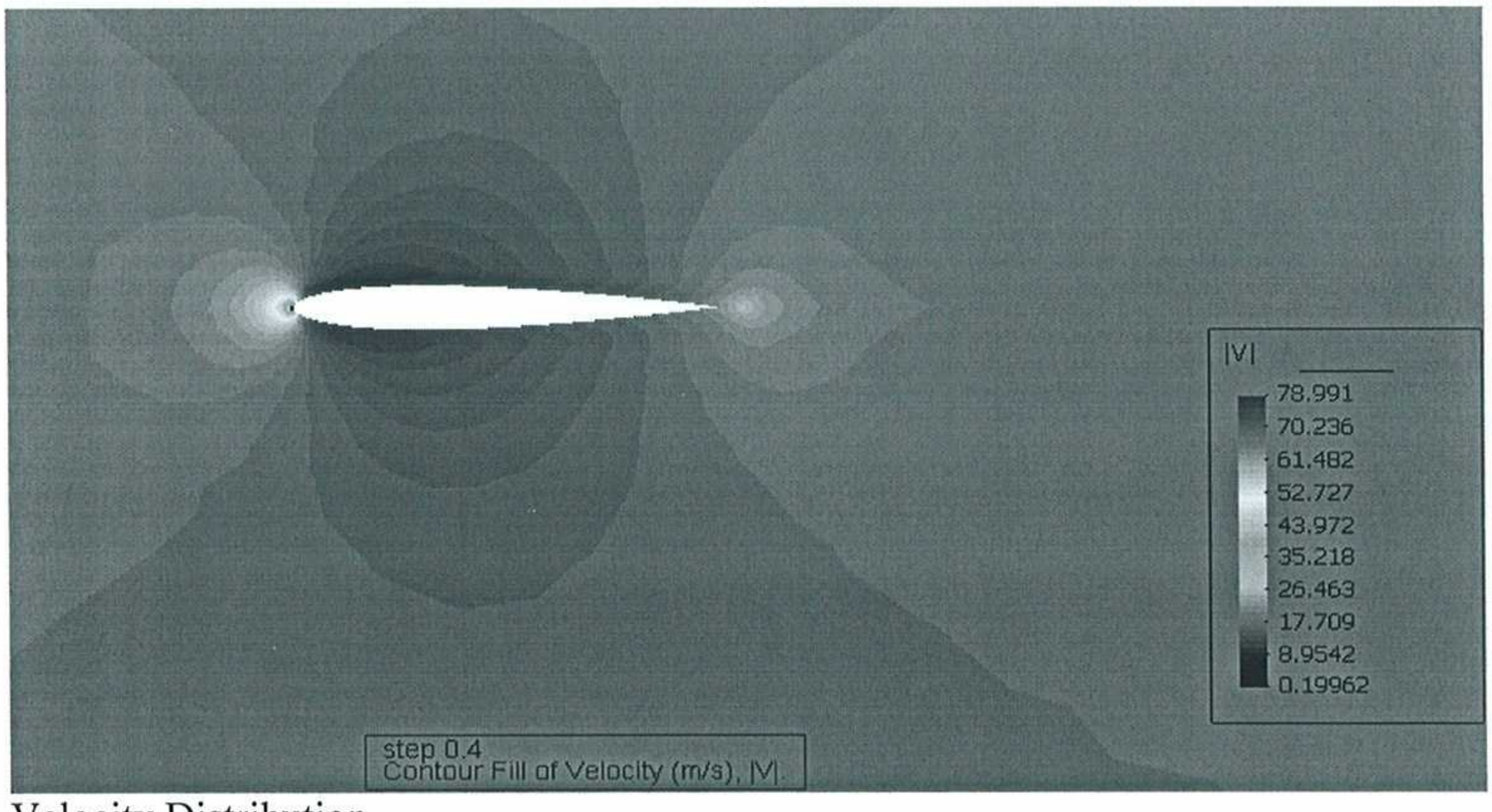
D.4 k-ε Launder Sharma Model (kεLS)



Y and X direction Total Pressure Force response and Y and X direction Viscous Force Response

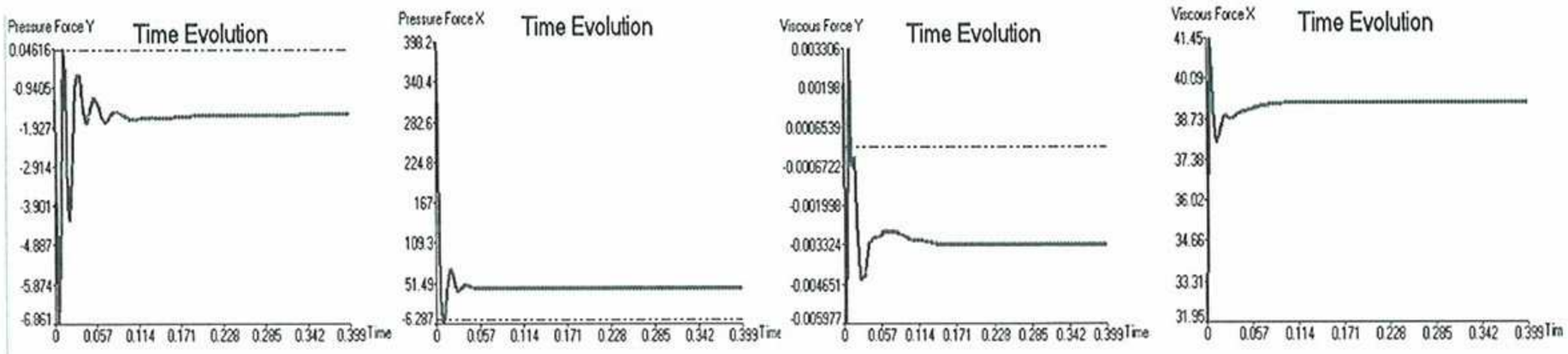


Eddy Viscosity Distribution

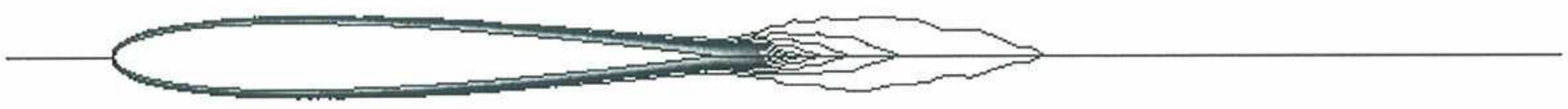


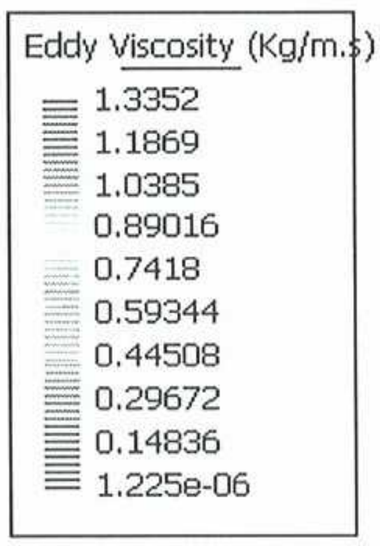
Velocity Distribution

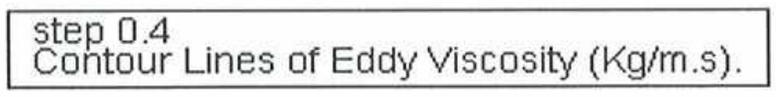
D.5 k-ε High Re Model (KεHRe)



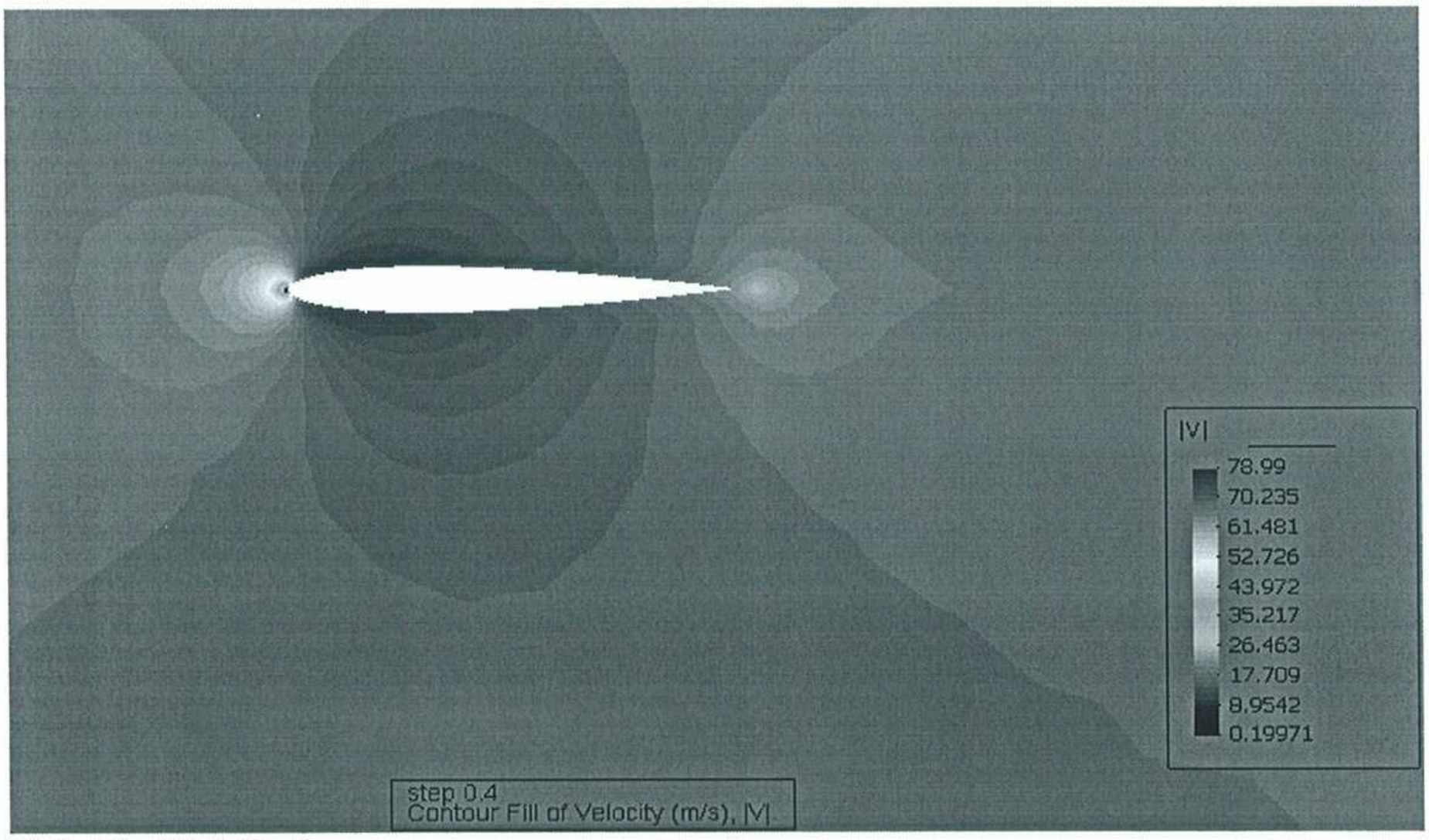
Y and X direction Total Pressure Force response and Y and X direction Viscous Force Response





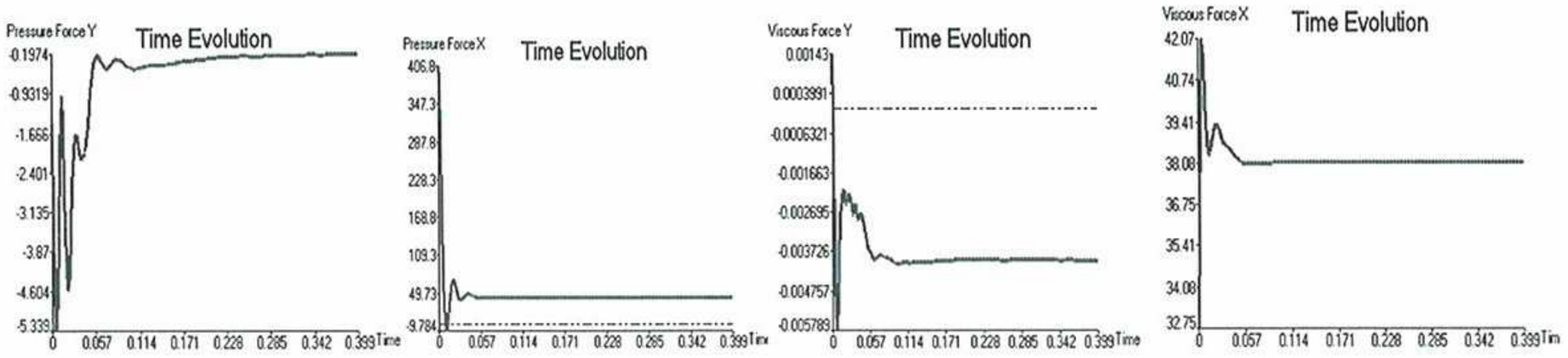


Eddy Viscosity Distribution

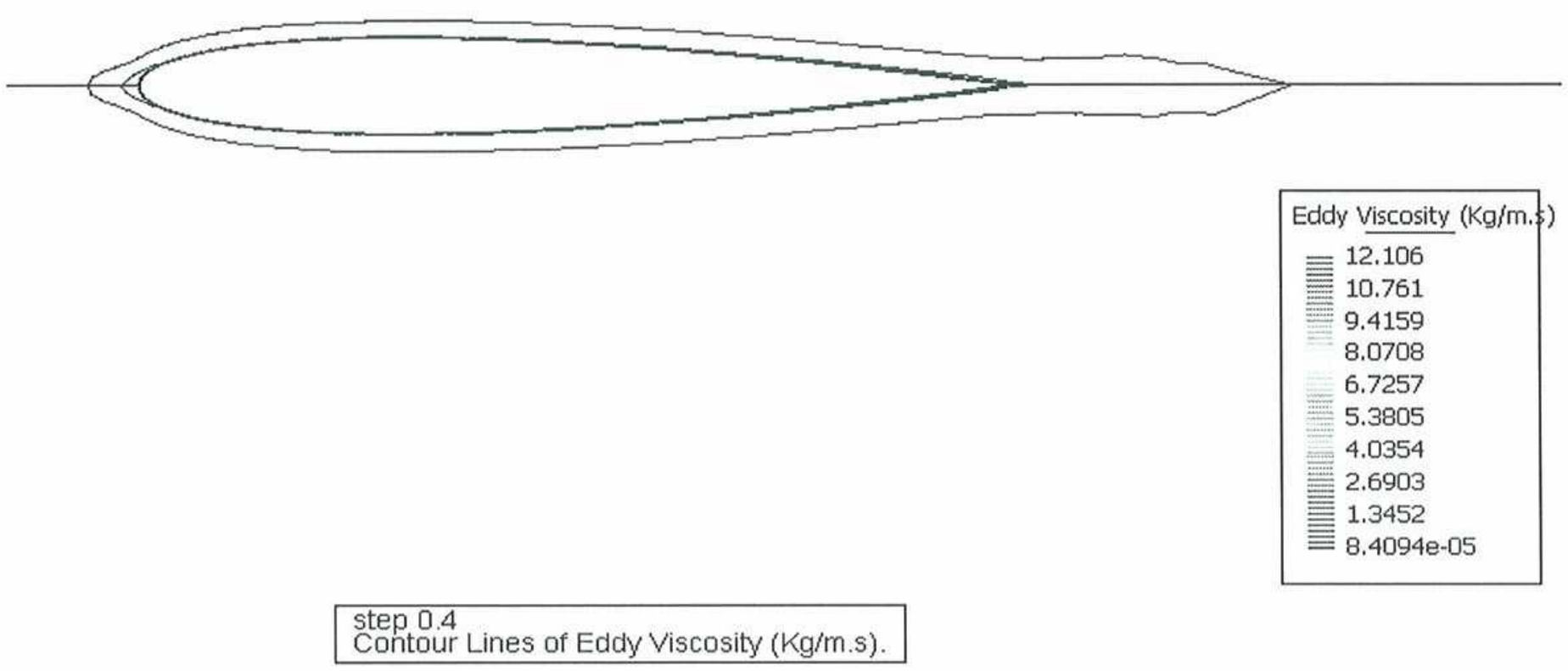


Velocity Distribution

D.6 k-ω Model (kO)



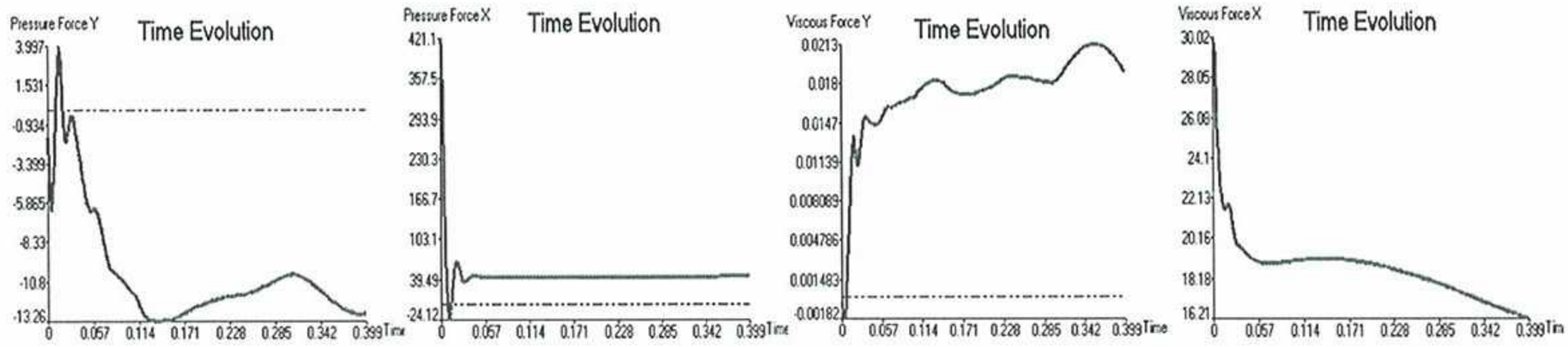
Y and X direction Total Pressure Force response and Y and X direction Viscous Force Response



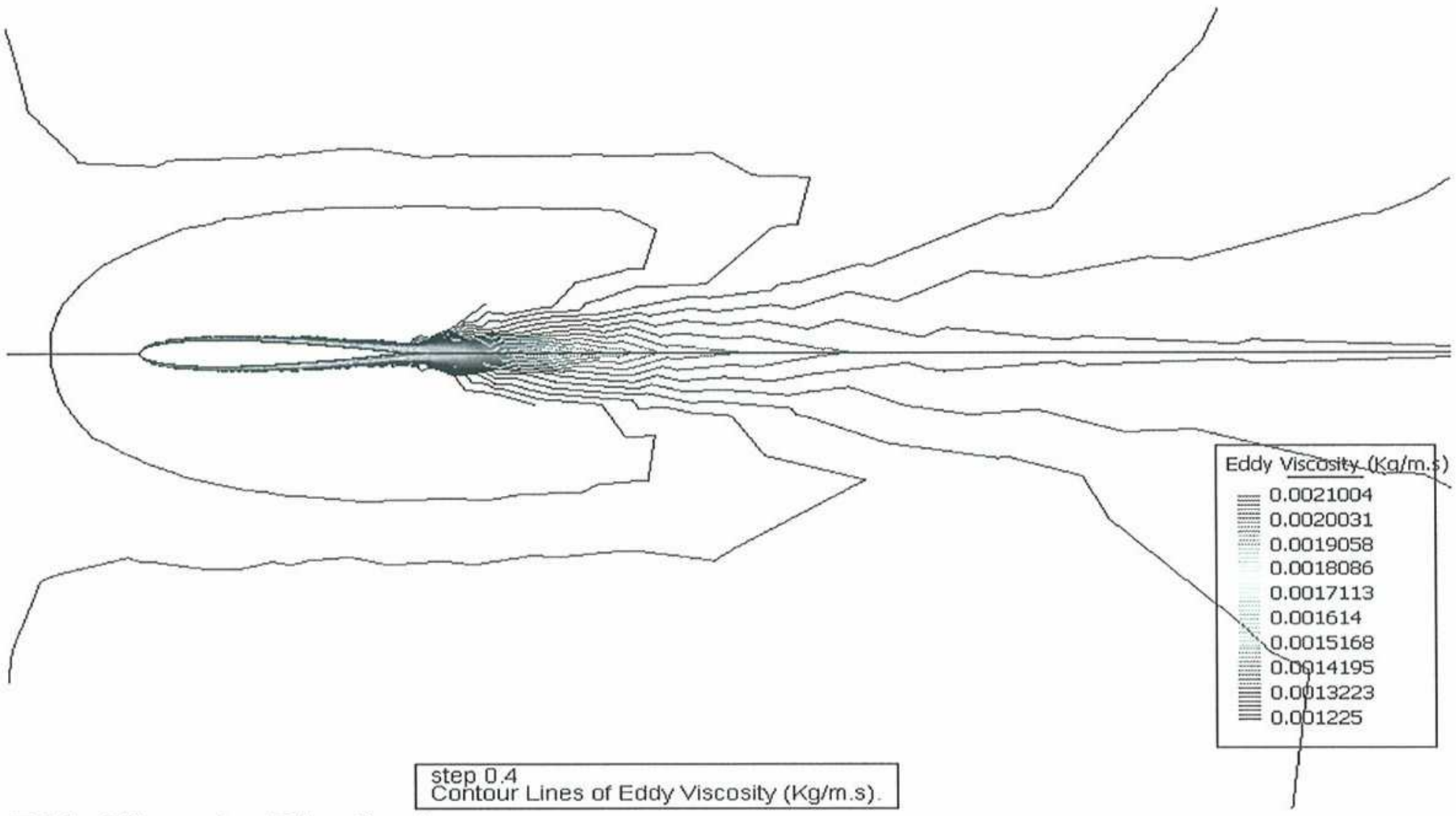
Eddy Viscosity Distribution

Velocity Distribution

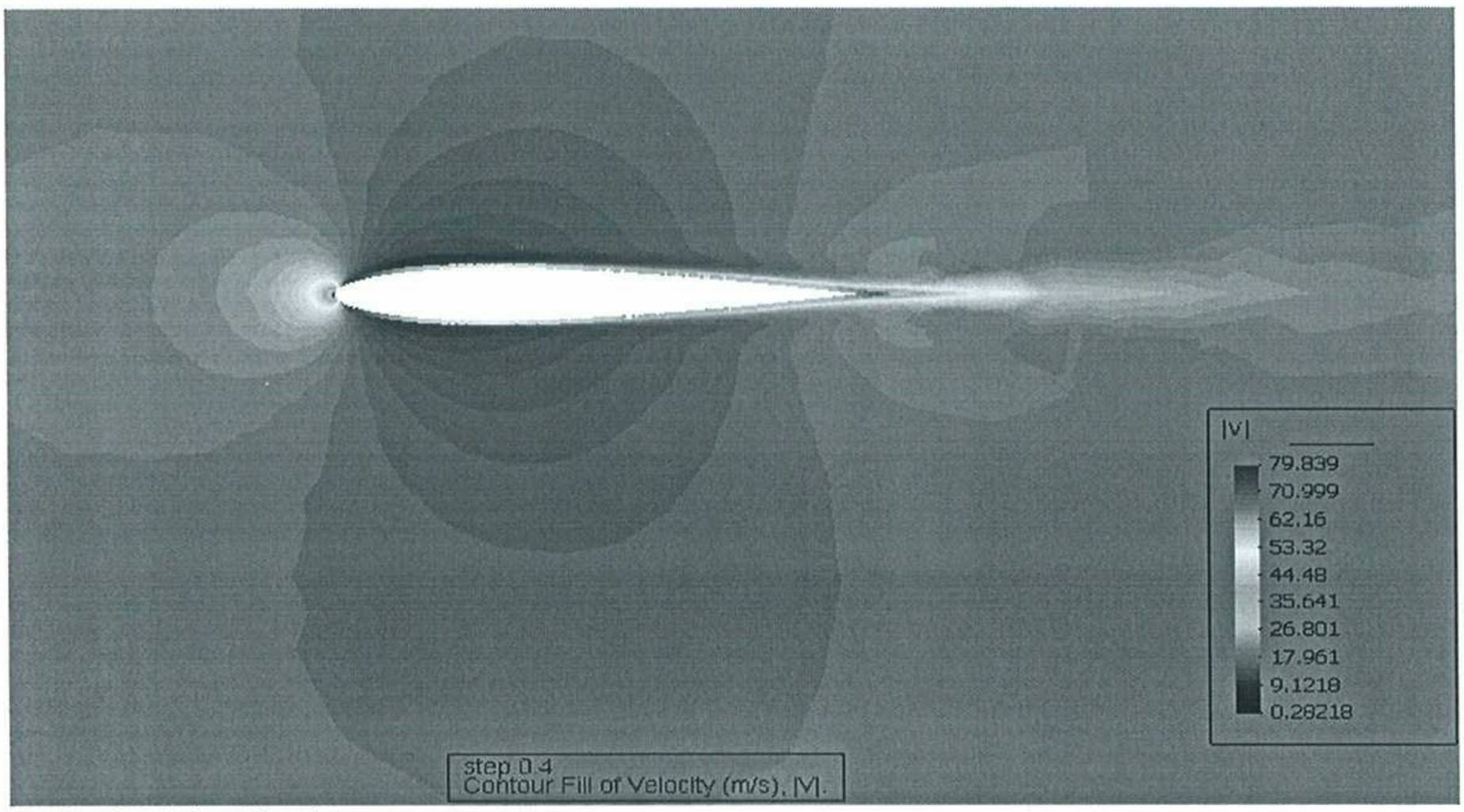
D.7 K-KT model (KKT)



Y and X direction Total Pressure Force response and Y and X direction Viscous Force Response

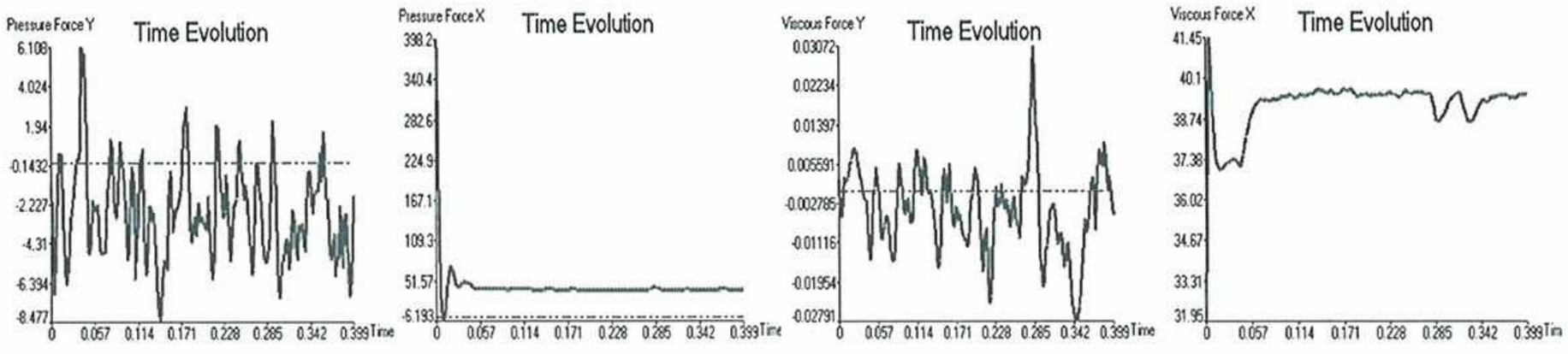


Eddy Viscosity Distribution

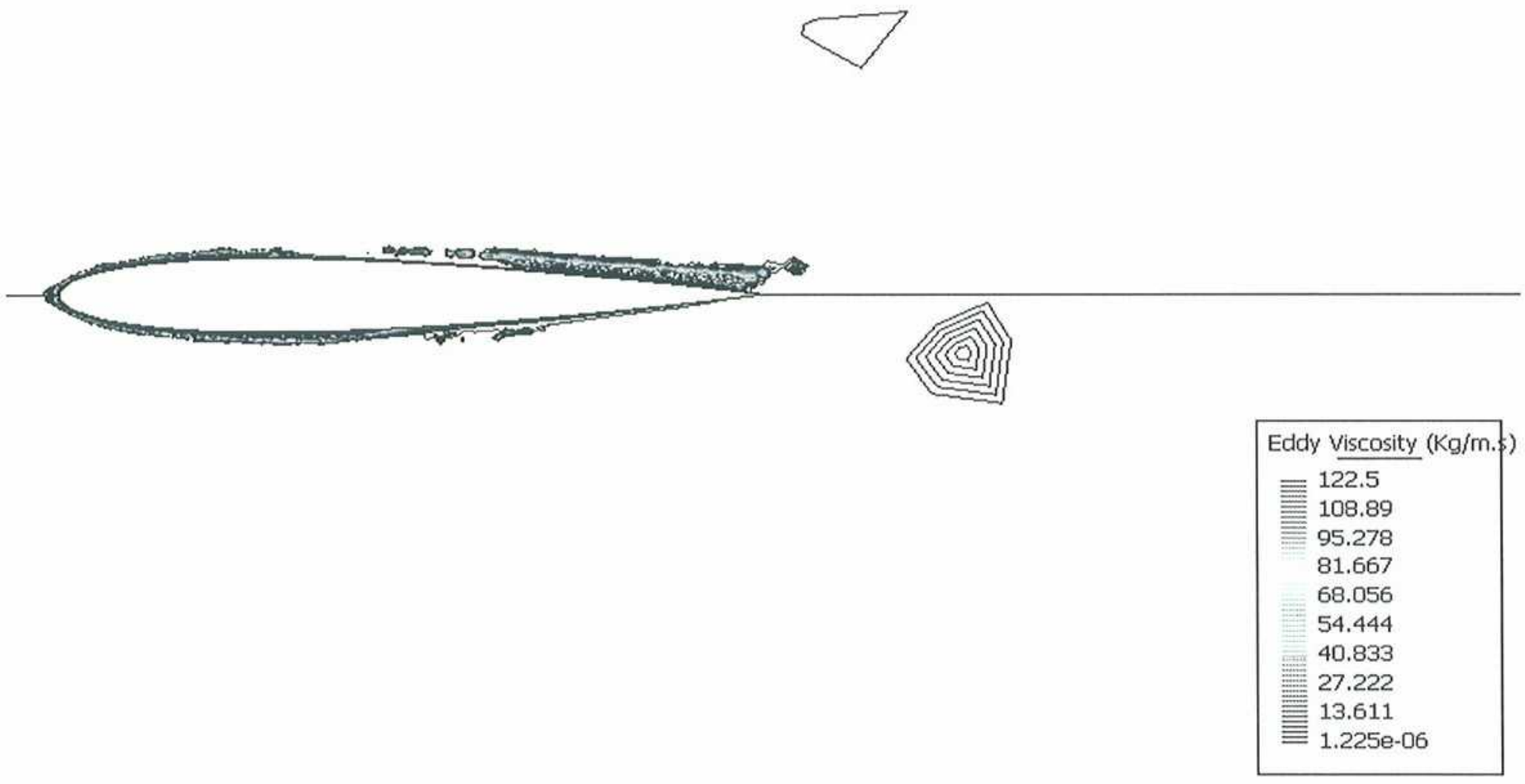


Velocity Distribution

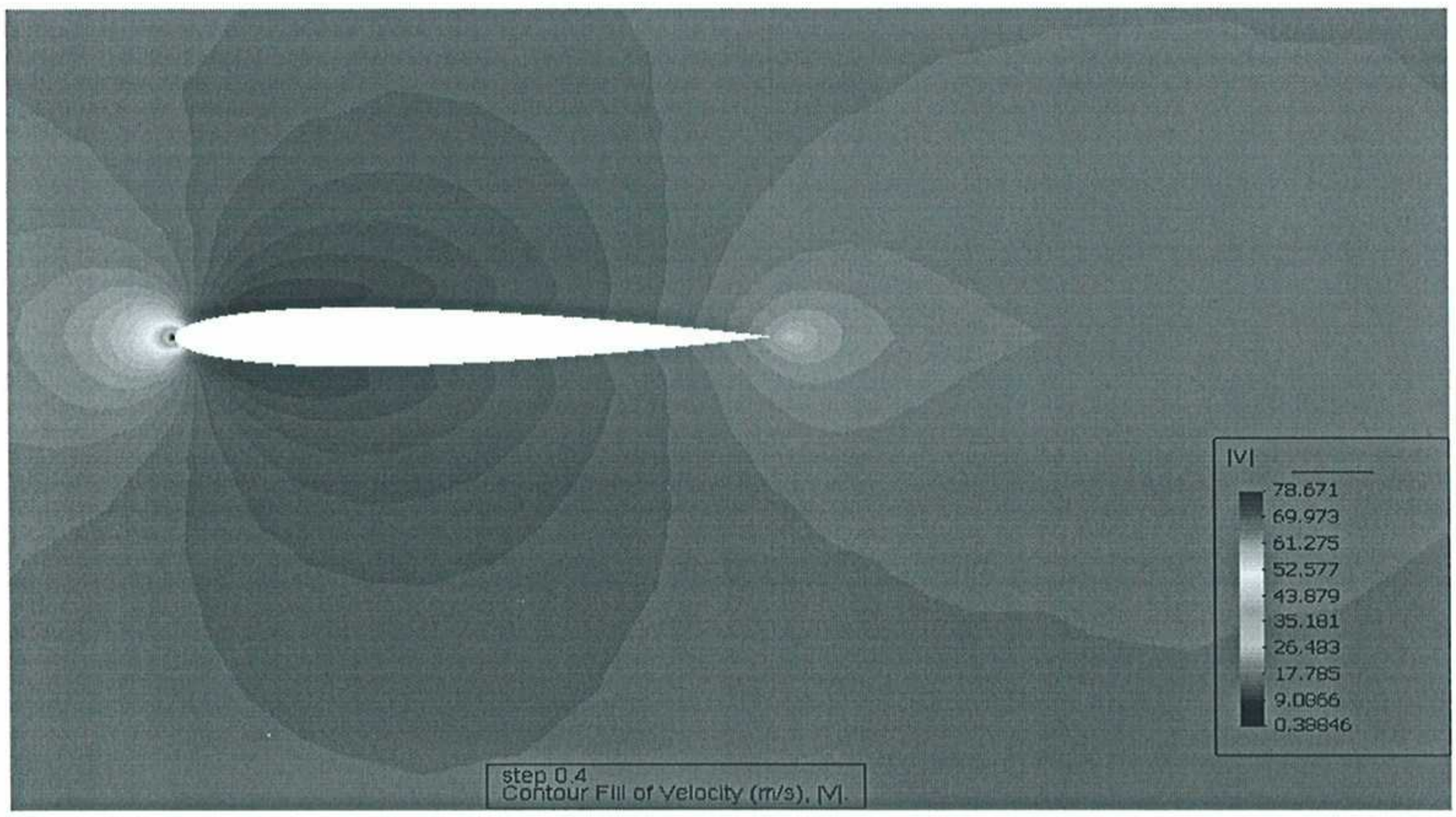
D.8 k-ε Lam Bremhorst Model (kεLB)



Y and X direction Total Pressure Force response and Y and X direction Viscous Force Response

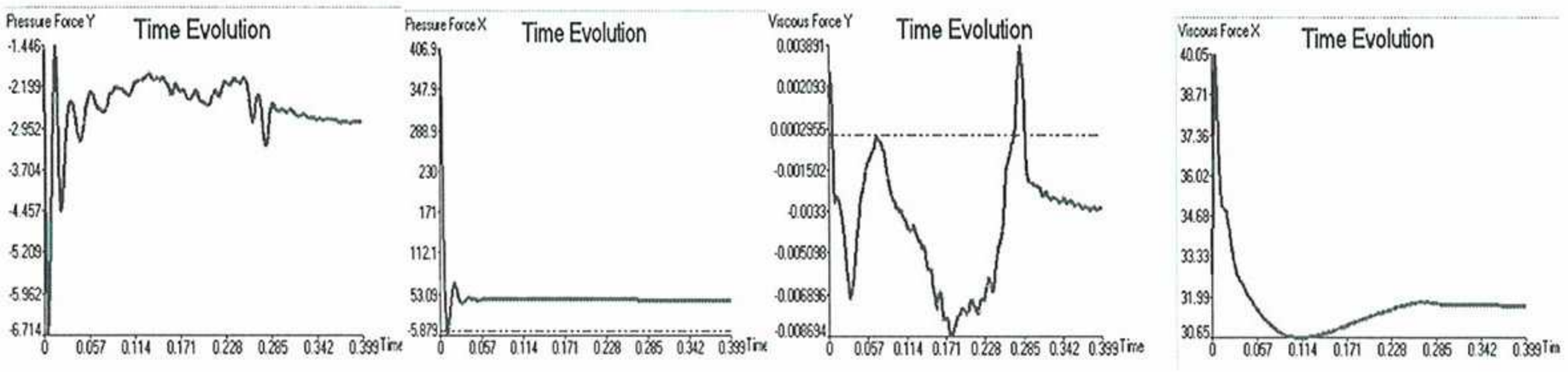


Eddy Viscosity Distribution

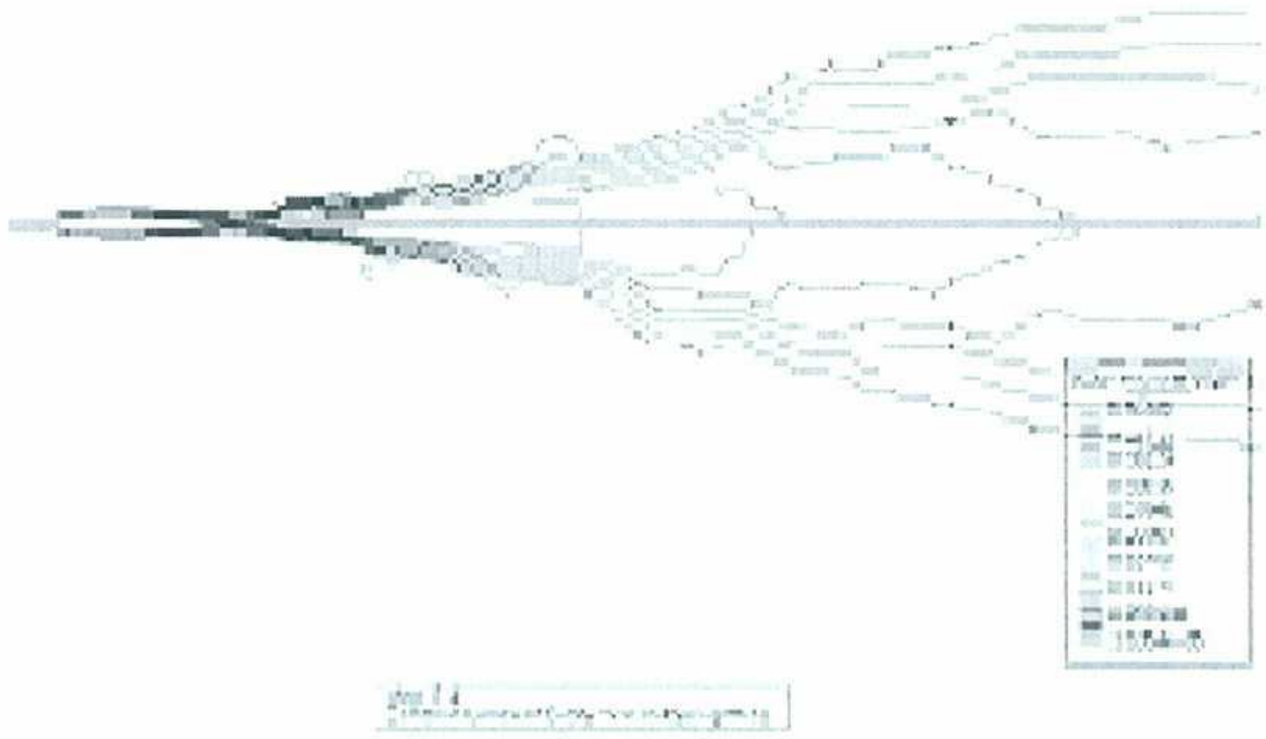


Velocity Distribution

D.9 k-ω SST Model (kOSST)



Y and X direction Total Pressure Force response and Y and X direction Viscous Force Response



Eddy Viscosity Distribution

Velocity Distribution

Appendix E: 3D Swept Results

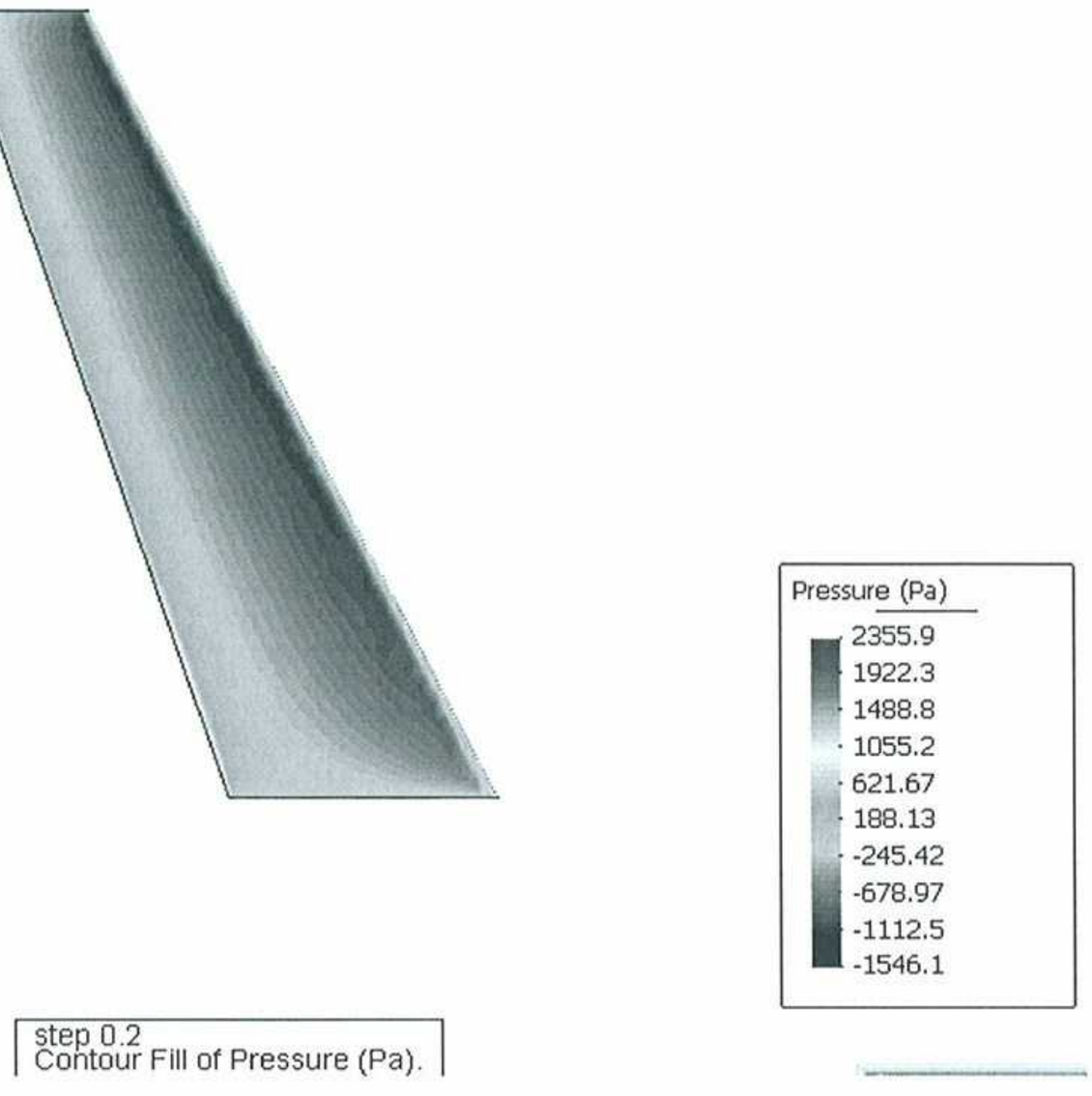


Figure E.1 Upper Surface Pressure

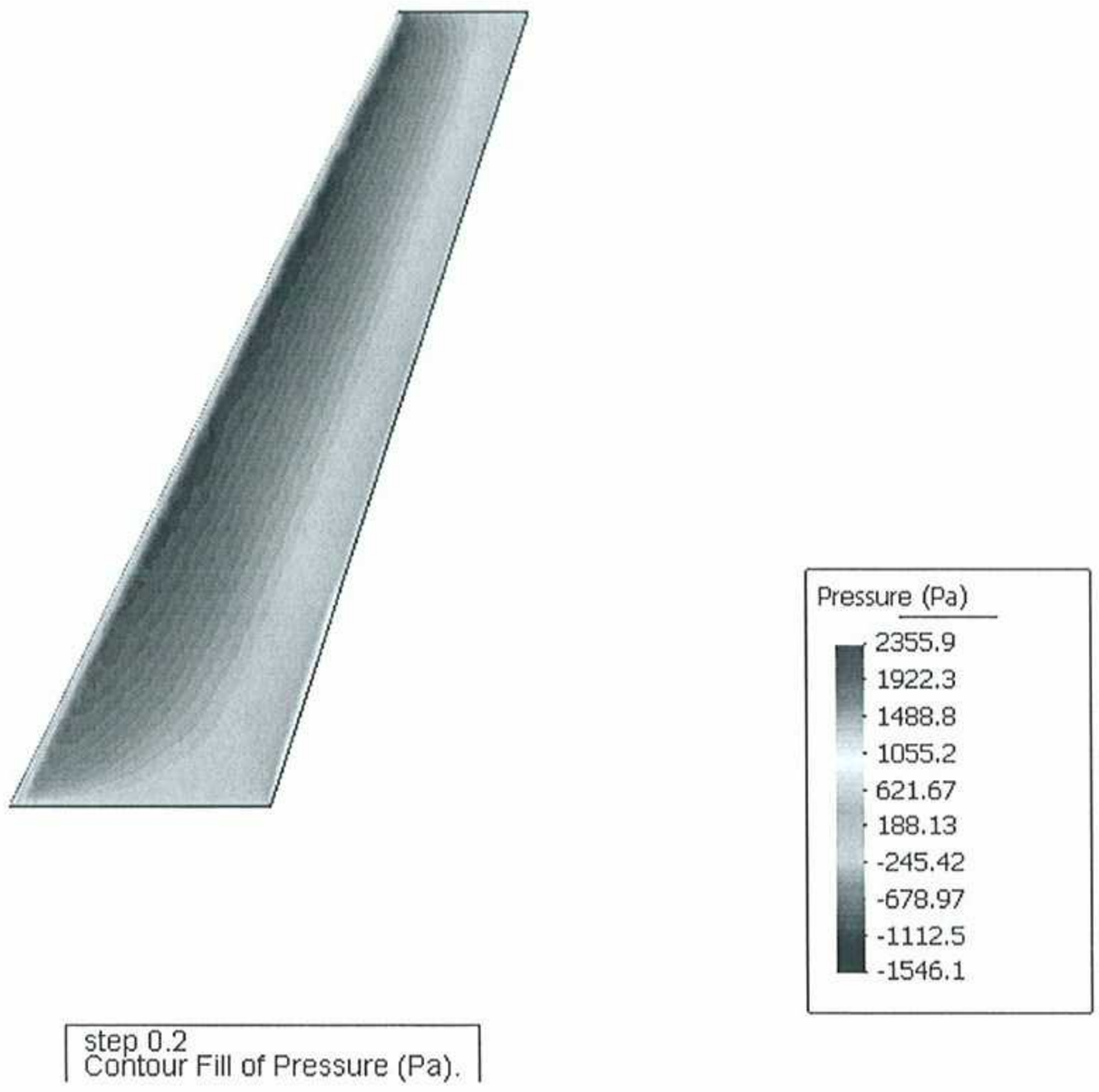


Figure E.2 Lower Surface Pressure

Figure E.3 Pressure

Figure E.4 Velocity

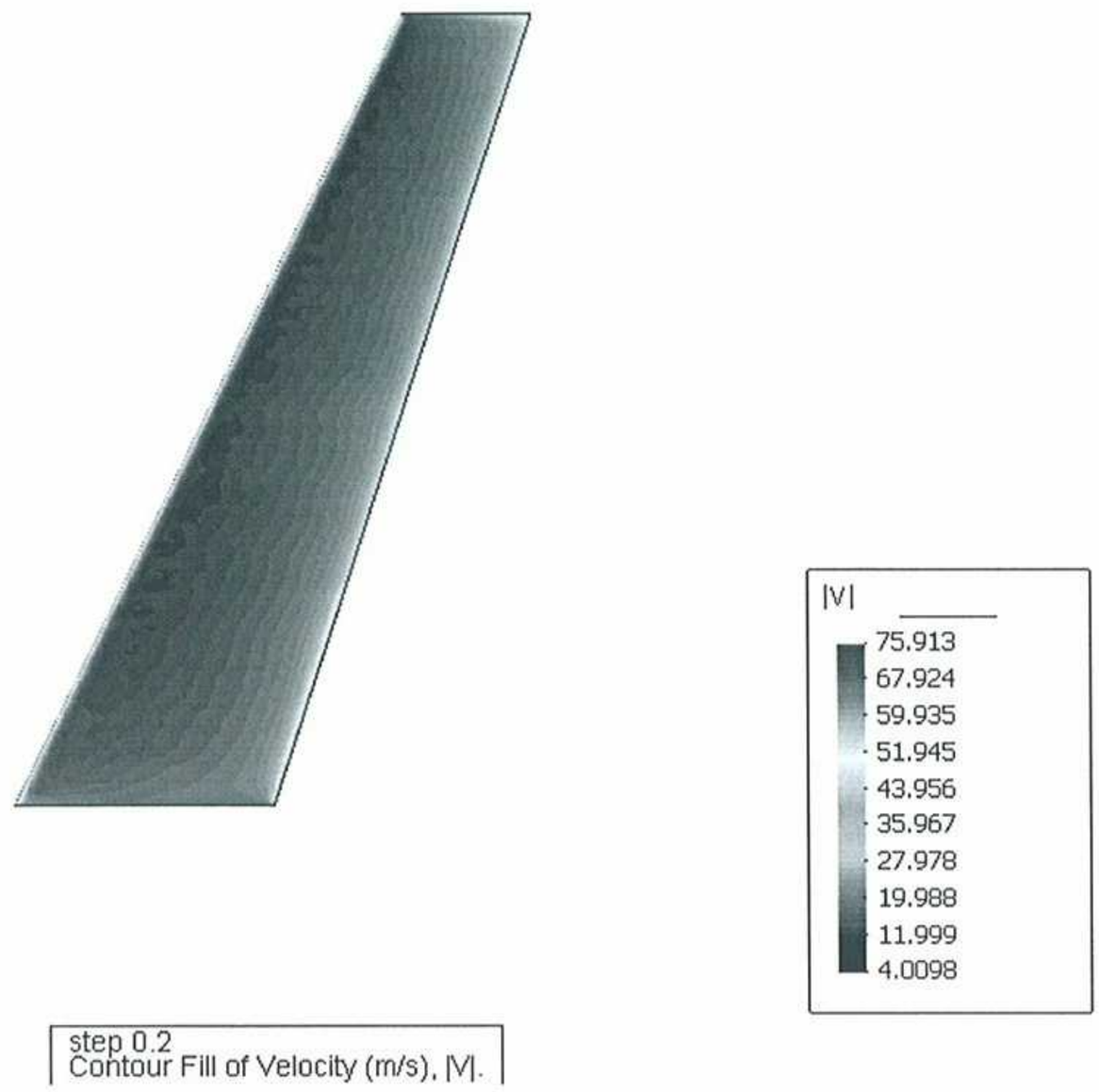


Figure E.5 Lower Surface Velocity

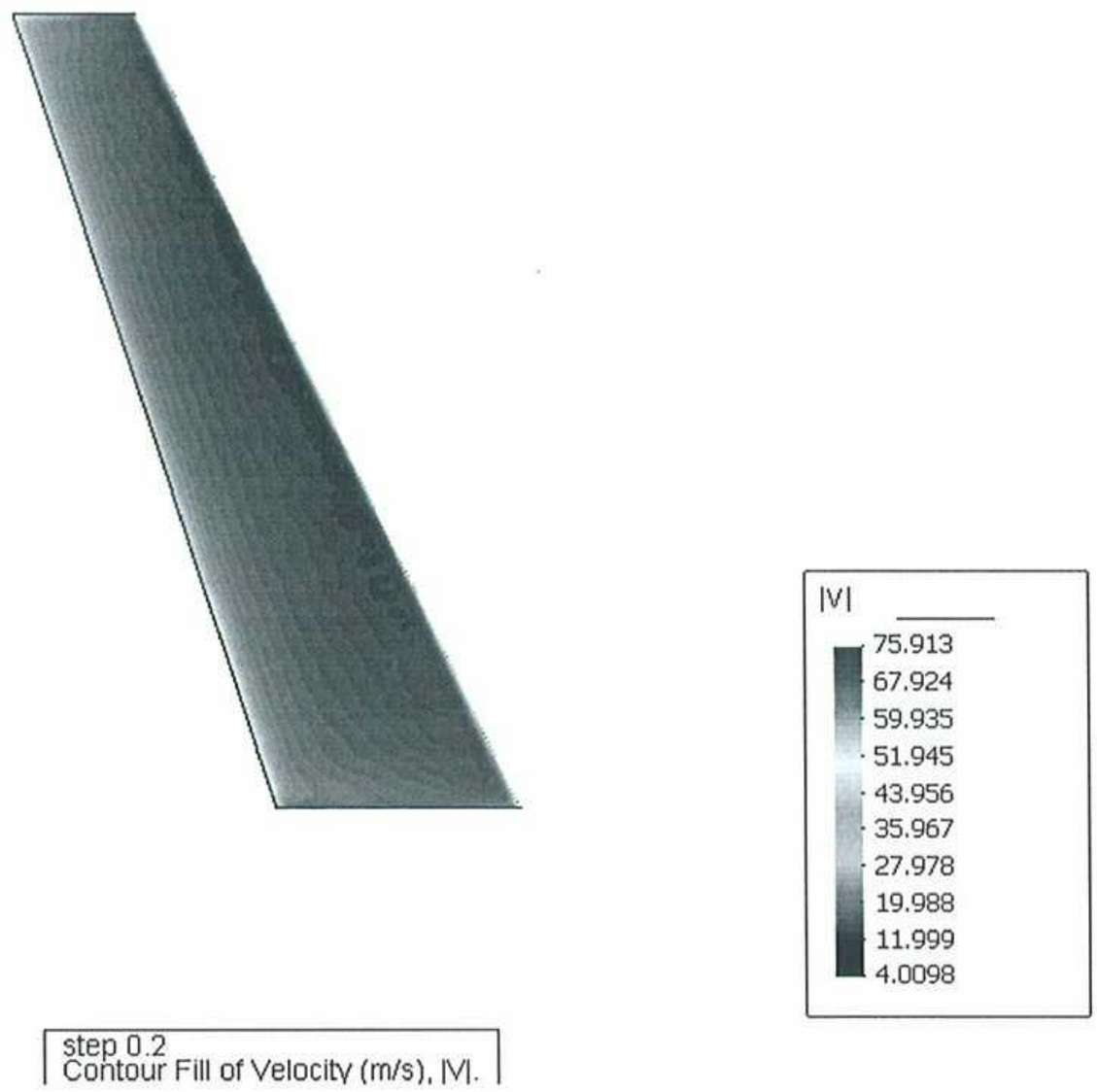


Figure E.6 Upper Surface Velocity

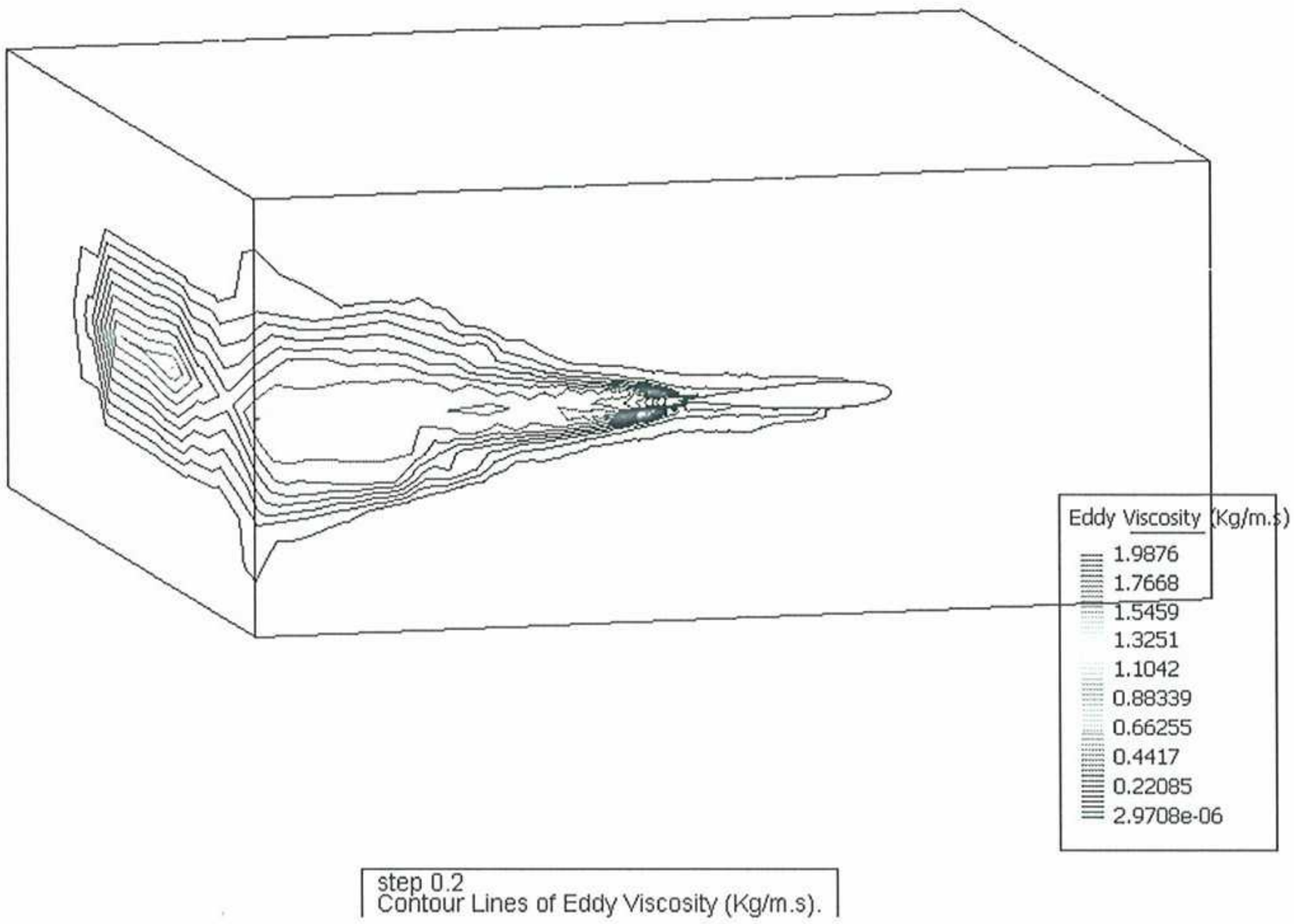


Figure E.7 Eddy Viscosity