Cranfield University

School of Mechanical Engineering

Waleed A. Wahba

Design Optimisation of Centrifugal Pump Impellers using Parallel Genetic Algorithm

Ph.D. Thesis

Cranfield UNIVERSITY

School of Mechanical Engineering

Department of Power, Propulsion and Aerospace Engineering

This thesis is submitted for the degree of

Doctor of Philosophy

Waleed A. Wahba

Design Optimisation of Centrifugal Pump Impellers using Parallel Genetic Algorithm

Supervisors: **Prof. R.L. Elder**

Dr. A. Tourlidakis

September 2001

Abstract

Computational Fluid Dynamics (CFD) techniques have settled to a stage, where it is possible to gain significant insight into fluid flow processes of turbomachinery. However, the purpose of fluid dynamics naturally goes beyond improved understanding to the aim of improving the performance of the engineering systems. Consequently, the present thesis investigates the use of an automated design optimisation method using CFD. This presents a new design method for an important turbomachinery part, blade profiles of centrifugal pump impellers, based on a shape optimisation algorithm in combination with CFD. The use of genetic algorithms in shape optimisation dose not allow the design engineer to use any derivative information on the evolution of the shape, but only simple evaluation techniques. An optimisation library (GAlib), based on a genetic algorithm (GA), was used. GA controls the evolution of a population of profiles towards an optimum design. The optimisation process can handle simple objectives as well as conflicting ones. The fitness value of each population element is evaluated using a CFD flow solver (Mac_LNS) based on finite-difference discretisation of the incompressible, Navier-Stokes (N-S) equations on structured polar-coordinate meshes. A number of design examples have been developed and the behaviour of the genetic algorithm has been tested using different kinds of objective functions. In addition, the algorithm was tested with a multi-objective function. Bézier curves were selected to represent the impeller profile. A symmetric profile, identical profile for the pressure side (PS) and suction side (SS), was used as a basic shape to generate the population elements. GAlib was modified to run as a parallel algorithm using Message Passing Interface (MPI). It is indicated that parallelisation using MPI is good technique to overcome the time taken by GA and CFD, and quite good optimisation convergence criteria was obtained by using parallelisation. The obtained results show that the genetic algorithm is capable of achieving satisfactory designs of centrifugal impeller blade profiles effectively and with a minimum amount of user expertise.

Acknowledgement

I would like to express my deepest gratitude and thanks to Prof. Robin Elder for providing the opportunities to discuss the different problems in spite of his limited free time.

My sincere gratitude and thanks are also due to Dr. Antonios Tourlidakis, for initiating the topic of this thesis, contributions and valuable supervision at any time. Also, thanks for his continuous sincere guidance to put the thesis in its final form, and the family media he established during his supervision.

I would like to express my deep gratitude and many thanks to my home, Egypt, for giving me the chance to come and study in UK.

I would like to express my deep gratitude and many thanks to the members and doctoral students of the turbomachinery department, Cranfield University, and all the members of Cranfield University for their faithful assistance.

I would like to express my deepest gratitude and many thanks to Massachusetts Institute of Technology (MIT) to use their software as part of my thesis "GAlib genetic algorithm package", written by Matthew Wall, MIT.

Special acknowledgement is due to my parents and my wife Dr. H.M. Diab, EAEA, Cairo, Egypt. I am deeply grateful to her for leaving her work and come to look after me during my Ph.D. study. Also, my great thanks to my lovely daughters Aya and Alaa for the great pleasure they gave me during staying with me in Cranfield, UK.

Waleed A. Wahba

September 2001

Cranfield University

Dedicated to My Beloved Motherland

Arab Republic of Egypt

List of Symbols

Latin

a	Speed of sound.
B	Impeller width.
C	Absolute flow velocity.
D	Diameter.
$d\phi$	Difference angle.
e	Unit vector in polar-coordinate system.
E, F	Flux vector.
f_{max}	Objective function value of the worst feasible solution in the population, in case of the main objective is minimum.
$f(\vec{x})$	Objective function.
$F(\vec{x})$	Sum of objective function $f(\vec{x})$ and a penalty term.
F_t	Time factor.
g	Gravity acceleration.
$g_j(\vec{x})$	The j^{th} inequality constraints.
$h_k(\vec{x})$	The k^{th} equality constraints.
h	Head losses.
H	Pump head.
i	Index; Node index (step level) in radial direction.
j	Index; Node index (step level) in tangential direction.
J	Number of inequality constraints.
k	Straight line equation constants.
K	Number of equality constraints.
l	Blade length.
n	Impeller rotational speed [rpm]; Number of iterations; Number of designed variables.

$NCON$	Total number of inequality constraints.
$NCON_f$	Number of flow-type constraints.
Ni	Total number of grid lines in radial direction.
$NOBJ$	Number of objective functions.
P	Bézier curve control point.
p	Static pressure.
Q	Flow rate; Conservation variable.
R	Impeller radius; Penalty parameter.
R_e	Reynolds number.
r	Polar-coordinate in radial direction.
S	Source term; Arc length.
t	Time.
u	Relative flow velocity vector presented in N-S equations.
U	Peripheral velocity.
W	Relative velocity.
$\vec{x}$	Designed variables vector.
x^l	Lower bound constraint for the design variables $\vec{x}$.
x^u	Upper bound constraint for the design variables $\vec{x}$.
Z	Number of impeller blades.

Greek

α	Arc length.
β	Blade angle measured from the tangential direction.
γ	Arc length.
δ	Artificial factor; Small positive value is converting equality constraints to inequality.
ε	Blade coefficient.
ε_e	Explicit smoothing coefficient.
η	Pump efficiency.
θ	Polar-coordinate in tangential direction.

μ	Dynamic viscosity.
ν	Kinematic viscosity.
ρ	Fluid density.
ε_j	Coefficient of the function increment, expressed in percent.
θ	Circumferential angle.
ϕ	Blade polar angle used to define Bézier curve control point.
$\phi_{overlap}$	Overlap polar angle between two blades.
ω	Impeller angular velocity.
λ	Friction coefficient.

Subscripts

e	Euler.
HEQ	Hydraulic.
i,j	Geometrical location of the calculating point in r, θ directions, respectively; index.
max.	Maximum value.
o	Stagnation.
r, θ	Components of a vector quantity in r, θ directions respectively.
th	Theoretical.
∞	Infinite number of blades.

Symbols

Σ	Summation.
Δ	Finite increment; Element length or angle; surface roughness coefficient.
$\overline{\Delta}$	Relative surface roughness coefficient.
∇	Vector differential operator.
∇^2	Laplacian operator.
$\otimes$	Cross product.
$\bullet$	Dot product.

Abbreviations

2D	Two-Dimensional.
3D	Three-Dimensional.
B	Backward.
CFD	Computational Fluid Dynanics.
CFL	Courant Friedrichs-Lewy stability condition.
DS	Downstream.
F	Forward.
GA	Genetic Algorithm.
GAs	Genetic Algorithms.
mgGA	Micro-Grain Genetic Algorithm.
fgGA	Fine-Grain Genetic Algorithm.
cgGA	Coarse-Grain Genetic Algorithm.
MO	Multi-Objective.
MOO	Multi-Objective Optimisation.
VEGA	Vector Evaluated Genetic Algorithm.
SGA	Simple Genetic Algorithm.
NSGA	Non-dominated Sorting Genetic Algorithm.
NLP	Non-Linear Programming.
ANN	Artificial Neural Networks.
SO	Single-Objective.
GP	Grid Points.
LE	Leading Edge.
N-S	Navier-Stokes.
PS	Pressure Side.
RMS	Root Mean Square.
SS	Suction Side.
TE	Trailing Edge.
US	Upstream.

Abbreviations used in the list of references

AIAA	American Institute of Aeronautics and Astronautics.
ASME	The American Society of Mechanical Engineers.
NASA	National Aeronautics and Space Administration.
MIT	Massachusetts Institute of Technology.
MTC	Military Technical College, Cairo, Egypt.
NEL	National Engineering Laboratory.
I Mech E	The Institution of Mechanical Engineers, London, UK.
NREC	Norther Research Engineering Corporation, USA.

List of Figures

List of Tables

Table Of Contents

Chapter 1

Introduction

1.1 DESCRIPTION OF THE RESEARCH WORK AND ITS IMPORTANCE

1.1.1 Centrifugal Machines

The purpose of a centrifugal machine (pump, or compressor) is to add energy to or produce pressure rise in the fluid, which flows through it. The work is done by the impeller and is evaluated in terms of the angular momentum imparted to the fluid. The casing (volute, and/or vaned or vaneless diffuser) collects the flow from the impeller and while further transforming part of the kinetic energy into pressure guides the flow to a suitable discharge opening.

Centrifugal machines produce a large head rise since the work input and the consequent head rise is proportional to the square of the impeller exit wheel speed. Therefore, the work input is proportional to r_2^2. The axial pump, lacking this attribute, achieves less head rise, but can have a large inlet area and hence can achieve very high flow rates. Even, with the recent availability of higher speeds, associated with the advancements in material and manufacturing technologies, small centrifugal impellers with higher-pressure rise are being considered to replace the multistage axial compressors in gas turbine industry and other pump applications.

The overall performance of a centrifugal machine is a function of the fluid flow behaviour inside the rotating impeller, the stationary volute, and the interaction between the two. However, due to the complexity of the relative flow structure, more attention must be given to the description of the relative flow in different impeller geometries and under different operating conditions. For the designer of a centrifugal impeller, the ultimate goal is to obtain high relative flow diffusion with uniform velocity profile at the exit where the impeller discharges the flow into the volute.

1.1.2 Centrifugal Pump Design

In today's competitive world, pump designers are tempted to just quickly copy existing pumps, or fall back on empirical design formulas established by statistical surveys of existing pumps. Such guidelines can be programmed into computer programs and can produce pump designs with the push of a button. Great problems arise, however, when the designs do not perform as expected. Without the deeper understanding of the flow pattern in pumps, the reasons for malfunctioning are incomprehensible. Statistically average designs may be adequate but will not provide a competitive edge and will leave the manufacturer in the same attitude. Today's pump designers need this push button (computer code) that has been built on a deep understanding of the flow pattern in pumps. Specially, present computers have higher speed and memory than those where empirical formulas were implemented. Computational Fluid Dynamics (CFD) codes have been improved to simulate the complete pump and reduce prototype testing and lead times for a new product. Even at the high speed of present computers and power of CFD, the major portion of the time is not consumed by the running of the program but by the time needed to judge the design. This time is to collect and input the data, reduction, plotting, and interpretation of the results. For example, figure 1.1 shows functional representation of a pump design software, named COMIG, Jansen (1995), the central core is the database (containing geometry of blade surface, etc.) that is generated during preliminary calculations. The engineer can direct the flow of calculations to the modules that surround the database. Even, it is easy for all modules to access the database but still it is the human decision (not automated), where, there is no module to judge each design case and give automated corrections.

In some design, it is attempted to automate the design by using inverse design and direct design methods. Although both methods provide the means of finding an efficient performance shape without resort to the expensive and extremely knowledge-based or try-and-error techniques, they have their own advantages and drawbacks.

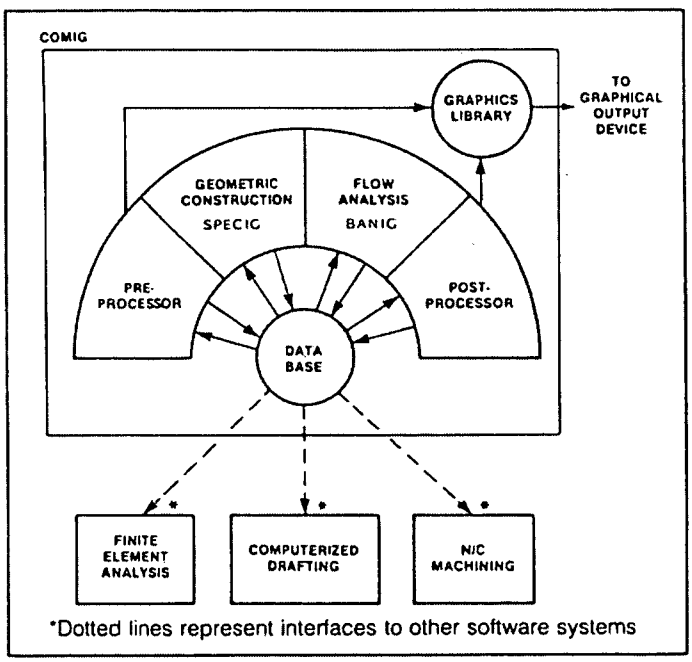


Figure 1.1, Functional representation of COMIG pump design system, Jansen (1995).

The inverse design method starts from guessed body shape and the flow solver computes the surface pressure distribution over this guessed body. Then the resulting pressure distribution is compared with the given target pressure and the difference between each other is calculated. The inverse design module provides the necessary correction to the body shape that minimises that difference. While this method is extremely cheap because it does not require a large number of iterations, the inverse design method has inherent difficulties, it is hard to define the target pressure distribution that produces optimum or desirable performance.

The direct design (optimisation) method couples a CFD code and an optimisation algorithm. The performance quantities such as head, efficiency, relative velocities distribution and pressure distribution are computed by the flow solver and the optimiser iteratively minimises an objective function, which relates shape and performances. The

drawback of some optimisation methods is high demand of calculation for computing the gradient of the objective function. There is also the possibility of finding a local minimum that is close to the initial condition because of a limited number of design variables that cannot span the whole space of feasible solutions. Genetic Algorithms (GAs) address these drawbacks, as it will be discussed later. Although optimisation procedure using GAs allows the design to have global and efficient design, as will be seen in the current work, it is expensive in terms of optimisation procedures (iteratively) or objective function calculations using CFD, especially if the trend is to improve the performance calculations. In this case, it requires intensive use of 3D Navier-Stokes solvers during the design process. Parallel programs help to reduce calculation time by dividing the task into large pieces and to solve these simultaneously using multiple processors, this will be discussed in this work. Another way to address the excessive time requirement is by using an approximate model inside the optimisation loop instead of the original model (N-S solver). Figure 1.2 describes how this method works.

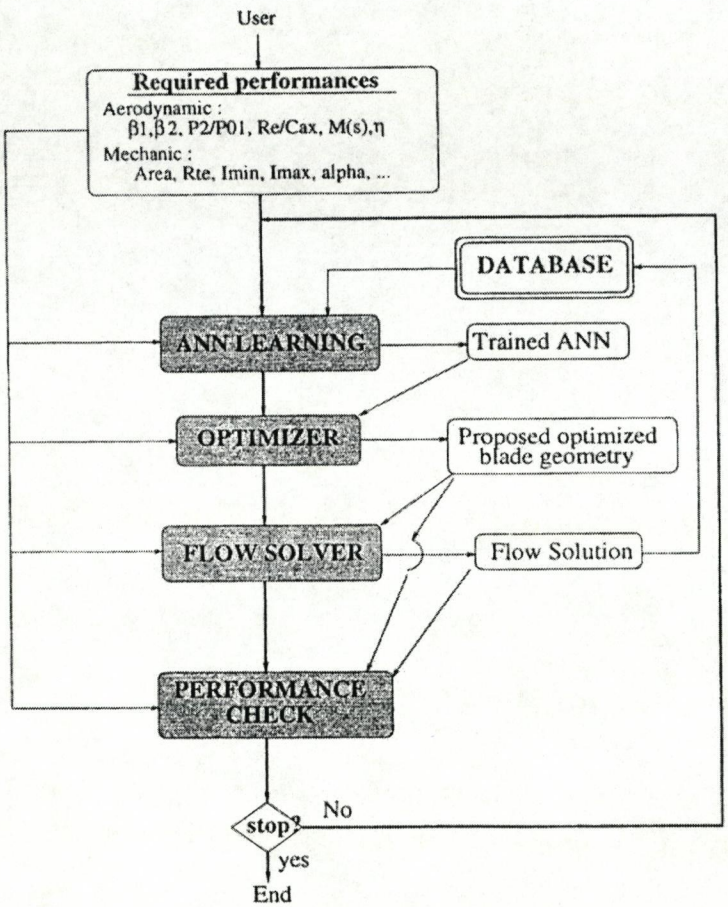


Figure 1.2, Design procedure using ANN.

The core of the design system is a database that contains the results of all N-S computations performed during the previous and present impeller design process. Artificial Neural Networks (ANN) can be used as approximate model and the identification process (learning process) is realised by back-propagation of the error. After the mapping of the database samples, the neural network is able to generalise, meaning that it can predict the performance of a new impeller geometry under given flow-field boundary conditions. An optimisation code proposes a new geometry to be analysed by the flow solver, which is added to the database. Finally the performances are checked. If the target performances have not been achieved, a new modification is initiated and the same process is repeated until convergence to an optimum geometry is reached.

1.1.3 Optimisation

Evaluation, nature's ability of continuous adaptation to ever-changing physical conditions, is a beautiful image for the concept of optimisation. In its efforts to create optimum chances of survival, humanity has benefited from distinct pioneering skills to construct devices designed for rational use of the resources available at any time. In modern society, the task of design, construction, and control of such devices is typically undertaken by engineers. Engineering design activities are concerned with the determination of designs, which meet a priori specified behavioural features. The specified design objectives are conventionally met through an iterative process of analyses, evaluations and modifications of design. In this sequential trial-and-error procedure, the designer must count on experience, sixth sense and ingenuity for every redesign, which makes engineering design a potentially creative discipline. However, in practice designers are often forced to depend on tried concepts, to cut a path through an incomprehensible number of feasible designs. This is why engineering design historically has been characterised by a slow gradual improvement of existing types of designs.

In parallel with computing speed and memory capacity of digital computers, a persistent systematisation of design methodology has led to tools for Design Optimisation (DO). These tools aim at an automation of the conventional design process through integration

of numerical tools for analysis, optimisation, and mathematical programming, so that the best design in terms of a pre-supposed criterion can be determined. Automated design optimisation is very attractive technology, because it substitutes workable designs with optimal ones, while cutting down on design times to enable a faster response to market changes. For manufacturing industries operating on today's competitive technological markets, these advantages may prove crucial for business success. An overview of important mechanisms stimulating the trend toward the application of DO is given in figure 1.3, Baysal and Eleshaky (1991).

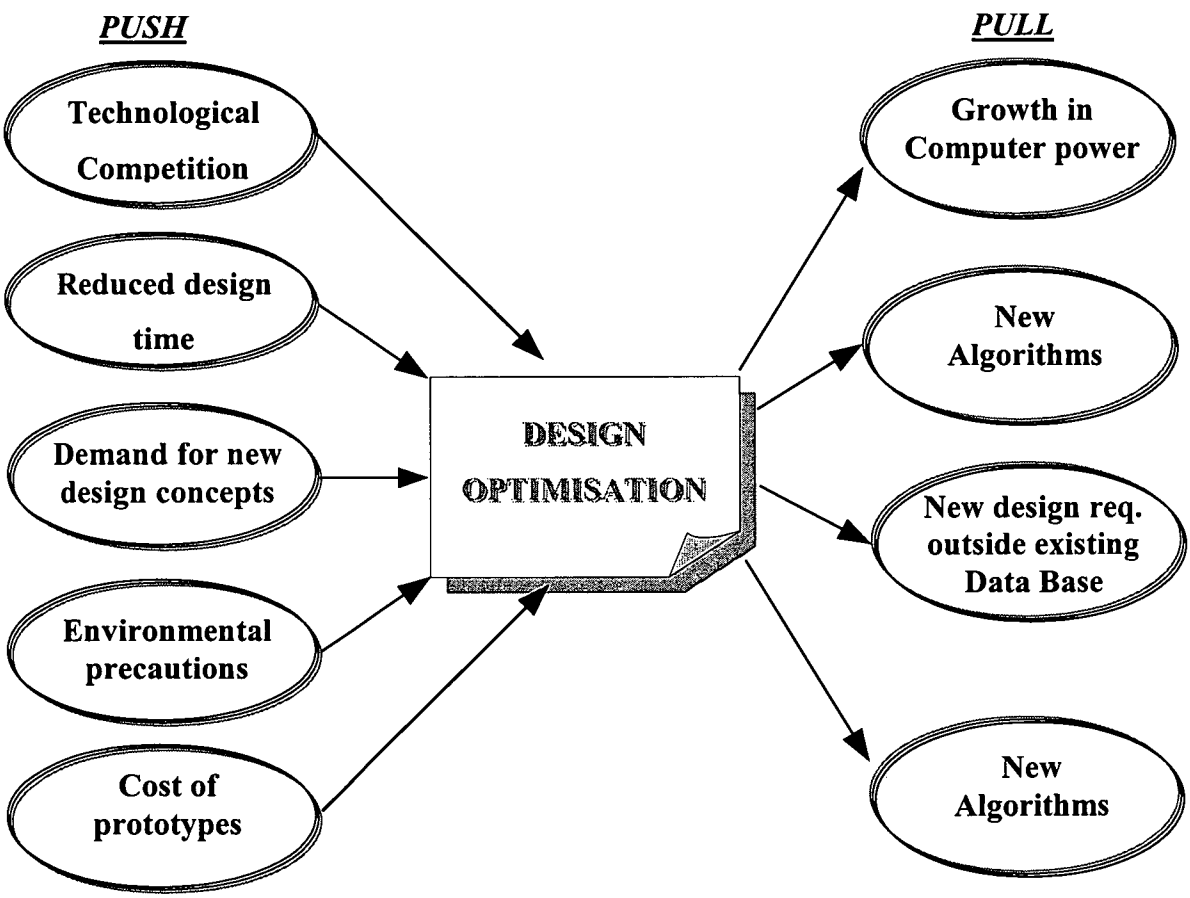


Figure 1.3, Factors stimulating the trend towards substitution of conventional design methods with tools for computer-aided design optimization.

1.2 AIM OF THE RESEARCH PROJECT

The aim of the present work is to develop and integrate methods of analysis, design, and optimisation of viscous flows governed by the Navier-Stokes (N-S) equations and to

apply this work to turbomachinery especially for the design of centrifugal impellers. Methods of design can be of human factor type, where the designer controls the design after analysis, and optimisation, type try-and-error, which is very complicated for human to reach the optimum. On the other hand, mathematical codes, for example optimisers, can do this powerfully and easily. Otherwise, viscous flows governed by N-S equations can be solved analytically by some simplifications, Euler/N-S equations, or numerically using for example Computational Fluid Dynamics (CFD).

Since the main aim is to merge two well-established fields of research, CFD and design optimisation, the interplay between these in the design process attracts special attention. Thus, the principal theoretical interest is levelled at methods of design for the coupled system of non-linear equations governing fluid flow. It will be chosen to adopt an existing N-S CFD code for use, which was previously developed by Wahba (1997). This flow solver is based on finite-deference discretisation of the incompressible, N-S equations on structured polar-coordinate meshes, as shown in Appendix A.

For an optimum fluid design there are a variety of design objectives to achieve, for example:

- **Loss minimisation**

  To minimise friction losses through an impeller passage.

  To maximise the mass flux through a pump passage with given maximum pressure rise.

- **Fashion design of parameter distributions**

  To obtain uniform velocity distributions.

- **Optimisation of fluid flow forces**

  To minimise fluid flow forces acting on structural components.

  To avoid flow cavitation in the impeller passage inlet.

These objectives can be accomplished by:

(i) studying steady viscous flow in centrifugal impellers as seen in Appendix A to understand the CFD technique which is used in this thesis;

(ii) trying to cast the impeller geometry in an automated optimisation design procedure with a CFD code, and proposing a number of criteria, integral, local and multi-criteria.

1.3 THESIS LAYOUT

The work is presented in seven chapters, with the first one as introduction. The first step in this project was reviewing optimisation methods, which are used in shape design, and for this reason a brief literature review is presented in the Chapter two. Also, this review contains GAs and constraints handling methods and their link with actual shape optimisation for fluid flow problems.

A decision of using GAs as an optimisation technology has been taken after a comprehensive study of the concepts of optimum engineering design. This study is presented in Chapter three, where it contains why GAs are used and how they are working. A link between a GA library (GAlib, Wall 1996) and a CFD code (Mac_LNS, Wahba 1997) has been properly carried out. Chapter four describes this link. Results and discussion are presented in chapters five and six.
Chapter five describes how GAlib works and how the parameterisation of its variables affects the design procedures. Chapter six focuses on the impeller design processes, which are based on various optimisation objectives and constraints. Conclusions and recommendations are presented in Chapter seven.

There are two appendices added to the thesis. Appendix A contains a brief overview of the Mac_LNS code used in this work. Appendix B contains two codes used in the present work, Eva_POP(GAPopulation & p) function used to parallelise the GAlib using MPI and De Casteljau's Algorithm used to develop Bézier curves.

Chapter 2

Literature Survey

2.1 INTRODUCTION

Several pump design procedures are described in the literature, including methods covered in books, engineering journals, or government reports. These procedures concern themselves primarily with design point operation, Whitfield and Baines (1990). So far, It is necessary at the outset to establish quite clearly the designers' objectives. The perfect hydraulic pump design is of no use unless it can be made and put into service, economically. Invariably there is a conflict created by the design needs of each objective, and compromise is necessary. This compromise may become a subconscious optimisation in a good experienced designer, nevertheless it is still desirable to have a quantitatively defined set of priority ranked targets in the design specification. Where, as is increasingly common, there are separate applications engineers, hydraulic designers, mechanical engineers and production engineers, it is even more important that the hydraulic designer recognises the effect of his/her decisions on the ability of the others to meet their commitments. Hydraulic design needs a list of requirements that may be one of conflict and compromise. For example, the impeller, which is easiest to cast, is unlikely to produce the best hydraulic performance. The hydraulic designer therefore needs a system, which allows rapid optimisation. From this point of view, this chapter will provide a literature review for these methods used for pump design. Also, it

will handle the interaction between CFD and design and the interaction between shape design and optimisation using inverse design and direct design. Finally, a comprehensive survey of GAs, as a powerful tool for optimisation design, will be presented containing GAs methods, constraint handling approaches and GAs parameterisation.

2.2 PUMP DESIGN

In this section, various aspects of classical theory are described in order to demonstrate the foundations of some of the current industrial hydraulic design and comparing methods. This is followed by design procedures used for pump design.

2.2.1 Pump Design Methods

Many methods have been provided for pump design, for example modelling, modified modelling, inlet design, free vortex, area ratio, design coefficients and through flow.

Modelling design is derived from consideration of geometric and fluid dynamic similarity. Its idea depends on dimensional analysis using Euler's equations to deduce the same relationships. This technique depends on geometric scaling which is not always feasible since mechanical sizes do not generally vary in the same proportion as hydraulic size. It is therefore necessary to make corrections, either to size or to performance predictions, particularly if the size ratio between the model and prototype is large. There is considerable discussion on the question of pump performance scaling. Nixon (1966) is a particularly comprehensive reference discussing all the related factors. Progress in this area has been slow, but other sources of information are: Nixon and Cairney (1972), Fay (1976) and Osterwalder and Ettig (1977). The modelling design method has been modified in some references like Karassik et al (1986), Stepanoff (1976), Varley (1961), Worster (1963), and Schweiger (1969). The modification was derived from the theory of more fundamental (larger eye diameter, impeller blade outlet angle, blade number, impeller outlet width, or collector throat area).

In case of inlet design, it is used for those pumps being designed from scratch, where it is first necessary to decide on the optimum inlet configuration. For more details see: Pearsall (1973 & 1977), and Anderson (1955).

Worster (1963) published an analysis of pump performance using free vortex method. This analysis showed the explicit relationship between the impeller and collector geometry in establishing pump performance. This method depends on defining the velocity distribution around the impeller. It also, produced a relationship between flow rate, total head, impeller design parameters and collector throat area.

The basis of the area ratio method, produced by Anderson (1938), is a collection of normalised design data, in the form of head and flow coefficients plotted against area ratio. Refer to Thorne (1979), it has been shown that there is a relationship between area ratio and specific speed for optimum efficiency, stable total head/flow rate characteristic and shape of impeller blade. In the area ratio method, for any specific speed there is an infinite range of designs possible.

Stepanoff (1976) has published the first and complete hydraulic design of a pump by using design coefficients. It is based on velocity and geometry analysis charts plotted against specific speed, where the principal variables are head and flow coefficients based on dimensionless terms. The charts assume a consistency of design so that, only pumps of similar type, construction, number of impeller blades, etc., should be plotted on one chart.

The through flow analysis techniques available compute the behaviour of a large number of small elements of the flow in order to establish the effective energy transfer and velocity vectors at any point. Total flow rate and head are established by integrating across the passage blade-to-blade and wall-to-wall, and summating for the number of passages. These methods have extensive research in handling real flow in real machines analytically and numerically. But the instrumentation techniques necessary to measure internal flow in order to verify the predictions are extremely sophisticated. For modern computing power, these methods are suited to interactive design and so potentially

improve the design process by giving the designer the facility to look at many variables simultaneously and to select appropriate values quickly. This is leading to enhanced optimisation.

2.2.2 Pump Design Procedures

A calculation procedure to estimate the theoretical performance of a pump is an indispensable tool in pump design. Performance needs to be known, not only at the rated, best efficiency point, but also off design. Pump specifications often impose special requirements, such as head at shutoff, maximum power demand, rate of head rise to assure stability and so on. A good pump design procedure requires trial-and-error iteration, a check on predictable performance with a trial geometry, and progressive approximation to the optimal design configuration. The best hydraulic design does not necessarily correspond to the best commercial pump product. Compromises are unavoidable.

Excellent pump performance calculation procedures have been developed and published (Stepanoff 1948 and 1976, Turton 1994), and several computer programs are commercially available. A typical computer program developed by NASA from the COSMIC collection software is called PUMPA (http://www.openchannelfoundation.org/projects/PUMPA/). PUMPA is based on the Euler equation coupled with empirical correlations compiled from rocket pump data and does not identify the various loss mechanisms. Generally, the more accurate and detailed these calculations are, the greater the number of input variables needed; not only the desired head, flow rate, and rotational speed, but also the details of the geometrical description of the impeller and housing. The labour and cost required to prepare and enter the input variables limit the practical use of computer programs for simple, inexpensive pumps. Traditionally centrifugal impellers have been designed using semi-empirical techniques such as, the area ratio method. These methods are effective, but largely treat the impeller as a "black box". The uncertainty associated with these methods often requires experimental verification and optimisation of the performance on first prototype impellers. This experimental work is both costly and time consuming leading to increases in both product development life cycle costs and

time to market. In a recent study it was shown that a delay of six months in bring a product to market could reduce profits over the product's life by a third*.

2.3 CFD AND DESIGN

Over the last twenty years, much work and progress has been made using CFD. This work has led to significantly new physical insights into the behaviour of flows ranging from laminar to turbulent, from non-reacting to reacting, from Newtonian to non-Newtonian, etc, there is much technology now available to reliably compute flows in complex geometry with complex physics. Computation now stands as an equal partner with mathematical analysis and experimental inquiry. CFD has become such an effective tool that many researchers who previously would depend only on experiment to uncover fluid phenomena now use CFD to achieve their goals more rapidly and cost effectively, Orszag and Staroselsky (2000). Moreover, up to now, several factors have limited the impact of CFD in the industrial design environment. Among these factors, two seem especially limiting. Firstly, traditional CFD methods have mostly focused on the analysis of existing configurations with the purpose of identifying possible problems and shortcomings of the existing design: little or no guidance is offered to the designer on how to improve the existing design or how to avoid the configuration shortcomings. Secondly, the complexity of the required calculations has lengthened the turn-around time to a point where CFD cannot play an effective role in the continuously shrinking design cycles. For numerical techniques to be employed successfully in a design environment, these two key issues need to be dealt with effectively. So, current efforts attempt to use CFD in design by linking it with CAD (Al-Zubaidy, 1995) or optimisation algorithms (Narducci et al., 1995; Johansen et al., 1997 and Madsen et al., 1997).

In recent years, there has been significant effort to infuse design sensitivity analysis into CFD analysis; gradient-based non-linear programming algorithms then utilise the sensitivities to perform the optimisation efficiently; Newman et al (1999), Taylor et al (1992) and Appa et al (1998). These developments have received considerable attention

---

* Solid Modeling – Today's Engineering Reality, Published by Hewlett Packard.

in the aerospace, automotive, and biomedical industry due to the massive potential of design sensitivity analysis as a design tool. Newman et al (1999) and Taylor et al (1992) applied the sequential implementations of design optimisation method for the design of airfoils, wings, wing-bodies and complex aircraft configurations using both the potential equation and the Euler equations. A two-step design optimisation process employing features of structural, aerodynamic, propulsion and flight control systems is developed by Appa et al (1998). Similar rapid evolution of design optimisation based on CFD analysis is also reported in the automotive, biomedical and electronics industry literature. A design optimisation methodology for compact heat exchangers used in the automotive industry, like the radiators and condensers, for 2D fin louver is presented by Bouzida (1997). Another detailed study of CFD design optimisation of finned heat sinks for impingement cooling of electronic packages was carried out by Kondo et al. (1998). In Siegel and Makhijani (1998), numerical optimisation has been documented for practical biomedical design constraints. A multi-disciplinary shape optimisation using generic algorithms has also been reported for aerodynamic and Electro-magnetic applications by Makinen et al. (1999).

From these documented contributions in the literature, it is evident that CFD can and should be used for the purpose of design in order to cut down the cycle time for a new or improved product. However, for a practical problem involving large matrix of candidate designs or design variables, the numerical optimisation procedure may require prohibitively expensive computational resources, but parallel calculation can be applied for both CFD and optimisation to solve this problem.

2.4 SHAPE DESIGN

The bulk of work relating to design optimisation in fluid mechanics has been related to aerospace applications, considering, for example, the optimum shape of airfoils or parts of aircraft combination, as mentioned in the last section. For the most part, up to the late eighties, these studies relied on irrotational approximations of the fluid flow, which is justifiable for the analysis of high-speed external flows around aircrafts. However, the current use of Euler or even N-S solvers for these problems has increased the computational costs to an extent, where efficient optimisation methodologies have

become essential to the prospects of CFD-optimisation. Moreover, since the present scope is that of internal flows, there is a predictable call for efficient techniques to limit the computational expenses involved in the optimisation of N-S flows (Taylor et al., 1991).

Methodologies for solving shape design problems can be distinguished into two classes: inverse methodology and optimisation (direct) methodology. The distinction is based on how the design problem is formulated.

2.4.1 Inverse Methodology

Historically, the predecessor of numerical design optimisation in fluid mechanics is the inverse design methods. An inverse design problem is posed by prescribing a target pressure or velocity distribution around an object, and is solved by determining the shape of this object that achieves the specified distribution. Lighthill, 1945, devised an inverse design method for the incompressible flow past airfoils, making use of conformal mapping and potential flow solutions. State-of the art inverse methods are based on the Euler equations and wall modifications by means of a so-called transpiration model, Giles and Drela (1987) and Demeulenaere (1997). Inverse methods are primarily dedicated to the design of airfoils, wings, and turbomachinery cascades, but have also been applied for the design of duct geometry, Cabuk and Modi (1991), and Cabuk and Modi (1992). Dulikravich (1991) published surveys of inverse design methods.

In the field of turbomachinery blade design, von Karman Institute (VKI) has a valuable thrust in inverse design. VKI used the inverse design with many flow solvers like, potential flow, Euler, and Navier-Stokes, (Van den Braembussche, 1998; Bogers, 1998; Demeulenaere, 1998(a&b); and Nielsen and Myllerup, 1998).

The main drawback of the use of inverse design methods is that these require an a priori definition of an "optimal" target distribution by the designer. This obviously demands a close understanding of the flow phenomena considered, and furthermore involves a problem regarding the existence of solutions. For these reasons, it is a desperate undertaking to apply inverse design methods for general purposes, while the methods may prove very efficient for specific problems, which are well understood, such as for

the design of pump impellers, Zangeneh et al. (1996). Numerical optimisation techniques avoid the complications described, and furthermore encompass the solution of inverse problems, which may be formulated as problems of minimising a norm of the difference between desired and actual performance.

2.4.2 Optimisation (Direct) Methodology

In the optimisation (direct) methodology, the design problem is based as a minimisation (min) or maximisation (max) problem of an objective function subjected to constraints on the geometric and fluid properties. Optimisation methods assist the designer in locating the min/max of the objective while satisfying the constraints. From the practical point of view, optimisation methods, pioneered by Hicks et al. (1974), are more attractive since these methods can handle a large class of design problems, including those classified as inverse problems. Direct methods can be distinguished into two categories: global methods and local methods. Global methods, such as those based on the GAs, Doorly (1997), are aimed at obtaining the global optimum, these methods are most useful for cases in which multiple minima/maxima are present in the design space, where in real life, the majority of design problems have multiple objectives (minima/maxima). GAs are used in the present work and more detailed description about that will follow later.

Local methods use the information on the gradient of the objective for locating the optimum. Therefore, for cases with multiple minima/maxima, local methods are limited to produce only one of the objective (i.e., the local optimum), the actual value of which depends on the starting point of the optimisation process. Recent developments in the gradient-based optimisation methodology suggest that two main streams may be distinguished: the method of sensitivity analysis and the variational method. This distinction is based on how the gradient is computed. The sensitivity analysis method has the advantage that the sensitivities of the flow properties on the grid points can be determined, where, the gradient of an objective function can be computed easily using the chain rule. For more details, see Pandya (1997), Haftka (1986a), Elbanna and Carlson (1990), Baysal and Eleshaky (1991), Taylor et al. (1991). The variational method needs the values of the so-called adjoint variables as the solution of a set of adjoint equations. For more details, see Alonso (1997), Jameson (1995), and Reuther

(1999). Gradient-based optimisers have successfully been applied to a variety of design problems (Kroo, 1988 and 1990 and Gage, 1992). Unfortunately, these methods cannot be applied to problems with discrete variables or discontinuous objective and constraint functions, because gradients are not defined in these domains. Furthermore, gradient-based optimisers must start with a specific parameterisation of the design and are limited to finding optimal values of the chosen design variables. The most obvious limitation of gradient-based optimisers is their need for gradient information. So, finite-difference estimates must be calculated in most cases, which account for the bulk of the computational effort. Automatic differentiation may offer a fast alternative for gradient calculation (Stamatiadis, 2000 and Hong Hu, 1999), but even this method is not effective for domains in which gradients are undefined.

2.5 EVOLUTIONARY ALGORITHMS

Mankind tries to learn from the nature's principles. Whereas, in previous times people only copied structures or shapes found in nature, e.g., wings of a bird, in the mid sixties, strategies were developed, emulating the principles of organic evolution in order to model adaptation or optimisation strategies. Today, these algorithms are termed as *Evolutionary Algorithms* (EAs). In general EAs simulate evolution using four main elements: 1) encoding structure that will be simulated; 2) operators that affect the individuals of a population; 3) a fitness function that indicates how "good" a certain individual is with respect to the others; and 4) a selection mechanism. EAs are divided into three main streams, whose motivations and origins were completely independent from each other.

First, *Evolution Strategies* (ES), invented by Schwefel (1995) around 1965 at the Technical University of Berlin, were first used for experimental optimisation. With the appearance of the first computers, this strategy developed towards a numerical optimisation strategy based on real-coded vector representations.

The second is *Evolutionary Programming* (EP), which was invented by Fogel et al. (1966), who described an EP for the evolution of finite state machines to solve

prediction tasks. In the beginning of last decade, his son Fogel (1995) extended his father's work to real-valued object variables.

Third, Holland (1975/1992) is a pioneer in the area of *genetic algorithms* whose research interests was devoted to the study of general adaptive processes. Hollstien (1971) and DeJong (1975) applied these GAs based on binary variables to parameter optimisation. Finally, Goldberg (1989) presented a monograph building a reference book in the field of GAs. The current work will use GAs, and for this reason the next subsections will discuss work which has been done in this area and constraints handling strategies.

2.5.1 Genetic Algorithms

In the last few years research devoted to GAs and their application has significantly increased as indicated by the existence of several conferences on the topic. In particular the use of GAs has gained some popularity in optimisation, Goldberg (1989), and has been identified as a potential technique to be evaluated in heuristic search and combination problems. GA is a global search technique. It simultaneously evolves many points in the parameter space. By working with a population of solutions, the algorithm can search many local minima/maxima and thereby increase the likelihood of finding the global minima/maxima. The main advantage of GAs methods is that they do not require any mathematical increase to the numerical solution methods used to represent the ill objects. GAs have recently been applied to turbomachinery, for example, turbomachinery cascades (Oksuz and Akmandor, 2001), vaned diffusers for centrifugal compressors (Benini and Tourlidakis, 2001) and pumps (Oyama and Liou, 2001 and Wahba and Tourlidakis, 2001a).

There exists an excess of studies investigating the interactions between different GA parameters for different application of GAs. This is rightly so, because GA parameters (such as population size, choice of GA operators, operator probabilities, representation of decision variables, etc.) interact in a complex way. More importantly, their interactions are largely dependent on the function being optimised (Hart and Belew, 1991). Since these interactions are complex and a complete analysis of all their interactions is difficult to achieve, researchers have used different analysis of all their

pair-wise effect on GA's performance. These isolated studies are valuable and have provided useful guidelines for choosing GA parameters, such as population size, Goldberg, Deb, and Clark (1992) and control maps for operator probabilities, Goldberg, Deb, and Thierens (1993). In order to observe the interactions of various GA parameters, empirical studies have also been used, Esheelman and Schaffer (1993). To study the dynamics of these interactions, more sophisticated stochastic models have also been developed and analysed, Suzuki (1993).

2.5.2 Parallel GAs

Recent advances in computational science can be attributed to the development of new efficient algorithms and the development of fast hardware. Computer hardware has been developing in an explosive fashion over the past two decades, providing practical solutions to many problems. The concept of parallel computation has been greatly speeded up and it is important to develop efficient algorithms, which take advantage of these developments. On the other hand, it is rather easy to parallelise GAs, unlike many traditional optimization methods, because of the inherent parallelism of these algorithms (Dorigo, and Maniezzo, 1993). The individuals can be modified and, most importantly, evaluated independently of each other, e.g., using a master-slave approach (Bäck, 1996). It results in a speed-up scaling linearly with the number of processing units. The fact that standard selection acts on the whole population, thus limiting the speed-up, motivated the development of parallel GAs using local selection within "sub-populations". For two of these algorithmic variants, migration models (Rudolph, 1991) and diffusion models (Spiessens and Manderick, 1989), it has been observed that local selection techniques not only yield a considerable speed-up on parallel computers, but also improve the robustness of the algorithms (Gorges-Schleuter, 1992) and ease the search.

2.5.3 Constraint Handling

An aspect normally disregarded when using EAs for optimisation is that these algorithms are unconstrained optimisation procedures, and therefore it is necessary to device ways of incorporating the constraints (normally existing in any real-world application) into the fitness function. Different techniques are available for handling constraints, e.g., Lagrange multipliers, Adby and Dempster (1974), and mathematical

programming, Mital (1977) and Bazaroo and Shetly (1979). However, the problems that are undertaken by GAs, often do not allow the use of these methods because the constraints cannot explicitly be defined in terms of the problem parameters, i.e., only after evaluation of a trial solution it may become clear whether or not a constraint is violated. Coello (1999) has achieved a very good survey of EAs constraint handling techniques.

One technique to handle constraints is to use crossover and mutation operators such that the constraints are always satisfied. The operators designed for handling combinatorial problems with GAs, Goldberg (1989), gives the most illustrative example for this kind. The second technique is to represent the optimisation problem such that it is impossible to violate a constraint. Two other types of constraint handling techniques comprise the repair and penalty functions. Penalty functions were introduced by Courant, 1943, and constraints are added as a weight term to the objective function that must be optimised, that means, the constrained problem is transformed into an unconstrained problem. In other words, the simplest approach to constraint handling in optimisation algorithms has been to assign infeasible individuals an arbitrarily low fitness (Goldberg, 1989). This is possible given the ability of optimisation algorithms to cope with the rapid fitness changes, which arise on the constraint boundaries. In this approach, provided feasible solutions can be easily found, any infeasible individuals are selected out and the search is not affected much. Michalewicz and Schoenauer, 1996, handled more efficiently certain types of constraints, such as bounds on the decision variables and other linear constraints. They mapped the search space so as to minimise the number of infeasible solutions it contains and/or designing the mutation and recombination operators carefully in order to minimise production of infeasible offspring from feasible parents. This and the previous approach are complementary, and are often used in combination with each other.

In the case where no feasible individuals are known, and cannot easily be found, simply assigning low-fitness to infeasible individuals makes the initial stages of evolution degenerate into a random walk. To avoid this, the penalty imposed on infeasible individuals can be made to depend on extend to which they violate the constraints. Such

penalty values are typically added to the unconstrained values before fitness is computed (Goldberg, 1989). Although penalty functions do provide a way of guiding the search towards feasible solutions when these are not known, they are very much problem dependent. Some infeasible solutions can, even with the penalty, be seen as better than some feasible ones, which can make the population evolve towards a false optimum. In response to these difficulties, Richardson et.al. (1989) have described guidelines on the use of penalty functions.

Deb (2000) has proposed one of the most recent approaches to constraint handling. This method is based on penalty functions but does not require any penalty parameter and offers distinction between feasible and infeasible solutions. It uses the idea of comparing infeasible solutions depends only on terms of constraint violation. This perspective is supported and extended in the present work.

2.6 CONCLUSION

From this literature review, several pump design procedures are described. These procedures concern themselves primarily with design point operation, whereas other methods depend on copying existing pumps, or on empirical design formulas established by statistical surveys of existing pumps. Therefore, the designer needs a system, which allows rapid optimisation.

It is concluded that, CFD can and should be used for the purpose of shape design in order to cut down the cycle time for a new or improved product. It is also clear from the literature review that, there are a lot of attempts to find new methods that are intended to minimise the number of interventions of the designer and to shorten the design time. Experiments or N-S calculations are currently performed on designs produced by a try and error procedure which allows good design, but is very expensive in terms of computer and/or designer time. Its outcome strongly depends on the expertise of the designer. Direct optimisation method is one of these methods, which the designer can get his objective with minimum effort and lesser experience. GAs offer a much more robust way to handle the optimisation problem especially multi-objective optimisation.

Chapter 3

Multi-Objective Optimisation Using Genetic Algorithms

3.1 INTRODUCTION

Searching a complex design space involves a trade-off between two apparently conflicting objectives: exploiting the best solution currently available and robustly exploring the unsearched space.

The most direct way to take advantage of the current best solution is to improve upon it. This ensures that information about the best solution currently available is not lost and that maximum use is made of this information. This method examines all good points neighbouring the current best solution, but on the other hand does not examine points that are far from the current solution. Examining regions of unexplored search space, even if far from the current solution, might provide knowledge that leads to the discovery of an even better solution. Greedy search, Duxbury and Dobrin (1999), is a good example of a search strategy that exploits the present information for finding an improved solution. Since, greedy algorithms take such a limited view of where the opportunities for improvement exist, they are vulnerable to getting trapped in regions far removed from the optimal solution. If focusing on an apparently most promising

region does not lead to the optimal point, a good search strategy should move on to other regions of the space and continue its search.

Two kinds of calculus based methods exist for finding the optimal value of a function using the improvement method. One is dependent on the availability of the function's derivatives: the function is evaluated at all points of its domain where the gradient is zero and the best among them is selected. The second method seeks the optimum by searching in the direction related to the local gradient. This is simply the notion of *hill climbing*: to find the local best, search the function in the steepest permissible direction until a best value is reached. The problem with the first method is that the derivative is not always available for all search spaces, and the problem with the second method is that the procedure can get stuck in local optima and not be able to achieve the global optimum. Another interesting point to note is that the success or failure of these algorithms is very dependent on the single starting point.

Both these methods are local in scope where the optima they seek are the best in the neighbourhood of the current point, but not necessarily the best in the overall picture. These procedures depend to a great extent on where the search is started. Since the real world of search rarely ever has a continuous and unimodal function, both the above methods are of little use when solving real world problems. As researchers have recognised the problems with calculus-based methods, randomised search algorithms have gained popularity. Random search, on the other hand, makes no use of the currently available information and blindly samples every region. For very small size problems, this is probably acceptable, but for any good size problem this method is intolerably inefficient. For searching discrete design spaces, enumerative schemes are probably the most simple and straightforward, but, once again, are effective only if the search space is relatively small. Enumerative schemes evaluate the objective function at every point of the design space, but if the design space is large, this technique becomes prohibitively time consuming.

GAs, on the other hand, are much more robust in their search in multimodal, real valued search space. Search using GAs strikes a reasonable balance between exploiting the

available information and searching through unexplored regions. The feature of survival of the fittest conserves the current useful knowledge and the recombination operators make use of the current knowledge to search unexplored regions of the design space. In contrast to other, more standard, search algorithms, GAs base their progress on the performance of a population of candidate solutions, rather than on a single candidate solution. The motivation behind this is that, by simultaneously searching many areas of the design space, the risk of getting stuck at a local optimum is greatly reduced.

GAs have been established that they are more robust in their search than other conventional algorithms. GAs are computerised search algorithms based on the mechanisms of natural selection. They lie on one of the most important principle of Darwin: *survival of the fittest*. John Holland, in the 1970s, thought that he could incorporate such a technique in a computer algorithm, to solve difficult problems through evolution. This technique, close to the laws of nature, uses a population of potential solutions represented by string of binary digits, called chromosomes or individuals, which is submitted to many transformations, called genetic operations such as *selection*, *crossover* and *mutation*. The population is going to evolve during the generations according to the fitness value of the individuals; then, when a stationary state is reached, the population has converged to the solution of the given optimisation problem. Goldberg (1989) introduced a significant thrust in the research field of GAs.

GAs are different from normal optimisation procedures, e.g. simple or conjugate gradient methods or steepest descent methods, in many ways:

- They work with a coding of the parameter set and not the parameters themselves. The advantage of working with a coding of parameter set is that the coding discretizes the search space, even though the function may be continuous. On the other hand, since GAs require only function values at various discrete points, a discrete or discontinuous function can be handled with no extra cost. This allows GAs to be applied to a wide variety of problems.
- They work simultaneously with a population of potential binary coded solutions, not only with one solution. So, it is very likely that the expected GA solution may be a global solution. Also, multiple optimal solutions can be captured in the population easily.

- They use probabilistic rules, the genetic operators are applied with probabilities, and not deterministic ones.
- They investigate in a search space composing a database of the solutions, which implies that they cannot fall into a local optimum;
- Two keywords are linked to GAs: *exploration* and *exploitation*. Exploration of the search space is important at the beginning of the GA process while exploitation is desirable when the GA process is close to the global optimum.

3.2 GAs: WORKING PRINCIPLES

The basic structure of a GA consists mainly of the following steps: (1) Random initialisation of a population of individuals. (2) Evaluation of individuals, which takes a member of the population and decides about its fitness (cost functional) value according to the information that it encodes (most GAs use a simple binary encoding "string of bits" to store the solution information). The evaluation function, or fitness function, is domain dependent and unique to every encoding of every problem, the closer that an individual is to encoding the ideal solution to the given problem, the higher the fitness of that individual. (3) Application of genetic operators (selection, crossover and mutation) to the population and return to step 2 until the best individual is reached. In the next subsections, Genetic operators will be described:

3.2.1 Selection process (or reproduction)

Selection is the process that copycats *survival of the fittest* observed in natural genetics. It is usually the first operator applied on a population. Once the population of search points are evaluated and given a fitness value based on their predicted "goodness" in searching for the optimal value, selection probabilistically selects a few good points and moves them over to the next generation. In other words, it picks randomly two individuals from the current population and selecting the best according to its fitness. This best individual is retained for the next (new) population. The two individuals are then put back in the current population and the process is restarted until a new population is completed. This is a way of maintaining the useful information already available.

Several different methods may be used for selection, each with a slightly different effect. One common technique assigns each chromosome a probability of selection that is proportional to the individual's fitness divided by the fitness of the entire population. This selection can be easily implemented using a "roulette-wheel", which is an imaginary wheel. This wheel is split up into as many parts (slices) as the population, which assign to each individual a slice of the wheel proportional to the fitness of the individual.

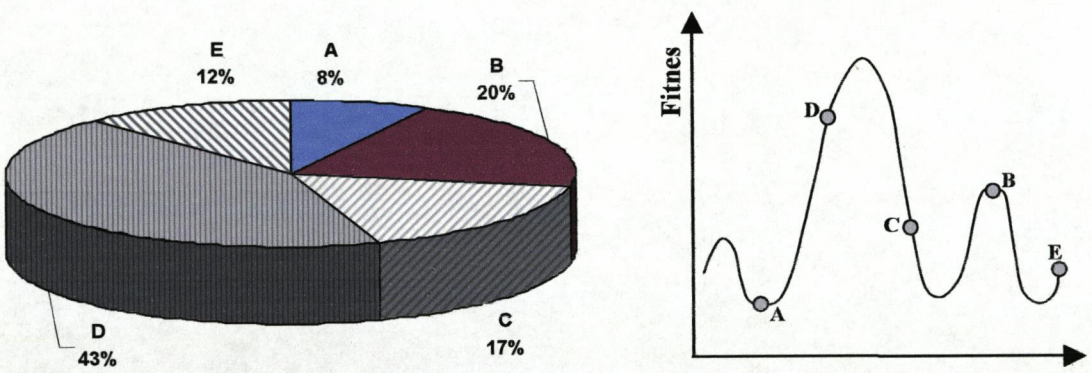


Figure 3.1, Selection process using roulette-wheel.

As an example, let us imagine that the search is being conducted from five points; A, B, C, D, and E. Let the relative fitness of each be 8%, 20%, 17%, 43%, and 12% respectively. Then the wheel would look like the wheel shown in Figure 3.1. Now if the wheel is spun, when the wheel stops the probability that the arrow would be on A is 0.08, on B is 0.20 and so on. This means that the probability of D (the predicted best point to lead to the optimal value) being selected is the maximum and the probability of A (the predicted worst point to lead to the optimal) being selected is the minimum. Essentially, the role of selection is to ensure that the points that are predicted to lead quickly to the optimal value are preserved. For more details, see Deb (1995) and Goldberg (1989).

3.2.2 Crossover operator

As reproduction does not create new individuals, the crossover operator is needed to increase diversity between the population. Crossover proceeds in two steps. First, two

individuals are selected from the existing population. The chances of any individual being selected for crossover are dependent on the fitness value of that point. The higher the fitness of an individual, the higher its chances of being selected for mating. Second, each pair of individuals (known as the parents) undergoes crossing-over to create two new individuals (known as offspring or children) by swapping all characters between certain position. The idea is that by rearranging the genetic material of the parent chromosomes, one or both of the new chromosomes will contain the good properties of each, resulting in better chromosomes. For example, one-point crossover begins by choosing a random point somewhere in the middle of the two selected chromosomes. Taking all of the information to the left of the crossover point of one parent and all information to the right from the other parent then creates the first new chromosome. The same is done for the second chromosome, only the left side is taken from the second parent and the right is taken from the first, see in figure 3.2. As with selection, there are several methods for performing crossover and detailed descriptions can be found in Goldberg (1989).

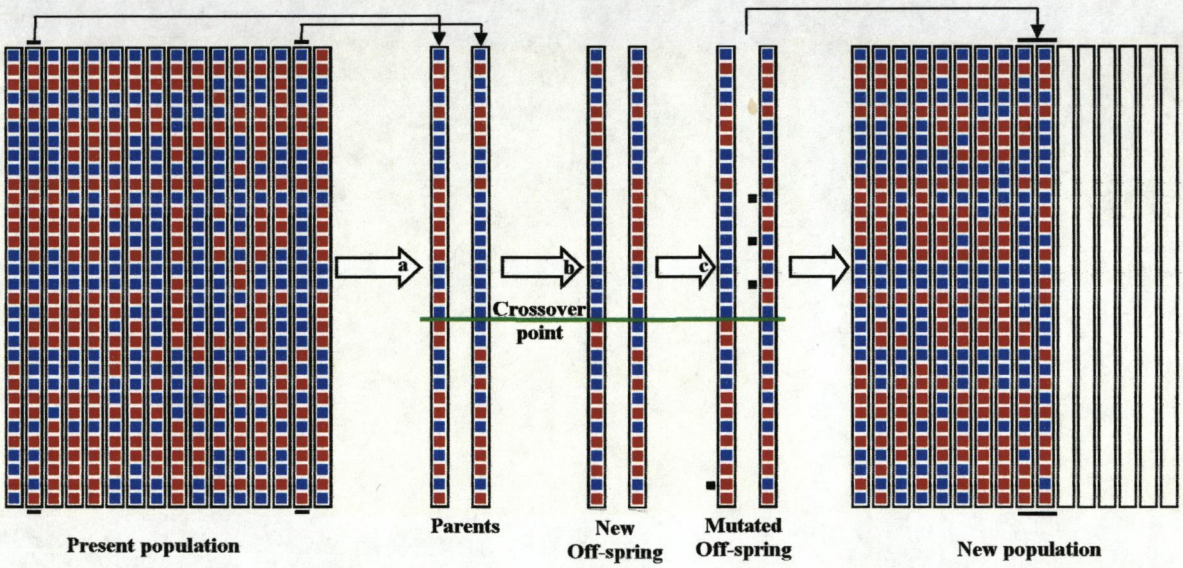


Figure 3.2, Structure of Genetic Algorithm, represents:
a) Selection; b) Crossover; c) Mutation.

3.2.3 Mutation operator

The mutation is the final basic genetic operator for GA. It is needed because reproduction and crossover can occasionally lose some potentially useful genetic material, by creating a point in the neighbourhood of the current point (local search around the current solution). Mutation is then an insurance policy against premature loss of important notions. This works by randomly changing parameters in a chromosome with low probability because it might otherwise cause the GA to become too randomly in its search. In a binary encoding context, this simply means, changing a 1 into a 0 and vice versa, randomly, with a small probability. For example, as seen in figure 3.2, take string equal 01110 and assume that position 3 is chosen randomly to mutate. The new string would be 01010. The mutation is also used to maintain diversity in the population.

3.2.4 GAs termination

There are several ways to stop a GA. One method is to stop after a predetermined number of generations or function evaluations. Another is to stop when the average quality of the population does not improve after some number of generations. Another common alternative is to halt the GA when all the individuals are identical, which can only occur when mutation is not used.

3.3 MULTI-OBJECTIVE OPTIMISATION

In every day life, several scores must be satisfied simultaneously to obtain an optimal solution. Multi-Objective Optimisation (MOO) extends optimisation theory by permitting multiple objectives to be optimised simultaneously. MOO is known as multicriteria, multidisciplinary, or a vector optimisation problem. Some of the main approaches proposed in the literature are presented below, for a detailed survey of this methods see Coello (1999).

3.3.1 Vector Evaluated VEGA

Schaffer (1985) modified the selection operator of a Simple Genetic Algorithm (SGA) so that at each generation a number of sub-populations was generated by performing proportional selection according to each objective function in turn. These sub-

populations would be rearranged together to obtain a new population, on which the GA would apply the crossover and mutation operators in the usual way.

3.3.2 Hierarchical

The basic idea of this technique is that the designer ranks the objectives in order of importance. The optimum solution is then found by minimising the objective functions, starting with the most important one and proceeding according to the order of importance of the objectives, Rao (1984). Another version of the algorithm reported by Fourman (1985) consisted of randomly selecting the objective to be used at each generation.

3.3.3 Weighted sum

This is perhaps the most commonly used method because of its simplicity. This method takes each objective function and multiplies it by a fraction of one "weighting coefficient". These modified functions are then added together to obtain a single cost function. Hajela and Lin (1992) included the weights of each objective in the chromosome, and promoted their diversity in the population through fitness sharing. Their goal was to be able to simultaneously generate a family of optimal designs corresponding to different weighting coefficients in a single run of the GA. Besides using sharing, Hajela and Lin used a vector evaluated approach based on VEGA to achieve their goal.

3.3.4 Multiple objective genetic algorithms

Fonseca and Fleming (1993) have proposed a scheme in which the rank of a certain individual corresponds to the number of chromosomes in the current population by which it is dominated. All non-dominated individuals are assigned rank 1, while dominated ones are penalised according to the population density of the corresponding region of the trade-off surface.

3.3.5 Non-dominated sorting genetic algorithm

The Non-dominated Sorting Genetic Algorithm (NSGA) was proposed by Srinivas and Deb, 1993, and is based on several layers of classifications of the individuals. Before the selection is performed, the population is ranked on the basis of non-domination: all non-dominated individuals are classified into one category (with a dummy fitness value,

which is proportional to the population size, to provide an equal reproductive potential for these individuals). To maintain the diversity of the population, these classified individuals are shared with their dummy fitness values. Then this group of classified individuals is ignored and another layer of non-dominated individuals is considered. The process continues until all individuals in the population are classified.

3.3.6 Niched Pareto GA

Horn and Nafpliotis, 1993, proposed a tournament selection scheme based on Pareto dominance. Instead of limiting the comparison to two individuals, a number of other individuals in the population were used to help determine dominance. When both competitors were either dominated or non-dominated (i.e., there was a tie), the result of the tournament was decided through fitness sharing (Goldberg and Richardson, 1987). Population sizes considerably larger than usual were used.

In the present work, the hierarchical approach was used for its simplicity although it has the disadvantage of limiting the choices available to the designer, making the optimisation process a rather difficult task. The approach allows the designer to rank the objectives in a descending order of importance, from 1 to k. Each objective function is then minimised/maximised individually subject to an additional constraint. This additional constraint, equation 3.1, does not allow the minimum/maximum for the new object to exceed a prescribed fraction of a minimum/maximum of the previous object. This approach and others are presented in Fonseca and Fleming (1996).

$$f_j(\bar{x}) \leq \left(1 \pm \frac{\varepsilon_j - 1}{100}\right) f_{j-1}\left(\bar{x}^{(j-1)}\right), \qquad \text{for} \quad j = 2,3,...k \tag{3.1}$$

where, ε_j is the assumed coefficient of the function increment, expressed in percent.

$\leq$ for minimisation, and $\geq$ for maximisation.

Also, weighted sum is used in the present work. Some weights are tested, and comparison between it and hierarchical approach has been taken place in Chapter six.

3.4 CONSTRAINT HANDLING METHODS

Many real-world search and optimisation problems involve inequality and/or equality constraints and are thus posed as constrained optimisation problems. A constrained

optimisation problem is usually written as a non-linear programming (NLP) problem of the following type:

$$\textbf{Minimise } f(\vec{x}) \tag{3.2}$$

subject to:

$$g_j(\vec{x}) \geq 0, \qquad j = 1, J \tag{3.3}$$

$$h_k(\vec{x}) = 0, \qquad k = 1, K \tag{3.4}$$

$$x_i^l \leq x_i \leq x_i^u, \qquad i = 1, n \tag{3.5}$$

$f(\vec{x})$ is the objective function with number n of variables, $\vec{x}$, and J, K are number of inequality and equality constraints, respectively. $g_j(\vec{x})$ and $h_k(\vec{x})$ are the j^{th} inequality and k^{th} equality constraints, respectively. The parameter x varies in the range $\left[x_i^l, x_i^u\right]$.

Constraint handling methods used in classical optimisation algorithms can be classified into two groups: generic methods and specific methods that are only applicable to a special type of constraints. Generic methods, such as the penalty function method, the Lagrange multiplier method, and the complex search method, Deb (1995), are popular, because each one of them can be easily applied to any problem without much change in the algorithm. However, specific methods, such as the cutting plane method, the reduced gradient method, and the gradient projection method, Deb (1995), are applicable either to problems having convex feasible regions only or to problems having a few variables, because of increased computational burden with large number of variables. In the next two subsections, penalty function methods and penalty function based on feasibility will be presented where the later one will be used in the present work.

3.4.1 Penalty Functions

The most common approach in evolution algorithms (mainly with GAs) community to handle constraints (particularly, inequality constraints) is to use penalties. The penalty function approach involves a number of penalty parameters, which must be set right in any problem to obtain feasible solutions, as seen:

$$F(\vec{x}) = f(\vec{x}) + \sum_{j=1}^{J} R_j \left| g_j(\vec{x}) \right|^2 \tag{3.6}$$

$F(\vec{x})$ is defined as the sum of the objective function $f(\vec{x})$ and a penalty term which depends on the constraint violation $g_j(\vec{x})$. R_j is to make the constraint violation of the same order of magnitude as the objective function value. Equality constraints, equation (3.4), are usually handled by converting them to inequality constraints as follows:

$$g_{j+1}(\vec{x}) \equiv \delta - |h_k(\vec{x})| \geq 0 \tag{3.7}$$

where, δ is a small positive value.

By penalty function, the constrained problem is transformed into an unconstrained problem. The existence of penalty parameter causes the optimal solution to depend on it, and users usually have to try different values of it, to find what value would be the best.

Because of dependency of GA's performance on penalty parameters researchers devise sophisticated penalty function approaches such as multi-level penalty functions, Homaifar et.al. (1994), dynamic penalty functions, Joines and Houck (1994), and penalty functions involving temperature-based evolution of penalty parameters with repair operators, Michalewicz and Attia (1994). All these approaches require extensive experimentation for setting up appropriate parameters needed to define the penalty function. Michalewicz (1995) described the difficulties in each method and compared the performance of these algorithms on a number of test problems. In a similar study, Michalewicz and Schoenauer (1996) concluded that the static penalty function method is a more robust approach than the more sophisticated methods. This is because one such sophisticated method may work well on some problems but may not work so well in another problem. Deb, 2000, developed a constraint handling method, which is based on the penalty function approach which does not require any penalty parameter.

3.4.2 Penalty function based on feasibility

Deb (2000) proposed an interesting approach, which is based on penalty functions but does not require any penalty parameter and can offer distinction between feasible and infeasible solutions. Deb uses the following rules to compare two individuals:

1. A feasible solution is always preferred over an infeasible one.
2. Between two feasible solutions, the one having better objective function is preferred.

3. Between two infeasible solutions, the one having smaller constraint violation is preferred.

In other words, in order to evaluate any solution, it is a usual practice to first check the feasibility of the solution. If it is feasible, its objective function value is computed. If the solution is infeasible (at least one constraint is violated), the designer will never bother to compute its objective function value. It does not make sense to compute the objective function value of an infeasible solution, because the solution simply cannot be implemented in practice. Here, the idea of comparing infeasible solutions depends only on terms of constraint violation. Mathematically, the fitness function can be in the following form:

$$F(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if } g_j(\vec{x}) \geq 0 \\ f_{\max} + \sum_{j=1}^{J+K} |g_j(\vec{x})| & \text{otherwise} \end{cases} \tag{3.8}$$

The parameter f_{max} is the objective function value of the worst feasible solution in the population. Thus, the fitness of an infeasible solution not only depends on the amount of constraint violation, but also on the population of solution at hand. However, the fitness of a feasible solution is always fixed and is equal to its objective function value.

3.5 PARALLEL GAs

The basic idea behind most parallel programs is to divide a task into large pieces and to solve these simultaneously using multiple processors. This divide-and-solve approach can be applied to GAs in many different ways, and the literature contains many examples of successful parallel implementations. Some parallelisation methods use a single population, while others divide the population into several relatively isolated sub-populations. Some methods can exploit massively parallel computer architectures, while others are better suited to multi-computers with fewer and more powerful processing elements.

There are three main types of parallel GAs: (1) global single-population master-slave GAs, (2) single-population fine-grained, and (3) multiple-population coarse-grained GAs. In the next sub-sections, a brief idea about these types of structure and their advantages and disadvantages will be presented.

3.5.1 Micro-Grain GA

The micro-grain GA (mgGA), also referred to as master/slave GA, is the simplest parallel GA model. It is unique from the other parallel GA methods in process, where, there is a single population (just as in a simple GA) which is maintained principally by a master, but the evaluation of fitness is distributed among several processors acting as slaves (see Figure 3.3). Since in this type of parallel GA, selection and crossover consider the entire population, it is known also as global parallel GA. Ideally, there should be one processor for each individual in the population, but the chromosomes can be distributed among any number of slave processors. Maximum speedup can be obtained when each of the slaves receives an equal amount of work.

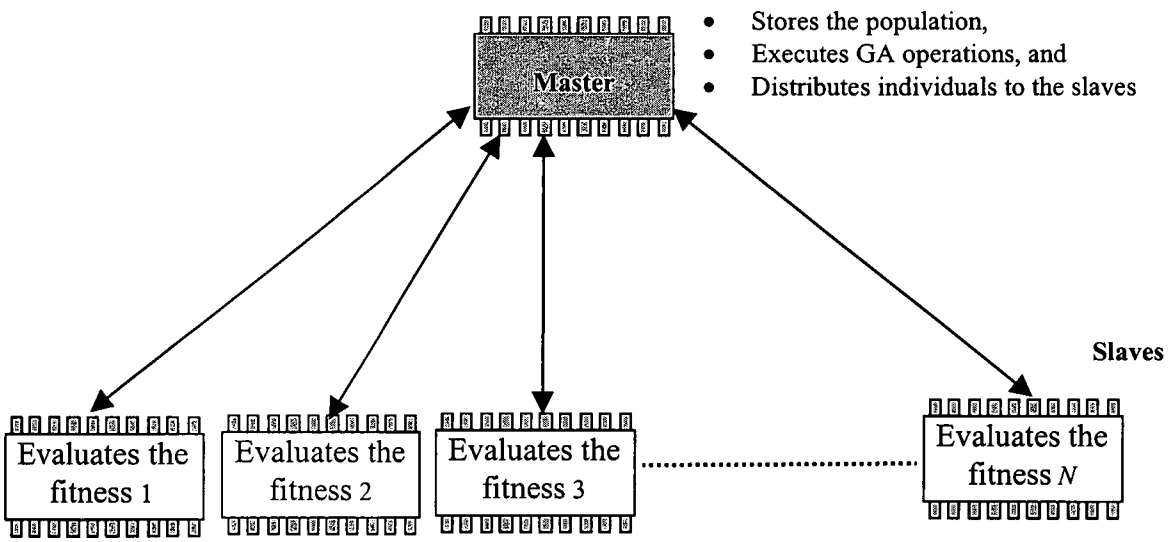


Figure 3.3, A schematic of a master/slave parallel GA.

The greatest advantage to the micro-grain method is its simplicity. This method does not require a particular network topology, although a highly connected network would be best to reduce communication overhead. The algorithm is roughly equivalent to the serial GA. The micro-grain is generally a good method for gaining speed with the genetic algorithm without introducing extra complexity issues.

3.5.2 Fine-Grain GA

The fine-grain GA (fgGA) model is a compromise between the single global population model described above and the models with several fully separated populations

discussed below. The fgGA may be equivalently viewed in two different ways. It can be modelled as a single global population with the restriction that a chromosome may only interact (or mate) with its neighbours. For example, the population can be mapped to one of many different topologies (i.e. linear, mesh, ring, etc). When selection is performed for crossover, only adjacent chromosomes are eligible to be selected together. Each chromosome will be neighbour to several other chromosomes, allowing genetic material to pass from one "neighbourhood" to the next. A fgGA mapped to a fully connected network would be roughly equivalent to a micro-grain genetic algorithm. A second way to view a fgGA is to show several separate but overlapping sub-populations. When selection is performed, only chromosomes within the same population may mate, but many chromosomes will be members of multiple populations and therefore free to pass material from one population to another.

Figure 3.4 shows how the population in fgGA might be divided. Again, if the algorithm is mapped to a fully connected network, then every individual could potentially be a member of every population, causing the fgGA to degenerate into mgGA. The purpose of the fgGA is to slow the spread of genetic information through the population while still allowing some information to migrate from one sub-population to another. One of the more significant issues associated with the fgGA is deciding what topology to use when mapping the GA population to a network. A highly connected network with low diameter will allow information to flow from one sub-population to another more quickly. This increases the chance that the GA may become dominated by one or a few strong individuals, causing the entire population to converge too soon. However, if the network topology is too sparse, then each sub-population will be very small and information will move slowly across the population. In this case, the algorithm may run very slowly, requiring an excessive number of generations to find the best solution, Goodman et al. (1994).

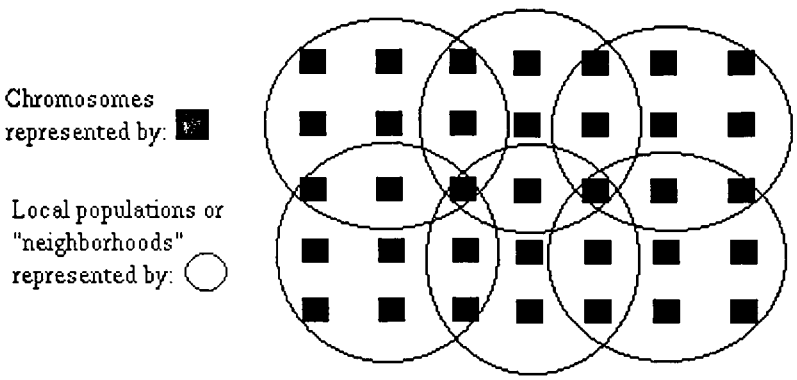


Figure 3.4, A schematic of a fine-grain parallel GA.

There are a few advantages to the fine-grain GA. For example, the representation is reasonably simple and avoids some of the more complicated migration issues that arise with the coarse-grain GA described in the next section. The boundaries between sub-populations are unclear, so there is no need for an explicit migration policy to allow information to move through the population. However, local populations are isolated enough to help prevent early domination of the entire population by a few strong individuals. This method is also very well suited to a multi-processor. All interaction is done at the "edges" of the populations, so there is no need to transport data across large sections of the network. The majority of exchanges will take place between neighbouring or nearby processors, thereby reducing communication overhead to a necessary minimum.

3.5.3 Coarse-Grain GA

The coarse-grain model of genetic algorithms (cgGA) is somewhat similar to the fine-grain model. The population is again divided into multiple sub-populations, but this time the boundaries between are sharp rather than unclear. In the fgGA model, information diffused through the network simply by gaining membership in new populations. Migration was inherently a part of the fgGA structure. This is not the case with cgGAs where migration must be explicitly handled, as seen in figure 3.5, rather than leaving migration to a random process of drifting as in fgGAs, or ignoring the potential benefits of dividing and migrating the population as the simple mgGA does. The implication of this theory is simple. A single large population will eventually converge, causing new chromosomes formed to have only insignificant differences from

their parents. This problem may be resolved by evolving many (probably smaller) populations simultaneously and independently. The populations will eventually converge on competitive, yet hopefully unique solutions. Evolution can then continue by swapping in some new material from other populations, which should increase local diversity.

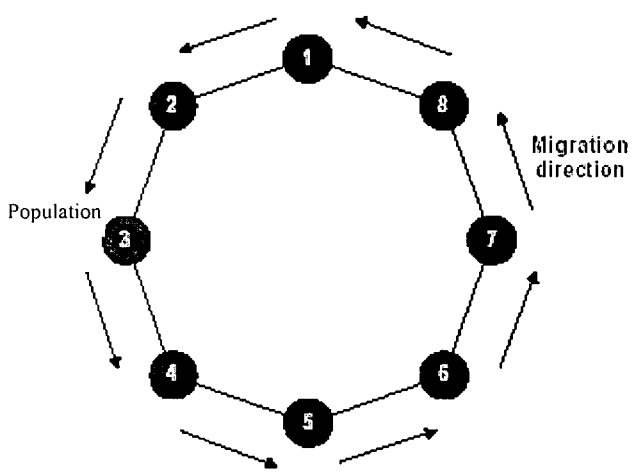


Figure 3.5, A schematic of multiple-population parallel GA. Each process is a simple GA, and there is (infrequent) communication between the populations.

The coarse-grain model is one of the most robust forms of genetic algorithms. The reduced need for inter-process communication helps to reduce one of the major restricted accesses in parallel processing so increasing overall efficiency. Even more significant is the ability to adapt many aspects of a cgGA to each specific problem. Unfortunately, this is also one of the greatest disadvantages with the coarse-grain method.

3.6 CONCLUSION

This chapter contained four main sections, GAs working principles; MOO methods; constraint handling methods; and parallel GAs methods. From the first section, it can be concluded that: GAs are original systems based on the supposed functioning of the Living. The method is very different from classical optimisation algorithms.

- Use of the encoding of the parameters, not the parameters themselves.
- Work on a population of points, not a unique one.

- Use the only values of the function to optimise, not their derived function or other auxiliary knowledge.
- Use probabilistic transition function not deterministic ones.

Six methods of casting MOO were presented in the second section, where hierarchical approach and weighted sum method are recommended to be used in the current work. And because the hierarchical method is using the multi-objectives as constraints, two methods of handling constraints are presented in the third section. In addition, it has been suggested to use Deb's method in this work. Finally, because of this project requires to link GA and CFD, which is computationally very expensive, it is recommended to use parallel GAs. Hence, parallel GAs methods are presented in the fourth section, which can be summarised follows:

There are many methods available for parallelising GAs. Each method entails a slightly different set of advantages and disadvantages. The parallel GAs described above have some potential for a speed increase over the standard serial GAs, but the speedup is not recommended for a simple distribution of work. While the micro-grain (master/slave) GA does rely on distribution of computation for its performance increase, the other methods rely on more complicated mechanisms. Fine-grain GAs, for example, use overlapping "neighbourhoods" of chromosomes to reduce the probability of premature convergence in addition to distributing the population over many processors. Both speed and effectiveness of the search are therefore enhanced by a fine-grain GA. Coarse-grain GAs moves one step further by segregating the population entirely and allowing only an occasional exchange of information. The coarse-grain GA is simply a network of tightly coupled serial GAs, so any increase in speed will be minor in comparison to the increase in cost. In this work, micro-grain (master/slave) GA is used because of its simplicity.

Chapter 4

Impeller Design Procedure

4.1 INTRODUCTION

Several pump design procedures are described in the literature. These procedures concern themselves primarily with design point operation. The design procedure advocated here will focus on impeller blade profile. First, whole pump design procedure will be described followed by old and suggested procedures used to design impeller blade profiles.

Since the head can hardly exceed U_2^2/g, a fast check can be made to see if a reasonable impeller size can achieve the desired head or if several stages will be required, where g is gravity acceleration and U_2 is the tip speed, it can be calculated from:

$$U_2 = \left(\frac{2\pi n}{60}\right)\frac{D_2}{2} \tag{4.1}$$

The number of impeller blades can be selected on a trial basis, usually six. More blades guide the flow better, increase the slip coefficient, and therefore increase the head somewhat. On the other hand, friction losses are increased. The value of the impeller inlet diameter D_1 is usually selected to minimise the inlet relative velocity. Low velocities favor low losses (Vlaming, 1989). Low inlet relative velocities minimise the diffusion, the ratio between inlet and exit relative velocity W_1/W_2 in the impeller, which may lead to flow separation if excessive. If pump specifications require a low

NPSHR value, then minimising the inlet relative velocity may not have first priority, and the possibility of cavitation must be examined in detail. Selection of the inlet diameter, blade angles, and location of the leading edge of the blades must proceed by trial and error. The impeller width B from hub to shroud can be calculated from the relative velocity in the radial direction W_r and the flow rate Q at a given radius r:

$$B = \frac{Q}{2\pi r W_r} \tag{4.2}$$

Again, from Euler's fundamental equation it follows that the total head generated by a pump depends on many variables such as; the peripheral velocity U_2 and the meridional velocity C_{m2} at the impeller outlet, the blade angle β_2, the number of blades z, and the inlet to outlet impeller diameter ratio D_1/D_2. The same total head may be attained with a smaller peripheral velocity U_2, by using an impeller of smaller diameter (keeping the same rotational speed n) but having a greater angle β_2 and a greater number of blades z. Lazarkiewicz (1965) divided the design of the impeller to sections, impeller inlet, impeller outlet, and blades design.

In the present work the process of blade design will be focused in two dimensions (2D) by using GA optimisation linked with CFD. The description will contain traditional methods used before, and how to apply the optimisation one by parameterising the blade shape using Bézier curves. Finally, the optimisation procedure will be considered using one and multiple processors; and how it is linked with CFD solver.

4.2 BLADE DESIGN

The desired suction head and correct incidence at the pump inlet determine the inlet blade angle β_1 from the tangential direction, and the desired pump head determines the exit blade angle β_2 from the tangential direction. And as well, the major dimensions of the impeller; inlet diameter D_1, exit diameter D_2, and impeller width B; can be calculated. It does not however define the geometrical shape of the impeller. The increment change in the angular momentum corresponds to the pressure difference exerted on the blade, the blade loading. Ideally, it should increase gradually from the

inlet, reach a maximum at the middle of the blade length, and taper off at the exit (Tuzson, 2000). Heavy blade loading at the inlet may increase inlet losses. Heavy blade loading at the exit may create exit losses, pressure fluctuations, and noise. Therefore, the angular momentum is gradually incremented from the inlet to its exit value. Where the angular momentum rC_t is a stepwise-increasing fraction of its value at the impeller exit:

$$R_2 C_{t2} = \frac{gH}{\eta\omega} \tag{4.3}$$

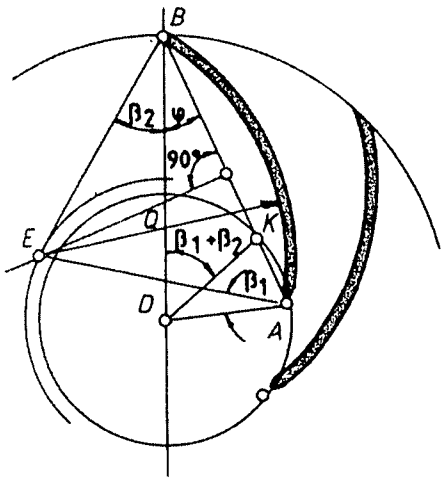
In impellers with two-dimensional blades, the hub and shroud lines coincide, the axial coordinate z is constant, and the other two coordinates (radius r, and circumferential angle θ) are enough to define the blade shape. Traditionally, the angular momentum along the streamlines has to be calculated at each r and θ.

Consequently, the main dimensions of the impeller (D_1, B_1, D_2, B_2) and angles β_1 and β_2 can be calculated; these parameters do not however define the shape of the impeller. For this reason, we have to study methods that were used before to design the blades shape.

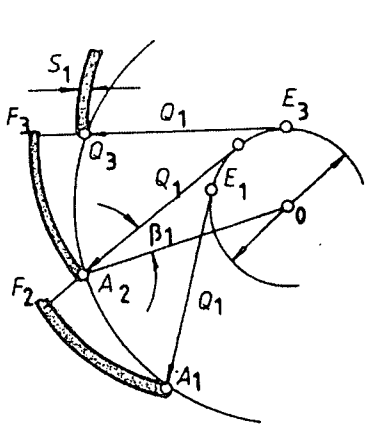
4.2.1 Traditional Methods

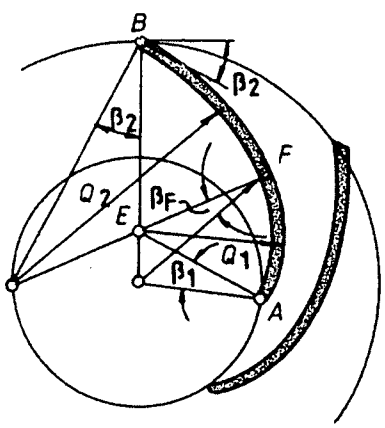
There were three principal methods for determining the shape of blades: circular arc method, point by point method, and the conformal representation method.

- *Circular arc method* is the oldest and simplest method. No attempt is made to obtain a gradual change of the velocities or the blade inclination angle. So, it is therefore difficult to shape the impeller passage correctly. The angle of inclination of the blade changes greatly along the blade and intermediate values of blade angle may be considerably larger than the outlet one. One or two arcs of a circle may define the blade; the latter however, gives better results, see figure 4.1.



a) One arc





b) Two arcs

Figure 4.1, Shape of impeller passages using circular arc method.

- *Point by point method* is based on the assumption that the transition from the inlet to the outlet blade angle depends on the radius. A smooth curve is drawn through a series of pre-determined points, see figure 4.2. This method gives the designer the greatest freedom in shaping the blades.

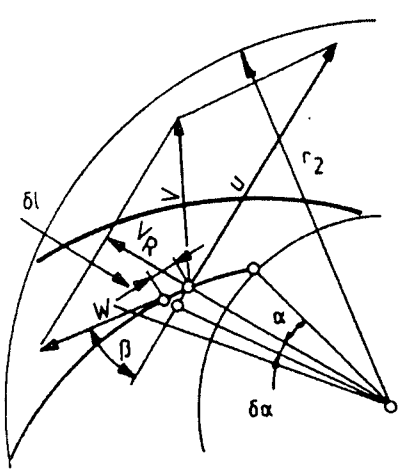


Figure 4.2, Shape of impeller passages using point-by-point method.

- *Conformal representation method* is widely used in USA under the name method of error triangles. This method is relatively simple and less difficult than the point by point method of determining blade shapes.

From the above discussion it can be seen that, methods have been used before provide the same profile to all impellers, and there is no impeller profiles classification based on their objectives. For example: the length of the blades, and hence the passage length, can be different for the same radii R_1 and R_2, the same angles β_1 and β_2 and the same number of blades. In short passages the angle of divergence may be excessively large, which increases the chance of separation and formation of damaging eddies; if the passages are very long and the angle of divergence very small, the losses due to separation are reduced, while the friction losses are increased. The sum of all losses should be kept to minimum for the highest possible efficiency; hence it is necessary to make some compromise. In this respect the results of investigations carried out by research engineers in the Hydromechanical Institute of the Polytechnic in Braunschweig are very instructive, Lazarkiewicz (1965). They compared six impellers with the same number of blades and angles but with blades of varying length, forming passages with various angles of divergence, as seen in figure 4.3. The best shape was given by impeller II, in spite of the relatively large angle of divergence of the passage.

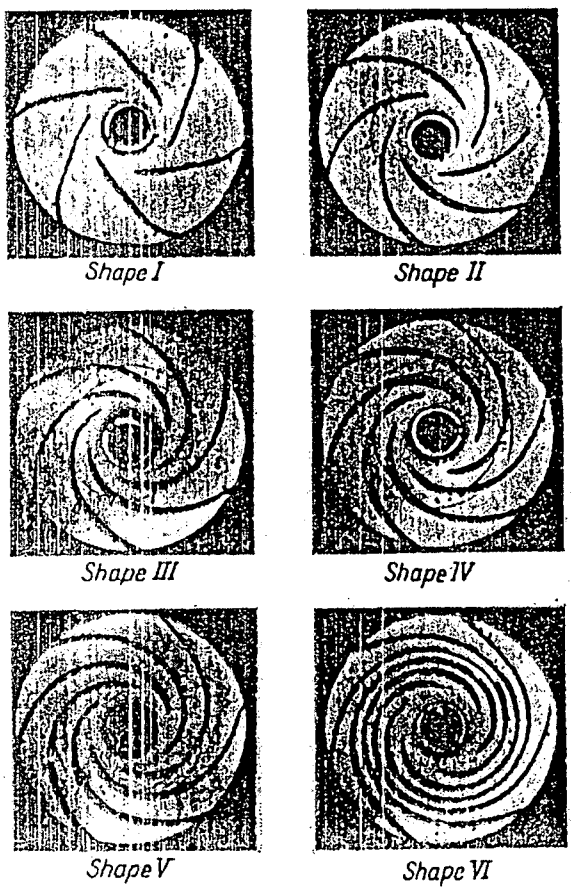


Figure 4.3, Shapes of impeller passages.

Shape I: short blades with small frictional surface, but with large losses due to separation; Shape II: optimum shape of blade; shapes III-VI: as the blade length increases, so also do the friction losses, but the losses due to separation decrease. The sum of the losses is greater than for shape II (Lazarkiewicz, 1965).

4.2.2 New Method

With the help of optimisation linked with CFD, the designer can obtain his own impeller design by specifying his needs (objectives) from this impeller with less effort, experience and amount of experimental work. The task is to design the surface of blade profile of a centrifugal impeller, which provides a high performance to the impeller.

For an impeller with inlet and outlet radii R_1, R_2, two consecutive blades, as seen in figure 4.4, define the shape of an impeller passage. The contour of the blade profile to be optimised is defined by a Bézier curve starting at the leading edge and ending just at the trailing edge. The positions of the control points of the Bézier curve are defined by the location of seven or eight points (P_1, P_2, ... P_6) with respect to the start control point

P_0. The line between every two-consecutive control points is called control line, and its length depends on the angle differences between the current control point and previous one from the impeller centre. For seven control points, the first and the last control points (P_1, P_6) defined by the flow inlet and outlet angles β_1 and β_2 with respect to (P_0, P_5) and can be changed by the change of the blade angles. For the remaining control points (P_2, P_3, P_4, P_5) change by the perturbation of difference angles $d\phi_2$, $d\phi_3$, $d\phi_4$, $d\phi_5$, until reaching the optimum solution. These changes are limited within specified overlap angle $\phi_{overlap}$ between two successive blades.

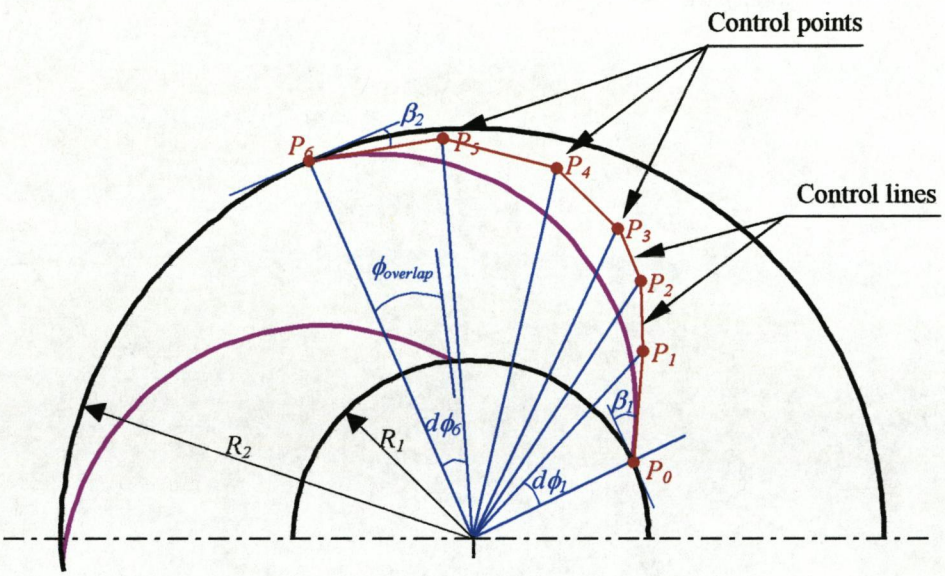


Figure 4.4, Blade definition through Bézier curves.

4.3 OPTIMISATION PROCEDURE

The optimisation problem, in general, is stated mathematically as follows:

$$\textbf{Minimise } F_k\left(Q(\phi),\phi\right) \qquad k = 1,\ NOBJ \tag{4.4}$$

subject to the following constraints:

1. flow-type inequality constrains

$$G_j\left(Q(\phi),\phi\right) \le 0, \qquad j = 1, NCON_f \tag{4.5}$$

2. geometric-type inequality constraints

$$G_j(\phi) \le 0, \qquad j = NCON_f + 1, NCON \tag{4.6}$$

3. side constraints

$$\phi_{LE} \le \phi \le \phi_{TE} \tag{4.7}$$

where, F_k denotes the objective functions and Q is the vector of the conserved variables of the fluid flow (solved by CFD analysis "flow solver"). *NOBJ* denotes the number of objective functions, $NCON_f$ is the number of flow-type constraints and *NCON* is the total number of inequality constraints. ϕ_{LE} and ϕ_{TE} are the lower and the upper bounds of the design variables. A sensible choice of these bounds is necessary to maintain an acceptably smooth hydrodynamic shape.

It appears that there are two cornerstones in the design of the problem in order to formulate the optimisation problems: design variables and design objectives.

4.3.1 Design Variables

An important step in the formulation of any optimum design problem is for the designer to explicitly identify an appropriate set of design variables. The countless number of different ways to do so makes the decision of which parameters to pre-assign and which to consider free, far from trivial. The requirement that the chosen design variables must influence the performance measures; mostly requires them to influence the CFD solutions. This leads to the following categorisation of relevant types of design variables, as originally suggested for structural optimisation by Olhoff and Taylor (1983).

- *Geometrical design variables*

  These variables are classified into three classes. The first one is sizing design variables, which represent a description of the dimensions of geometrical properties. The second class is shape design variables, which represent a description of the surfaces bounding the fluid flow. Whereas, the third category is topological design variables, which represent a description of the topological properties of the flow domain and do to control the design domain.

- *Loading design variables*

  These variables describe position and distribution of some boundary conditions.

- *Material design variables*

  These variables describe properties of the fluid, e.g. the viscosity.

- *Manufacturing design variables*

  These variables describe parameters relating to manufacturing processes, which may influence the performance and cost of the construction, e.g. relative surface roughness.

The work presented in this thesis concerns shape optimisation, for which reason the most interesting class of design variables are the geometrical ones.

4.3.2 Design Objectives

It is now in place to consider the criteria, objectives and constraints, used in defining optimum design problems, equations 4.4 to 4.7. From a mathematical point of view, it is possible to divide these into the following categories:

- *Integral criteria*

  In shape optimisation for fluid devices, these cover global properties of the flow-field or geometry. For example loss minimisation, the objective is to minimise the overall energy loss in a flow device. Another example is to maximise the possible mass flow rate in the centrifugal impeller given a pressure rise between inlet and outlet.

- *Local criteria*

  These cover any sub properties of the flow-field or geometry. For example, to maximise a velocity component at a certain location with the object of providing optimal ventilation of risky chemicals changes.

- *Min/Max and Max/Min criteria*

  These types of criteria are among the most frequently used in practical engineering optimisation problems. A typical application of such criteria is for the fashion design of parameter distributions at certain geometry locations. For example the design of good blade profile to obtain a uniform downstream velocity distribution by minimising the maximum velocity at the positions in question. Further example could be to maximise the minimum static pressure in hydraulic designs, in order to keep it above local saturation pressure and so avoid cavitation.

- *Multicriteria*

  They refer to the kind of problems, where the objective is comprised of a set of distinct criteria. The optimal solution for such vector problems is an element of the set of Pareto optimal designs, and can be weighted to be solved, see section 3.3.

The present work contains examples of designs using min/max, local, and integral types of criteria, whereas the point of including problems of multi-criteria optimisation has to be implemented.

4.3.3 Serial GA

The simplest form of a genetic algorithm is the serial GA. Here, there is one population and only one processor to perform the algorithm, as seen in figure 4.5.

For the current work, an already existing library has been utilised. A flow optimisation procedure by means of a C++ Library of GA (Wall 1996), GAlib version 2.45, consists of initialisation, evaluation, selection, crossover and mutation. GAlib contains four types of genetic algorithms. The first one is the standard "Simple GA" described by Goldberg (1989).

```
- Initial random population
- Evaluate(calculate fitness of individuals)
- While(generation < MAX_GEN_NO)
        select(select N members of population)
        crossover
        mutate
        evaluate
        generation = generation + 1
- EndWhile
- Return(individual with greatest fitness)
```

Figure 4.5 Code for a serial genetic algorithm.

The second is "Steady-state GA" which uses overlapping populations, and the user can control overlapping percentage. The third is "Incremental GA" in which each generation consists of only one or two children. And the last is "Deme GA" which evolves multiple

population in parallel using a steady-state algorithm. GAlib has many other attractive features like three scaling schemes which give the user the choice to select linear, sigma truncation, or power law scaling to scale the fitness scores. Finally, GAlib is fully controllable by the user.

GAlib has been used in the present study in combination with a CFD analysis code (section 4.5) to optimise the geometry of blade profiles of centrifugal pump impellers. As mentioned above, the contour of the blade profile to be optimised is defined (assume zero thickness) by six difference angles $d\phi_i$ starting from the leading edge to the trailing edge. The inlet and outlet angles (β_1, β_2), and inlet and outlet radius (R_1, R_2), as shown in figure 4.4, are kept fixed during the optimisation process. While the difference angles $d\phi_2$, $d\phi_3$, $d\phi_4$, $d\phi_5$, are perturbed to change the geometrical shape using De Casteljau's algorithm, Qiu-Lin Ding (1987).

4.3.4 Parallel GA

There are many reasons for parallelising the genetic algorithm. The most obvious is of course speed. GAs are computationally expensive compared to most of the more deterministic forms of search and optimisation, especially if the objective function was obtained through CFD, where CFD takes a long time to converge. Additionally, the probabilistic approach taken by a GA as it must search a larger area (more objective function call) than a deterministic algorithm. These factors make GAs ideal for parallelisation. In some cases, the GA can be parallelised to the point that each individual chromosome is attributed to its own processor to perform necessary computations. This essentially reduces the running time for each generation to the amount of time required for performing the genetic operations on just one individual. Speed is not the only reason for parallelising a genetic algorithm. Some implementations of parallel GAs have a significantly higher cost than serial GAs, instead they increase the chance of finding the optimal solution.

In this work GAlib has been modified to use any number of processors using a master/slave method, figure 3.3, where the master stores the population, executes GA operations, and distributes individuals to the slaves. The slaves only evaluate the fitness

of the individuals in this time the master evaluates one of the fitness as well, as seen in figure 4.6.

```
- Initial random population                                  <Master>
- Distribute population on slave processes                   <Master>
- For each master & slave process p
        evaluate each chromosome
- Wait until all slaves have returned their results (fitness)
- While(generation < MAX_GEN_NO)                             <Master>
        select(select N members of pop)                      <Master>
        crossover                                            <Master>
        mutate                                               <Master>
        For each master & slave process p
                evaluate each chromosome
        Wait until all slaves have returned their results (fitness)
        generation = generation + 1                          <Master>
- EndWhile                                                   <Master>
- Return(individual with greatest fitness)                   <Master>
```

Figure 4.6 Code for a parallel genetic algorithm.

Message Passing Interface (MPI) is used in this parallelisation. MPI efficiently manages message buffers by sending and receiving directly from the user data structures not by buffers within the communication library and therefore buffering may be totally avoided. A program written for MPI is completely portable, and it is easy to recompile and run on any computer platform. The author's personal view is that, MPI is easier to use than the Parallel Virtual Machine (PVM). GAlib has parallel processing class, which is using PVM. If you compare between this PVM class and the new modification, which has been used in the present work. The modification is so simple where it is only to replace the DefaultEvaluator(GAPopulation &) function with new function called Eva_POP(GAPopulation &), Appendix B contains Eva_POP(GAPopulation &) function.

4.4 SHAPE PARAMETERISATION

In the 2D optimum shape design problems considered here, different impeller profiles represent the population of individuals. A natural parameterisation of the blade profile could be a point by point one, based on the discretisation of the shape. Such an approach presents two drawbacks:

- in order to obtain accurate results, the shape has to be designed by a lot of points and it is well known that the convergence of GAs depends on the number of parameters (Goldberg, 1998);
- the point to point representation is not fitted to the crossover operator. Indeed, the crossing-over of two individuals may create two new individuals that are non-feasible.

For this reason, several authors (Périaux, 1995; Poloni, 1995; and Quagliarella, 1995) adopted a strategy, in which the shape parameterisation procedure is based on parametric curves, like B-splines, qubic splines, and Bézier curves. A few control points are then sufficient properly to represent the whole shape.

4.4.1 Bézier curves

Bézier curves were discovered simultaneously by Paul de Casteljau at Citroen and Pierre E. Bézier at Renault around late 50s and early 60s. Bézier's technique is one of the most famous in computer-aided geometric design. Its basic idea is to find curves which only approximate or approach the given points, rather than passing through them like a qubic spline. This approach scheme is more convenient to the designer and the design process, Qiu-Lin Ding (1987). On the other hand, the smoothness properties of Bézier curves never imply the creation of non-feasible shapes by the crossover operator. Bézier curves have a useful convex hull property that restricts the curve to never leave the bounding polygon of the control points. The convex hull property is derived from the fact that a Bézier curve is a convex combination of the data points, figure 4.7.

As seen in figure 4.7, Bézier curve of order n is defined by the *Bernstein polynoms* $B_{n,i}$:

$$P(u) = \sum_{i=0}^{n} B_{n,i}(u) P_i \tag{4.8}$$

where the coefficients, the *Bézier coefficients*, are defined as follows:

$$B_{n,i}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i} \tag{4.9}$$

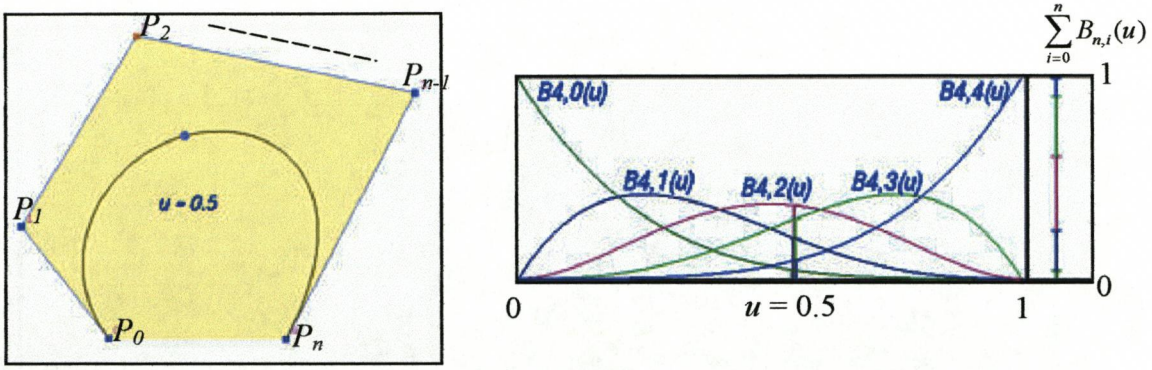


Figure 4.7 Bézier curve defined by five control points and Bézier coefficients

and the point that corresponds to u on the Bézier curve is the "weighted" average of all control points, where the weights are the coefficients $B_{n,i}(u)$. The line segments, P_0P_1, P_1P_2, ..., $P_{n-1}P_n$, are called *control segments*. The functions $B_{n,i}(u)$, $0 \le i \le n$, are usually called the *Bézier basis* functions. Note that the domain of u is [0,1]. If u is not [0,1] the Bézier basis functions will change to:

$$B_{n,i}(u) = \frac{n!}{i!(n-i)!} \left(\frac{u-a}{b-a}\right)^i \left(1 - \frac{u-a}{b-a}\right)^{n-i} \tag{4.10}$$

These new Bézier coefficients define a Bézier curve on the domain of $[a,b]$.

$P_i = (r_i, \phi_i)$ are the coordinates of the control points, as seen in figure 4.4. In this work, a 6th order (n=6) Bézier representation has been used (with 2 fixed points, P_0 and P_6, because they correspond to the leading and trailing edges of the impeller profile and 5 control points change with the variation of $d\phi_i$ (where $\phi_{overlap} < 50°$). The values of r_i are fixed and the only parameters that vary are the co-ordinates ϕ_i depending on i as:

$$\phi(r) = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} \left(\frac{r-R_1}{R_2-R_1}\right)^i \left(1 - \frac{r-R_1}{R_2-R_1}\right)^{n-i} \phi_i \tag{4.11}$$

where, $r \in [R_1, R_2]$, and ϕ_i is the sum of all $d\phi_i$ starting from i=1 because $\phi_0 = 0.0$.

A vector of $\mathbf{R}^4$ defines a chromosome; the genes are the co-ordinates of the control points. In addition, we need to use a geometrical constraint: the ϕ_i 's vary on intervals $[min_i, max_i]$ which is the search space. The parameters constituting the chromosomes are then binary encoded.

4.4.2 De Casteljau's Algorithm

The fundamental concept of de Casteljau's algorithm is choosing a point **C** in line segment **AB** such that the distance between **A** and **C** and the distance between **A** and **B** has a given ratio, say u, as seen in figure 4.8. As an example, if the procedure to find the point on the Bézier curve which corresponds to u=0.4 is described. For the curve is defined by the control polygon with the six points 0 to 5, figure 4.8. Firstly for the value of u on each line segment of the control polygon new points, 10, 11, 12, 13, and 14, are defined dividing the line segment to two parts according to the u ratio. This will continue for another n times. Where in this example n = 5, then point 50 is on the Bézier curve with the given control points.

The de Casteljau's algorithm can also be explained geometrically as follows:
By dividing the edges of the control polygon in the ratio (1-u) to u, connect the resulting points by straight lines, divide the new edges again in the same ratios, and repeat this process a total of n times, then the dividing point obtained in the last step is the point on the curve corresponding to u. Appendix B contains the full de Casteljau's algorithm.

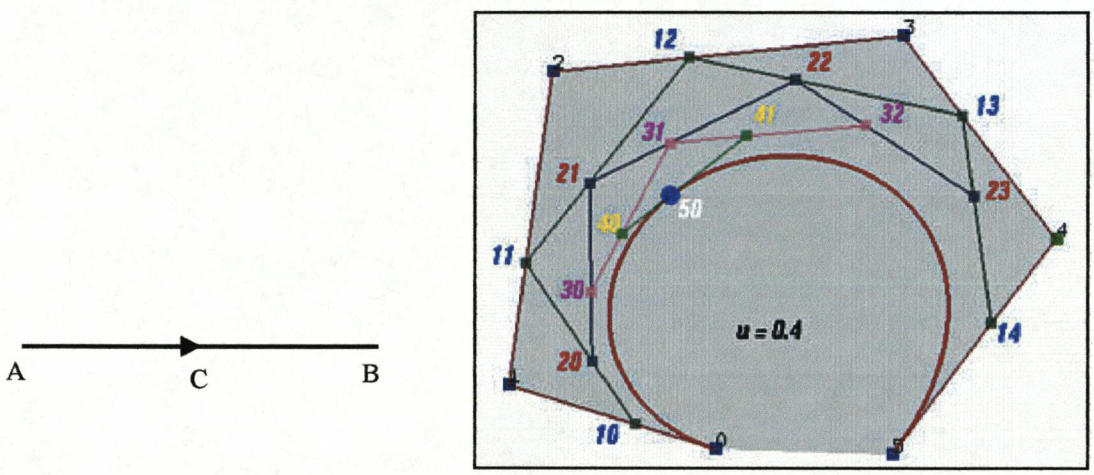


Figure 4.8 De Casteljau's algorithm to find a point on Bézier curve corresponding to the value of u.

4.5 IMPLEMENTATION OF CFD IN AUTOMATED DESIGN SYSTEM

Fluid flow treats many different settings and seems to pervade vital aspects of human life. So, it is playing an essential role in many engineering devices, such as thermal process equipment, tubomachinery, and aircrafts. Fluid flow is also important in weather systems prediction and human body itself, which relies on fluid flow for maintaining some of its basic functions such as the circulation and oxidation of blood. Altogether, the major of fluid flow processes makes the comprehension and prediction of these important, and to that end CFD has emerged as a rewarding approach.

4.5.1 CFD and Automated Design Requirements

With the objective of integrating CFD as part of an automated design process kept in mind, the choice of solver has especially highlighted the following qualities:

- ***Computational efficiency***

  Regardless of applied methodology, a predictable part of any optimisation process is the substantial number of analyses, which are required to explore the design space with the algorithm, so that it may come up with an optimum solution.

- ***Robustness***

  The automated design process sets high demands to the robustness of the CFD-code, because numerical difficulties may arise from the analysis of extreme geometrical variants. Especially, turbomachinery has a very complicated geometry.

- ***Interface capabilities***

  Both data-input and output formats of the applied solver should be clear and easy to follow.

- ***Code transparency***

  The implementation of semi-analytically derived design sensitivities requires close consideration of the modelling equations and boundary conditions actually implemented for the analysis. Therefore, advantage may be drawn from code-insight, for instance by directly utilising quantities from the analysis code, which are not normally accessible. The call for code transparency more or less disqualifies commercial codes, where access to source codes is very limited or non-existent.

4.5.2 FLow Solver

In the present study, the evaluation of a fitness value to be associated to an individual requires the calculation of the flow inside the impeller passage. The objective of this section is to briefly describe the main characteristics of the underlying flow solver for the solution of the 2D N-S equations. Appendix A contains more detailed for this flow solver, and the whole program is published by Wahba, 1997. The code is quite computationally efficient from the point of view that it treats 2D N-S equations in a blade-to-blade centrifugal impeller. It has its own grid generator with which it can capture a wide range of geometry, and offers robustness to the whole process. The source code is available, so, the user can control its inputs and outputs or add some calculation if the optimiser needs (interface capability and code transparency).

The flow field simulations are performed using the finite difference, MacCormack scheme (2^{nd} order of accuracy), viscid, laminar, incompressible CFD code **Mac_LNS** (Wahba, 1997). This code has been applied to analyse a variety of centrifugal impeller profiles, (Wahba et al., 1998 a&b).

In this work the code has been modified to represent the impeller profile with a Bézier curve instead of a polynomial, as discussed in subsection 4.2.2. The code produces consistent and repeatable flow simulations in the sense that small perturbations to design variables are accurately reflected in the flow field solution especially, when it is applied with a wide range of impeller profiles.

4.5.2.1 Mathematical model

Consider the two-dimensional, incompressible N-S equations for a constant property flow without body forces or external heat addition. The continuity equation, written in the system relative to a blade row, is:

$$\nabla \bullet \mathbf{u} = 0 \tag{4.12}$$

where $\mathbf{u}$ is the relative flow velocity.

The momentum conservation law for a blade rotating with angular velocity ω can be written as follows:

$$\frac{\mathbf{du}}{dt} + 2\omega \otimes \mathbf{u} - \omega^2 r = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\mathbf{F}_f \qquad (4.13)$$

where:

$\mathbf{F}_f$ is the general friction force equal to the gradient of viscous plus turbulent shear stresses.

$2\omega \otimes \mathbf{u}$ is the Coriolis acceleration.

$\omega^2 r$ is the centrifugal acceleration.

ρ is the operating fluid density.

4.5.2.2 Mesh update procedure

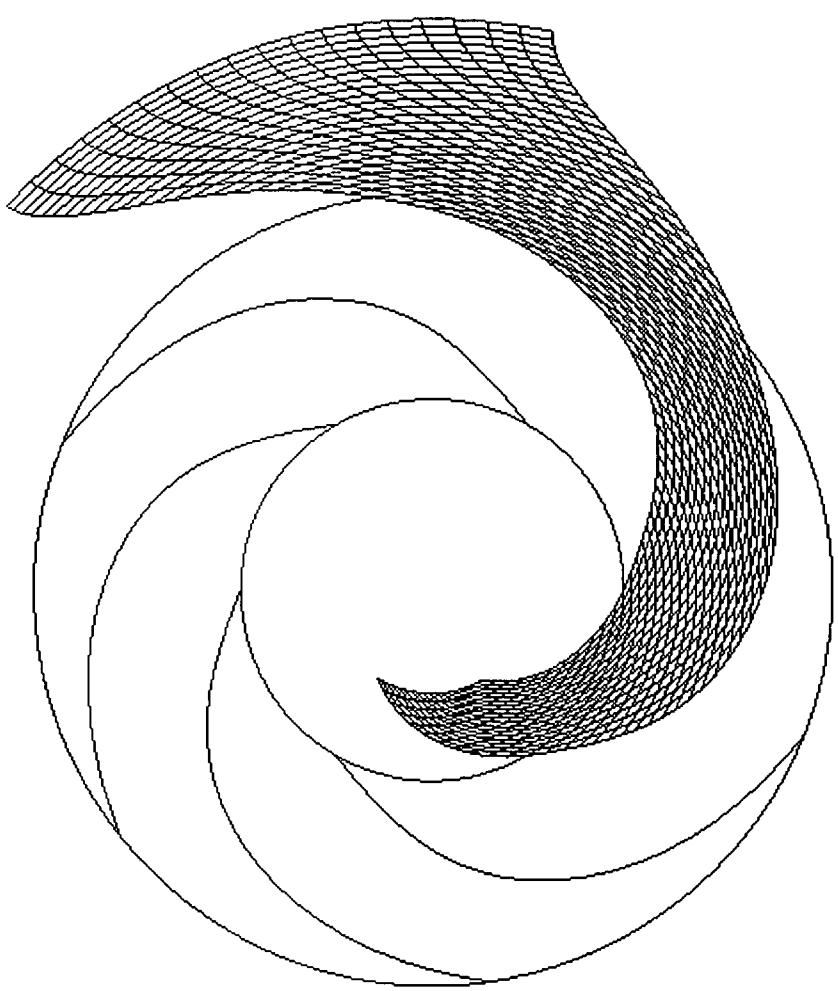


Figure 4.9, Computational grid domain.

With each individual, we have a new impeller shape. Once a new shape has been determined using the procedure described in Section 4.2.2 and 4.3, the overall computational mesh has to be updated for each new shape. In this flow solver, a structured mesh generator based on repetition of the blade curve along the blade-to-blade passage and crossed with circular curves along inlet-to-outlet is represented, figure 4.9.

4.6 CONCLUSION

This chapter investigates the use of genetic algorithms for shape optimisation. It fully describes details of how the GA optimisation procedure is linked to the CFD method to find out the optimum shape of a pump impeller profile. It is indicated that the CFD solver used is quite good to achieve the required work. It is indicated that Bézier curves is a good tool to present the impeller profile where a few control points are enough to represent the whole shape. It has been stated that traditional methods produce the same profile to all impellers, and there is no impeller profiles classification by their objectives. Some examples of the design variables and design objectives have been stated where design variables and design objectives are the main cornerstones of any design problem.

It is indicated that parallelisation is a good technique to overcome the time taken by GA and CFD, and by parallelisation this procedure can be extended to achieve design optimisation using CFD to 3D cases.

In the next chapter the GAlib, library used to carry out optimisation work, will be discussed and GA parameters will be studied. Results of these parameters will be presented and recommended to be used.

Chapter 5

GA Parameterisation

5.1 INTRODUCTION:

One of the most challenging aspects of using genetic algorithms is to choose the appropriate configuration parameter settings. Discussion of GA theory provides little guidance for proper selection of the settings. Several research papers have been published to fill this void, (Chapter 2). This chapter will present how GAlib has been configured to cover the needs of the current work. Because it is important to know why GAlib has been used in this work and how it works, a brief idea about GAlib will be presented. In addition, this chapter will contain an overview of how to implement a genetic algorithm, the programming interface for GAlib classes.

5.2 GALIB

5.2.1 Overview

GAlib is an object oriented C++ library of genetic algorithm objects, created by Matthew Wall (Wall 1996). The software and necessary documentation can be readily downloaded via ftp from ftp://lancet.mit.edu/pub/ga/. The source code is available at no cost for non-profit purposes. An extensive manual and associated set of example applications make this library extremely easy to use. The library includes tools for using genetic algorithms to carry out optimization in any C++ program using any

representation and any genetic operators. The user who uses the library works primarily with two classes: a genome and a genetic algorithm. Each genome instance represents a single solution to the user's problem. The genetic algorithm object defines how the evolution should take place. The genetic algorithm uses an objective function (defined by user) to determine how "fit" each genome is for survival. It uses the genome operators (built into the genome) and selection/replacement strategies (built into the genetic algorithm) to generate new individuals. There are three actions a user must take in order to solve a problem using a genetic algorithm:

1. Define a representation
2. Define the genetic operators
3. Define the objective function

GAlib helps with the first two items by providing many examples and pieces from which the user can build his representation and operators. In many cases the user can use the built-in representations and operators with little or no modification. The objective function is completely up to the specific problem. Once there is a representation, operators, and objective measure, the genetic algorithm can be applied to find better solutions to the problem.

When the user uses a genetic algorithm to solve an optimization problem, he must be able to represent a single solution to his problem in a single data structure. The genetic algorithm will create a population of solutions based on a sample data structure that is provided. The genetic algorithm then operates on the population to evolve the best solution. In GAlib, the sample data structure is called a GAGenome (some people refer to it as a chromosome). The library contains four types of genomes: GAListGenome, GATreeGenome, GAArrayGenome, and GABinaryStringGenome. These classes are derived from the base GAGenome class and a data structure class as indicated by their names. For example, the GAListGenome is derived from the GAList class as well as the GAGenome class. The user can use a data structure that works with his problem definition. For example, if he is trying to optimize a function that depends on 5 real numbers, then he has to use as his genome a 1-dimensional array of floats with 5 elements.

There are many different types of genetic algorithms. GAlib includes three basic types: "simple", "steady-state", and "incremental". These algorithms differ in the way that they create new individuals and replace old individuals during the course of an evolution.

GAlib provides two primary mechanisms for extending the capabilities of built-in objects. First of all (and most preferred, from a C++ point of view), the user can derive his own classes and define new member functions. If he needs to make only minor adjustments to the behavior of a GAlib class, in most cases he can define a single function and tell the existing GAlib class to use it instead of the default.

Genetic algorithms, when properly implemented, are capable of both exploration (broad search) and exploitation (local search) of the search space. The type of behavior that the user will get depends on how the operators work and on the "shape" of the search space. GAlib Features can be capsulated in these points:

- The library has been used on various DOS/Windows, Windows NT/95, MacOS, and UNIX configurations. GAlib compiles without warnings on most major compilers.
- Templates are used in some genome classes, but GAlib can be used without templates if the compiler does not understand them.
- Four random number generators are included with the library. The user can select the one most appropriate for his problem, or provide his own.
- GAlib can be used with PVM (Parallel Virtual Machine) to evolve populations and/or individuals in parallel on multiple CPUs. In this work GAlib has been modified to be used with MPI to evolve individuals in parallel on multiple processors.
- Genetic algorithm parameters can be configured from a data file, command-line, and/or code.
- Overlapping (steady-state GA) and non-overlapping (simple GA) populations are supported. The user can also specify the amount of overlap (% replacement). The distribution includes examples of other derived genetic algorithms such as a genetic algorithm with sub-populations and another that uses deterministic crowding.

- New genetic algorithms can be quickly tested by deriving from the base genetic algorithm classes in the library. In many cases the user needs only override one virtual function.
- Built-in termination methods include convergence and number-of-generations. The termination method can be customized for any existing genetic algorithm class or for new classes the user derives.
- Speciation can be done with either DeJong-style crowding (using a replacement strategy) or Goldberg-style sharing (using fitness scaling).
- Elitism is optional for non-overlapping genetic algorithms.
- Built-in replacement strategies (for overlapping populations) include replace parent, replace random, or replace worst. The replacement operator can be customized.
- Built-in selection methods include rank, roulette wheel, tournament, stochastic remainder sampling, stochastic uniform sampling, and deterministic sampling. The selection operator can be customized.
- "on-line" and "off-line" statistics are recorded as well as max, min, mean, standard deviation, and diversity. The user can specify which statistics should be recorded and how often they should be written to a file.
- Chromosomes can be built from any C++ data type. User can use the types built-in to the library (bit-string, array, list, tree) or derive a chromosome based on his own objects.
- Built-in chromosome types include real number arrays, list, tree, 1D, 2D, and 3D arrays, 1D, 2D, and 3D binary strings. The binary strings, strings, and arrays can be of variable length. The lists and trees can contain any object in their nodes. The array can contain any object in each element.
- All chromosome initialization, mutation, crossover, and comparison methods can be customized.
- Built-in initialization operators include uniform random, order-based random, and initialize-to-zero.
- Built-in mutation operators include random flip, random swap, Gaussian, destructive, swap subtree, swap node.
- Built-in crossover operators include partial match, ordered, cycle, single point, two point, even, odd, uniform, node- and subtree-single point.

- Objective function, which can be population- or individual-based.

5.2.2 Genetic Algorithm classes

GAGeneticAlgorithm class is the base genetic algorithm class which keeps track of evolution statistics such as number of mutations, number of crossovers, number of evaluations, best/mean/worst in each generation, and initial/current population statistics. It also defines the terminator, a member function that specifies the stopping criterion for the algorithm, where user can set or get the convergence percentage. The convergence is defined as the ratio of the N^{th} previous best-of-generation score to the current best-of-generation score. N is defined by the nConvergence member function. This class is automatically declared when any genetic algorithm is declared, and assigns default operators for it.

The user can maximize or minimize by calling the appropriate member function. Statistics can be written to a file for each generation or periodically by specifying a flush frequency. Generation' scores can be recorded for each generation or less frequently by specifying a score frequency. Parameters such as generations-to-completion, crossover probability and mutation probability can be set by member functions, command-line, or from file. The evolve member function first calls *initialize* then calls *the step member function* until the *done member function* returns gaTrue. It calls the flushScores member as needed when the evolution is complete. If the user evolves the genetic algorithm without using the evolve-member function, he has to be sure to call *initialize* before stepping through the evolution. The *step member function* can be used to evolve a single generation. The *flushScores function* should be called when the evolution is finished so that any buffered scores are flushed.
As mentioned before, GAlib has many different types of genetic algorithms "DemeGA, IncrementalGA, SimpleGA and SteadyStateGA". In the next section a brief discussion will be presented for these algorithms.

GASimpleGA class is a non-overlapping populations method. This genetic algorithm is the "simple" genetic algorithm of Goldberg (1989). It uses non-overlapping populations. When user creates a simple genetic algorithm, he must specify either an individual or a population of individuals. The new genetic algorithm will clone the individual(s) that he

specifies to make its own population. The user can change most of the genetic algorithm behaviors after creation and during the course of the evolution. The simple genetic algorithm creates an initial population by cloning the individual or population user pass when he creates it. At each generation the algorithm creates an entirely new population of individuals by selecting from the previous population then mating to produce the new offspring for the new population. This process continues until the stopping criteria are met (determined by the terminator). Elitism is optional. By default, elitism is on, meaning that the best individual from each generation is carried over to the next generation. The score frequency for this genetic algorithm defaults to 1 (it records the best-of-generation at every generation). The default scaling is Linear, the default selection is RouletteWheel.

GASteadyStateGA class is defined as an overlapping populations method. This genetic algorithm is similar to the algorithms described by DeJong. It uses overlapping populations with a user-specifiable amount of overlap. The algorithm creates a population of individuals by cloning the genome or population that user passes when he creates it. At each generation, the algorithm creates a temporary population of individuals, adds these to the previous population, and then removes the worst individuals in order to return the population to its original size. The user can select the amount of overlap between generations by specifying the pReplacement parameter. This is the percentage of the population that should be replaced at each generation. Newly generated offspring are added to the population, then the worst individuals are destroyed (so the new offspring may or may not make it into the population, depending on whether they are better than the worst in the population). If the user specifies a replacement percentage, then that percentage of the population will be replaced for each generation. Alternatively, a number of individuals can be specified (less than the number in the population) to replace each generation. Both cannot be specified in a parameter list containing both parameters, the latter is used. The score frequency for this genetic algorithm defaults to 100 (it records the best-of-generation every 100^{th} generation). The default scaling is Linear, the default selection is RouletteWheel.

GADemeGA calss depends on parallel populations with migration. This genetic algorithm has multiple, independent populations. It creates the populations by cloning the genome or population that the user passes when he creates it. Each population evolves using a steady-state genetic algorithm, but at each generation some individuals migrate from one population to another. The migration algorithm is a deterministic stepping-stone; each population migrates a fixed number of its best individuals to its neighbor. The master population is updated each generation with best individual from each population. If the user wants to experiment with other migration methods, a new class is derived from this one and a new migration operator is defined. The evolution behavior can be changed by defining a new step method in a derived class. nMigration, nReplacement and pReplacement are its main member functions, which control the number of individuals to migrate, number of individuals to replace and percentage of the population to replace for each generation.

GAIncrementalGA class depends on overlapping populations with 1 or 2 children per generation. This genetic algorithm is similar to those based on the GENITOR model. It uses overlapping populations, but very little overlap (only one or two individuals get replaced at each generation). The default replacement scheme is WORST. A replacement function is required only if CUSTOM or CROWDING is used as the replacement scheme. A DeJong-style crowding replacement scheme can be carried out by specifying a distance function with the CROWDING option, and also, the number of children that are generated in each generation by using the nOffspring member function. Since this genetic algorithm is based on a two-parent crossover model, the number of offspring must be either 1 or 2. The default is 2. The replacement method is used to specify which type of replacement the genetic algorithm should use. The replacement strategy determines how the new children will be inserted into the population. If the user wants the new child to replace one of its parents, the Parent strategy can be used. If the user wants the child to replace a random population member, he must use the Random strategy. If he wants the child to replace the worst population member, he must use the Worst strategy. If he specifies CUSTOM or CROWDING he must also specify a replacement function with the proper signature. This function is used to pick which genome will be replaced. The first argument passed to the replacement function is the

individual that is supposed to go into the population. The second argument is the population into which the individual is supposed to go. The replacement function should return a reference to the genome that the individual should replace. If no replacement should take place, the replacement function should return a reference to the individual. The score frequency for this genetic algorithm defaults to 100 (it records the best-of-generation every 100^{th} generation). The default scaling is Linear, the default selection is RouletteWheel.

5.2.3 Scaling Scheme

The scaling object is embedded in the population object. The scaling scheme object converts the objective score of each genome to a fitness score that the genetic algorithm uses for selection. It also caches fitness information for use later on by the selection schemes. It keeps track of the fitness scores (not the objective scores) of each individual in the population. GAlib contains a number of scaling objects derived from the base class. Here are the constructors for these scaling schemes:

- NoScaling: The fitness scores are identical to the objective scores. No scaling takes place.
- LinearScaling: The fitness scores are derived from the objective scores using the linear scaling method described in Goldberg's book (1989). The user can specify the scaling coefficient. Negative objective scores are not allowed with this method. Objective scores (*obj*) are converted to fitness scores (*f*) using the relation:

$$f = a \cdot obj + b \tag{5.1}$$

  where a and b are calculated based upon the objective scores of the individuals in the population as described in Goldberg's book (1989).
- SigmaTruncationScaling: Use this scaling method if the objective scores will be negative. It scales based on the variation from the population average and truncates arbitrarily at 0. The mapping from objective to fitness score for each individual is given by:

$$f = obj - (obj\_ave - c \cdot obj\_dev) \tag{5.2}$$

  where *obj_ave* and *obj_dev* are the objective average and standard deviation, respectively.

- PowerLawScaling: Power law scaling maps objective scores to fitness scores using an exponential relationship defined as

$$f = (obj)^k \tag{5.3}$$

- Sharing: The fitness score is derived from its objective score by comparing the individual against the other individuals in the population. If there are other similar individuals then the fitness is derated. The distance function is used to specify how similar to each other two individuals are. A distance function must return a value of 0 or higher, where 0 means that the two individuals are identical (no diversity). For a given individual,

$$f = \frac{obj}{\sum_{j=0}^{n} s(d_j)} \tag{5.4}$$

$$s(d_j) = \begin{cases} 1 - \left(\frac{d_j}{\sigma}\right) & d_j < \sigma \\ 0 & d_j \geq \sigma \end{cases} \tag{5.5}$$

where:

d_j is distance between the current individual and individual j.

n is number of individuals in the population.

The default sharing object uses the triangular sharing function described in Goldberg's book. You can specify the cutoff value (sigma in Goldberg's book) using the sigma member function. The curvature of the sharing function is controlled by the alpha value. When alpha is 1.0 the sharing function is a straight line (triangular sharing). If you specify a comparator, that function will be used as the distance function for all comparisons. If you do not specify a comparator, the sharing object will use the default comparator of each genome.

Notice that the sharing scaling differs depending on whether the objective is to be maximised or minimised. If the goal is to maximise the objective score, the raw scores will be divided by the sharing factor. If the goal is to minimise the objective score, the raw scores will be multiplied by the sharing factor. If the scaling object is associated with a population that has been created independently of any genetic algorithm object,

the sharing object will use the population's order to decide whether to multiply or divide to do its scaling.

5.3 GENETIC ALGORITHM CONFIGURATION

The genetic algorithm process requires that certain parameters be set to some initial value before starting a run. These parameters can effect the efficiency of the search process in several ways. The parameters that need to be initialised are the population size for the problem (population size is the number of chromosomes in the population); the selection scheme; the percentage of population created by crossover for each generation; and the percentage of population that is mutated in each generation. In the next sub-sections, some of these parameters will be discussed, from this work point of view. Steady-state GA has been used with overlap between generations by 50%. Roulette wheel used as a selection method. And because objective score (impeller's efficiency) is between zero and one (may become negative, in case of constraints), so, sigma-truncation scaling has been used in this work where the default liner scaling does not support negative objective scores.

5.3.1 Mutation rate

Mutation rate determines the probability that a mutation will occur. Mutation is employed to give new information to the population (uncover new building blocks) and also prevents the population from becoming saturated with similar chromosomes (premature convergence). Large mutation rates increase the probability that good schemata will be destroyed, but increase population diversity. Much experimental work has been done in order to determine the best setting for the mutation rate p_m of the GA, but no clear answer to this question could be given. Even Greenwell (1995) tried to prove a new approach that assumes more than one mutation rates and changes them during the search process. He compared that technique with some fixed rates, which has been suggested by others. Some common setting are p_m=0.001 (De Jong, 1975), p_m=0.01 (Grefenstette, 1986), and $p_m \in [0.005, 0.01]$ (Schaffer et al., 1989). The optimum value depends on the role-of-mutation (exploring the search space or maintaining diversity). If mutation is the source of exploration (when, for example, there is no crossover) then the mutation rate should be set so that a reasonable neighbourhood of solutions is explored. Typically, this involves changing around one

variable in the string, thus a mutation rate of $1/L$ is commonly used, where L is the length of the individuals' genetic representation. When mutation is used for maintaining diversity then the optimal mutation rate can depend on the size n of the population, for example, $p_m \approx \frac{A}{n\sqrt{L}}$, where A is constant equals 1.75, Schaffer et al. (1989). On the other hand, Hessner and Männer (1991) published that A= 1.0. Figure 5.1, shows the results of applying different values of mutation rates where crossover probability and population size are 0.6 and 30, respectively. It can be seen that large mutation rates is a source of noisy "good schemata can be destroyed because the chromosome string is in fact inverted", vice versa, for the low mutation rates, the conversion is smoothly and no valuable change in the range of =0.01 to 0.0083.

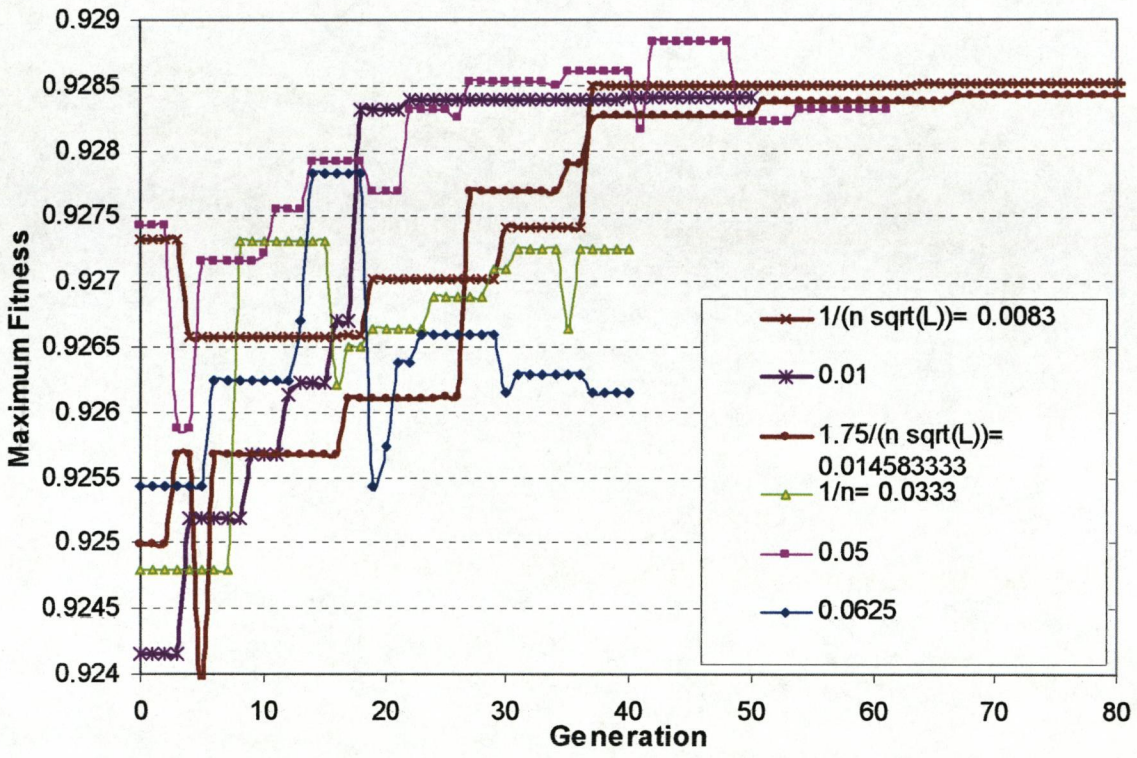


Figure 5.1, Comparison between different mutation rates.

5.3.2 Crossover probability

The crossover probability controls how often the crossover operator is applied. If there is no crossover, offspring is exact copy of parents. If there is a crossover, offspring is made from parts of parents' chromosome. If crossover probability is **100%**, then all

offspring is made by crossover. If it is **0%**, whole new generation is made from exact copies of chromosomes from old population (but this does not mean that the new generation is the same, because of mutation). Crossover is made in hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However it is good to leave some part of population survive to the next generation.

The higher the crossover rates, the more quickly new chromosomes are introduced into the population. If the crossover probability is too high, highly fit individuals are discarded faster than selection can produce improvements. On the other hand, if the crossover rate is too low, the search might stagnate for lack of exploration. Usually, crossover probability is used to be chosen between 0.5 and 1.0. Figure 5.2 represents some runs of GA with different crossover rates (0.5, 0.6, 0.66, 0.7, 0.8, 0.9, and 1.0) where population size and mutation rate are 30 and 0.0083, respectively. The result shows how crossover rate is very important to have diversity between the population.

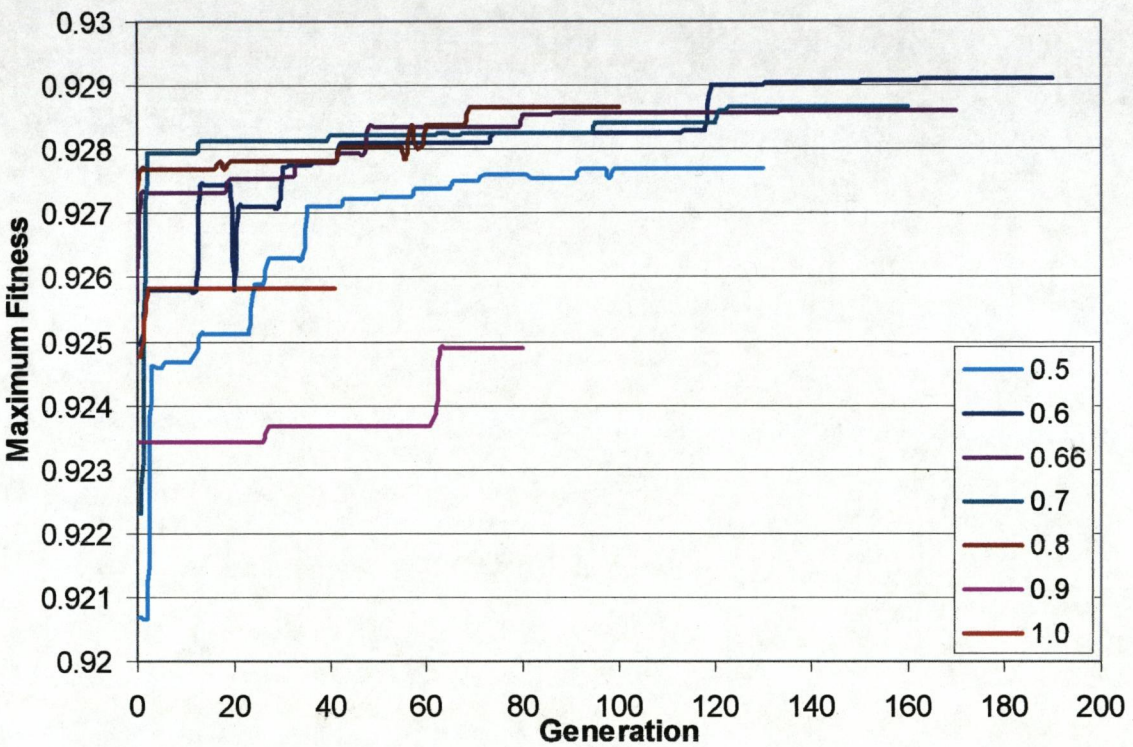


Figure 5.2, Comparison between different crossover rats.

5.3.3 Population Size

The population size, the number of chromosomes in the population, must be sufficiently large. Larger population sizes increase the amount of variation present in the initial population at the expense of requiring more fitness evaluations. The best population size is dependent and related to the length of the chromosome and problem complexity. For longer chromosomes and complex optimisation problems, larger population sizes are needed to maintain diversity (higher diversity can also be achieved through higher mutation rates and uniform crossover) and hence better exploration. Many researchers suggest population sizes between 25 and 100. On the other hand, maximum number of generations may be compensating little change in the population size. Even, it is not necessary to obtain the same results when evaluating a population size of 10 over 1000 generation and a population size of 100 evaluated over 100 generations, even though both processes would evaluate the same number of individuals. Figure 5.3, shows some results with changing the population size. From the figure we can see for 120 generation, there is an ascending order of the maximum fitness with respect to the population size (10, 20, 30, 40, and 80).

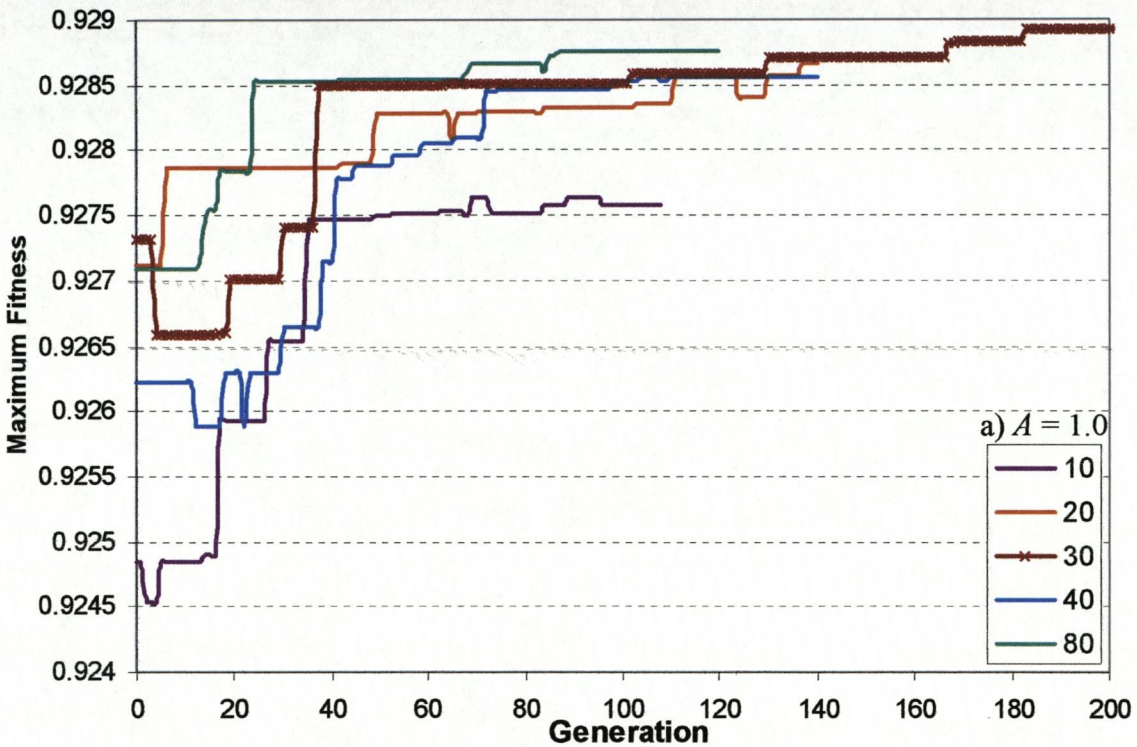


Figure 5.3a, Comparison between different population sizes

(Mutation $p_m \approx \frac{A}{n\sqrt{L}}$), A= 1.0

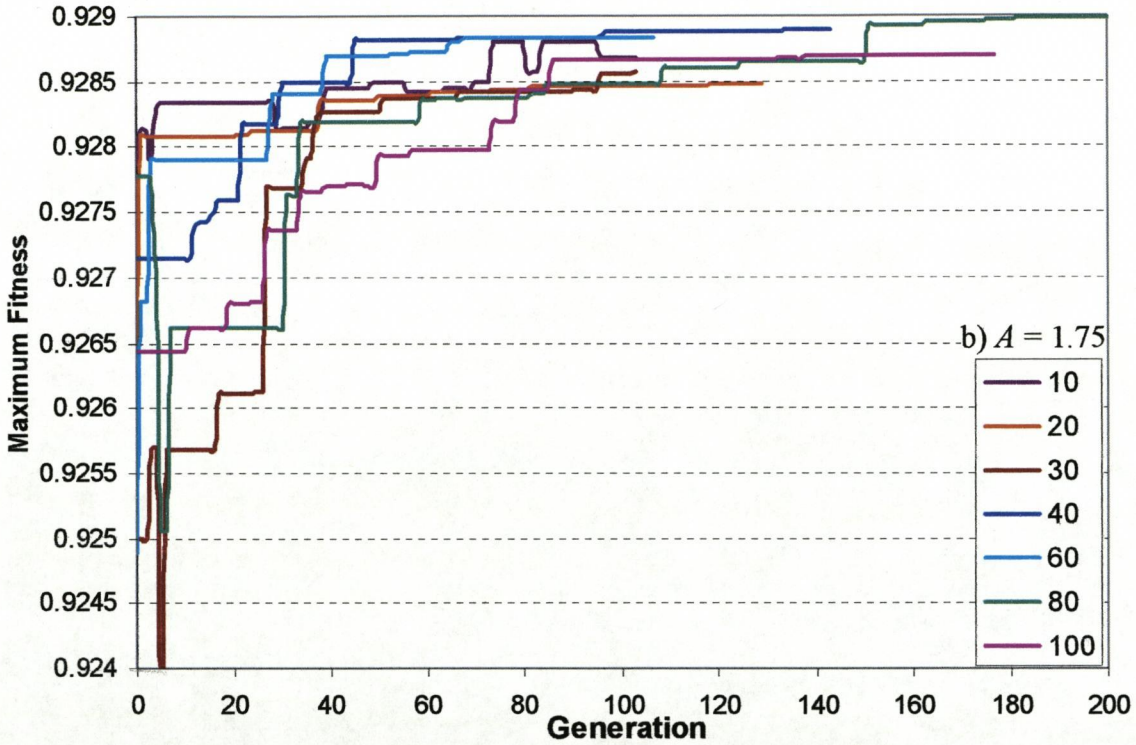


Figure 5.3b, Comparison between different population sizes

(Mutation $p_m \approx \frac{A}{n\sqrt{L}}$), A= 1.75

5.4 GAlib AND DESIGN-CASE INTERACTION

5.4.1 Technical Descriptions

The GA used in this study is a Steady State GA i.e. GA with overlapping populations. The selection scheme is a Roulette Wheel selector that selects individuals proportionally to the value of their fitness. The scaling method consistently used along this work is Sigma Truncation and the crossover is a single point crossover. The population size is 30 members. We can see final statistical report produced by the GA:

50..... # current generation.

0.996434..... # current convergence.

800..... # number of selections since initialisation.

440..... # number of crossovers since initialisation.

488..... # number of mutations since initialisation.

750..... # number of replacements since initialisation.

620..... # number of genome evaluations since initialisation.

51..... # number of population evaluations since initialisation.

1.27404..... # maximum score since initialisation.

1.10901..... # minimum score since initialisation.

For the CFD calculation, a convergence criteria of 10^{-5} has been used.

5.4.2 How GAlib is exploration and exploitation

When the aim is to convey to the human decision maker information concerning the best trade-off, consequence of data, it reduces to the visualisation of overall solutions for a single run. This can be represented in objective space by plotting the first objective component against the second, as shown in figure 5.4. It describes the relation between losses and head for an impeller, where it starts from the initialisation through some intermediate generations to the optimum max. head after 50th generation.

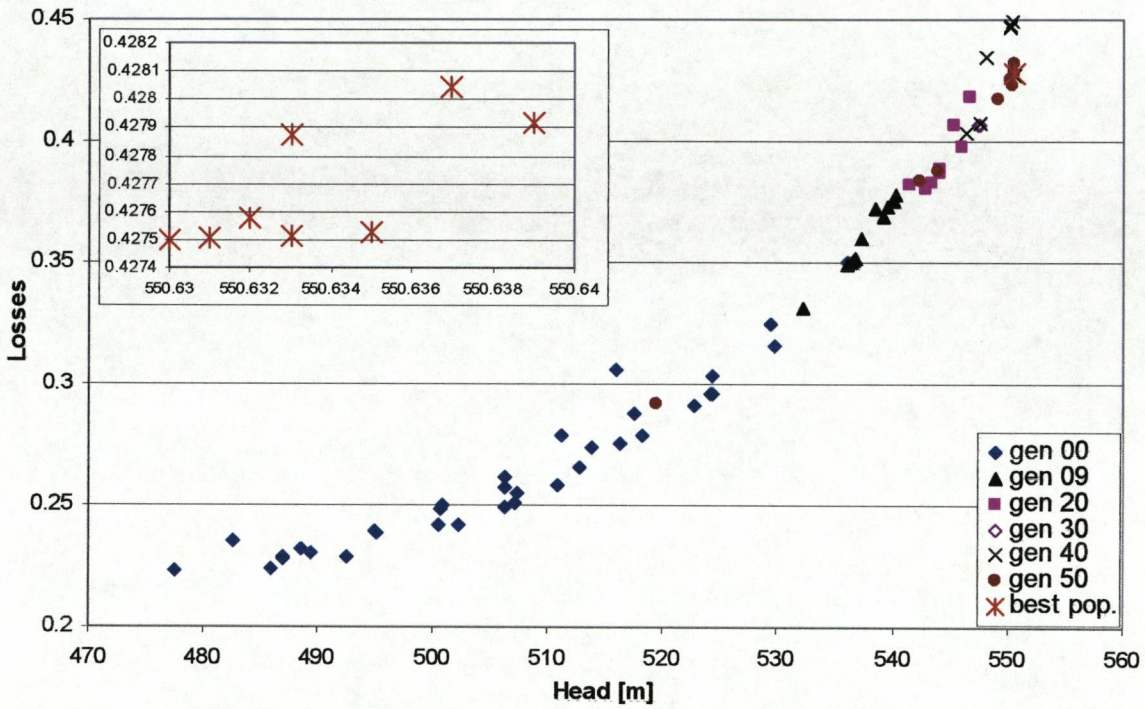


Figure 5.4, Visualisation of trade-off data in two dimensions, from single run, for one objective (max. Head).

Figure 5.5, shows the evolution of the genetic search. It reports the average fitness of the population of profiles and the fitness of the best profile for each generation. This graph shows a typical step-like shape due to the appearance of better individuals. From figures 5.4 and 5.5, we can see how GAlib is an *exploration* and *exploitation* as

mentioned in Chapter 3. In other words, it can be observed how the first generation tries to be discovery (has wide range), and how the last generation tries to give many solutions as possible.

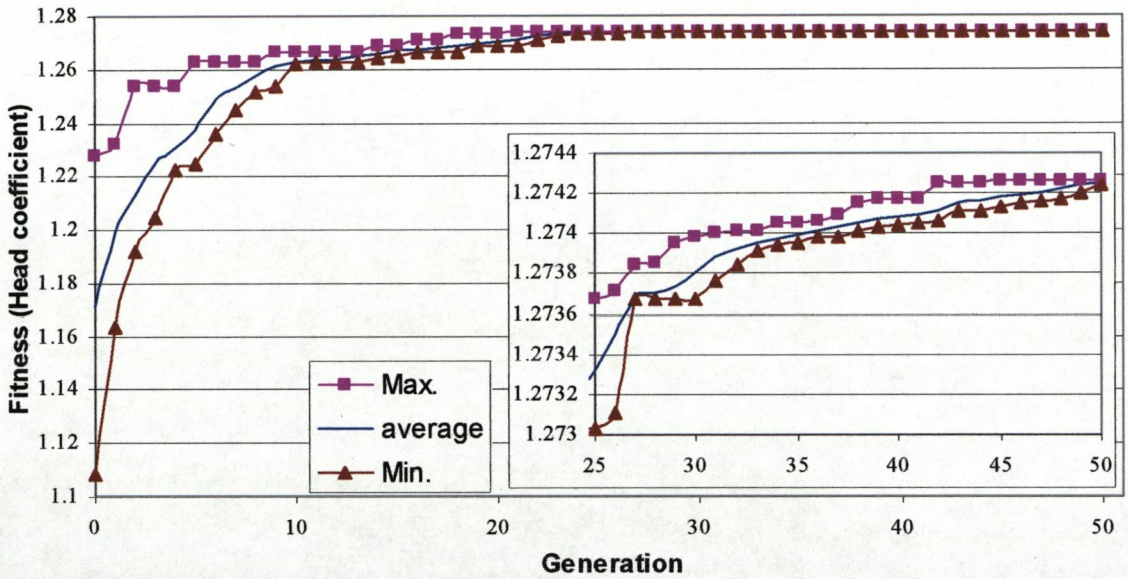


Figure 5.5, Genetic Algorithm Convergence History

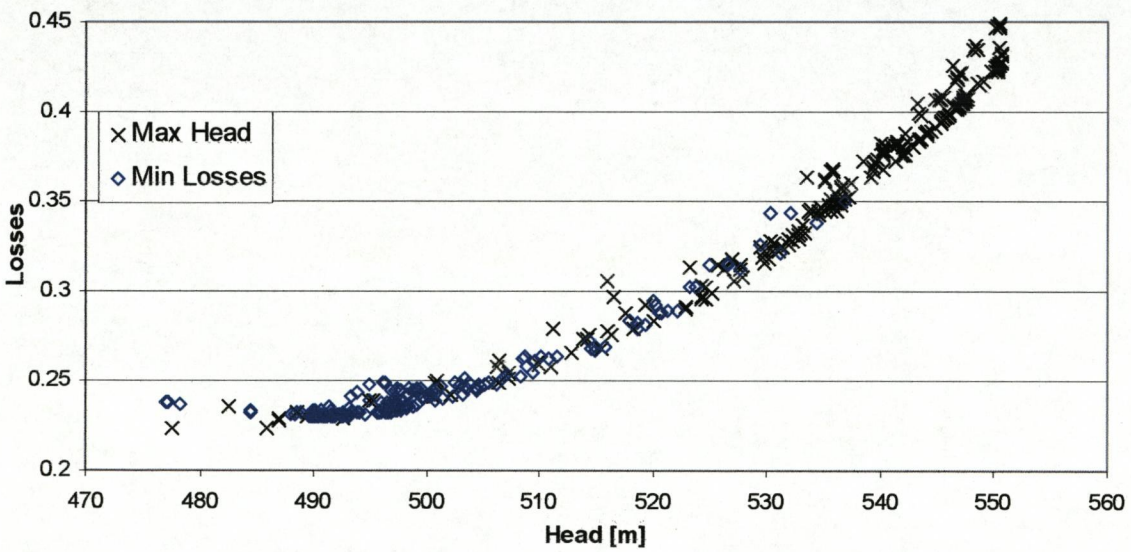


Figure 5.6, Visualisation of trade-off data in two dimensions, from multiple (two) runs, for two objectives (max. Head and Min. losses).

On the other hand, in order to gain insight into how well an optimiser can be expected to perform on a given problem, data from multiple optimisation runs must be considered. Figure 5.6 presents these runs for min. losses and max. head respectively. Human decision-maker can expect the optimum impeller profile is located with a mixture of min. losses and max. head (multi-objective optimisation).

Before going to talk about multi-objective optimisation (next Chapter), some impeller profile shapes can be presented here to know how GA works. Figure 5.7 shows how far the GA starts and how it comes close to the solution (after only one generation), which will be discussed more deeply later.

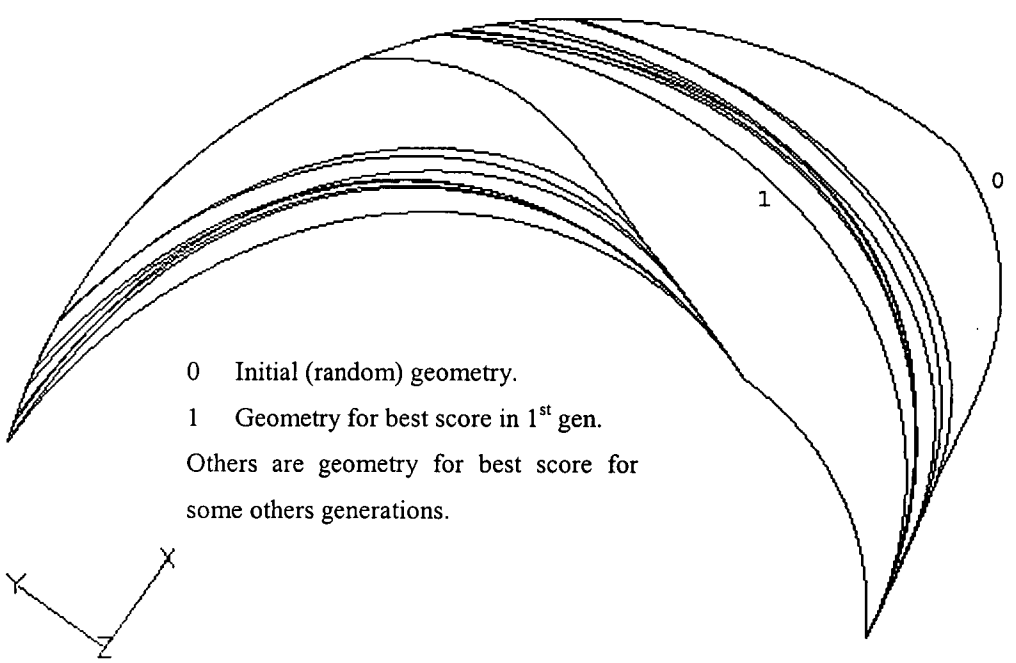


Figure 5.7, Impeller shapes for initial individual and sample of best score for number of generation.

5.5 CONCLUSION

This chapter contained a brief description to GAlib and how it can be used to support the current study.

Recommendations are often results of some empirical studies of GAs, for example:

- Crossover rate generally should be high, about 60%-80%. (However, it is recommended that around 60% is the optimum.)
- On the other hand, the mutation rate should be very low. Best rates reported are about 0.5%-1% (recommended that there is a relationship between mutation rate, population size and chromosome length).
- It may be surprising, that very big population size usually does not improve performance of GA (in meaning of speed of finding solution). For the present problem, good population size was found to be around 30-40, however sometimes sizes 50-100 are reported as best (recommended that population size is two times chromosome length, which is 16 bits in this case).

From this conclusion point of view, GAlib can be applied to optimise the design of centrifugal impellers. In the next Chapter, a detailed presentation of centrifugal impellers design results is given.

Chapter 6

Centrifugal Impeller Results and Discussion

6.1 INTRODUCTION

The rotating impeller imparts energy to the fluid, and is the most important, the only rotating element of the pump. The diffuser, following the impeller, can transform kinetic energy into pressure energy, where it is equivalent to approximately 20% to 40% of the total work input, but cannot increase the total energy of the fluid. The shape of the blades and the resulting flow pattern in the impeller determine how much energy is transferred by a given size impeller and how efficiently it operates. The theoretical energy increase, the theoretical head rise H_{th} through the impeller, can be found by applying the principle of conservation of angular momentum.

On the other hand and as mentioned in the optimisation procedure Section 4.3, "optimisation problem has two cornerstones: design variables and design objectives". The design variables will be discussed in the next subsection in order to show how these variables have been selected. The design objectives will be handled through the results, in Section 6.3.

6.1.1 Design Variable Selection

A critical element in the success of any shape optimisation method is its capability to generate a great variety of physically realistic shapes. Ideally, the geometry model should allow as much geometry flexibility as possible with as few design variables as possible. A parametric definition of the blade geometry is therefore required in order to have a limited number of variables. In order to limit the number of design variables, the blade shape must be defined by curves instead of a large number of points, as mentioned in Section 4.4; Bézier curves have been used in this current study. The blade shape is defined using 9 parameters. Four of them are the LE and TE angles (magnitude and direction), where the angle magnitudes will be discussed later in subsection 6.3.6. while the rest of the parameters lie between these four and present five difference angles, $d\phi_i$, start with i=2 to 6, where $d\phi_1$ and $d\phi_7$ are for LE and TE angles, figure 4.3. Table 6.1 presents limits of these difference angles.

Table 6.1, Design parameters limit.

$d\phi_2$ [degree]	$d\phi_3$ [degree]	$d\phi_4$ [degree]	$d\phi_5$ [degree]	$d\phi_6$ [degree]
$0 < d\phi_2 < 20$	$0 < d\phi_3 < 20$	$0 < d\phi_4 < 20$	$0 < d\phi_5 < 20$	$0 < d\phi_6 < 25$

To demonstrate the utility of the design method, a centrifugal impeller configuration, has been implemented to the optimisation procedures. An impeller, with 6 blades, has been selected for redesign. It has an inlet diameter of 35.6mm, an outlet diameter of 74mm and 23° and 17.74° as inlet and outlet blade angles, respectively, relative to the tangential direction. At 23750 rpm, the nominal flow rate is 0.0067 m^3/s.

6.1.2 Presentation of Results

In this Chapter, impeller design results will be presented and discussed. The results can be classified into three groups: parallel optimisation results, design objectives' results, and design using multi-objectives results.

In parallel optimisation results, a discussion of how it works will be presented and how many processors are selected to achieve the optimisation. For the design objectives' results, it is not easy to select objectives which can assist the designer to judge the best profile, especially because the fluid dynamics interaction of centrifugal impellers is very complicated, for example, separation, jet-wake flow pattern, etc. Consequently, some of

the design objectives' categories can be applied individually to find the most appropriate one. Also, combination of objectives will be studied. This will be discussed in the third results group "design using multi-objectives". Also, a comparison between two methods of handling multi-objectives will take place.

6.2 PARALLEL OPTIMISATION

As mentioned in subsection 4.3.4, GAs are computationally expensive especially if the estimation of the objective function is based on CFD calculations. In order to demonstrate how computationally expensive a serial GA, one objective for example "min. friction losses", is proposed using serial GA. Table 6.2 presents some important data "optimisation statistics" used to achieve this objective.

Table 6.2, Min. friction losses optimisation statistics.

Population size	30
Current generation	100
Current convergence	1.00226
Number of selections since initialisation	1600
Number of crossovers since initialisation	857
Number of mutations since initialisation	1374
Number of replacements since initialisation	1500
Number of genome evaluations since initialisation	1275
Number of population evaluations since initialisation	101
Average time to evaluate one genome for 10000 CFD iterations for computer PIII, 1000 MHz.	220 seconds

Roughly, as seen in Table 6.2, each fitness value calculation needs an average of 220 seconds for 10,000 convergence iterations, which is the average number of iterations. Sometimes 30,000 iterations are required depending on the complexity of the blade profile. On the other hand, the objective function has been executed 1,275 times for 101 population evaluations (initialisation and 100 generation as input) for 30 populations. Consequently the average time of 280,500 seconds is required to achieve this case using serial GA, neglecting the optimisation operators (selection, crossovers, mutation and replacement), individual evaluation time only. On the other hand, in case of parallel optimisation, with 15 processor, 36,507 seconds has been recorded for individual evaluation and optimisation operators. The next two steps describe how this number of processors has been selected:

- GA starts with randomly initialisation of a population of individuals, followed by individual evaluation for all 30 individuals. This occurs in three steps: 1) calculate first individual using the master processor to sign some commune variables; 2) distribute next 29 individuals to the slaves and master, and gather objective values from slaves in two times (2x15 processor). It can be noticed that the number of processors in two times are greater than the number of remained individuals, so, the last processor will update its vales and sleep during the second time; 3) apply optimisation operators using the master node.
- After the initialisation step, evaluation will continue for 100 generation assumed, for other cases it can be 50, 150 generation or specified convergence criteria, as previous. But if the number of individuals is less than or equal to the number of processors, (this is usually the case), one generation acts like one individual evaluation; if not it needs another time, which means two individual evaluations.

The main disadvantage of this method is that if the CFD calculation in one processor is huge, requiring completing its task a large number of iterations to converge, all processors have to wait until this processor completes its task. Nevertheless, parallelisation is a very good approach to perform the calculations more than seven times faster than the serial one, Wahba and Tourlidakis (2001b).

6.3 DESIGN OBJECTIVES

There are five classes of design objectives from a mathematical point of view: integral, local, min/max, max/min, and multi-criteria (for more information see subsection 4.3.2). For integral criteria, the designer can select one of these objectives: maximum impeller head, minimum losses (friction or total pressure losses), or maximum efficiency. On the other hand the local criteria can be achieved through minimum reverse flow through the impeller passage or minimum separated jet-wake flow. Two of the integral/local criteria or even a mixture of integral and local criteria (max/max, max/min, min/max or min/min) can represent the third and fourth type of criteria "min/max and max/min criteria". Also these two later types of criteria are a way to represent the fifth type of criteria called "multi-criteria".

6.3.1 Impeller Head

The impeller head represents the net work done on a unit weight of fluid in passing from inlet (LE) to outlet (TE). Hence, maximum impeller head can be assumed as an integral criterion (objective) for the impeller design. Theoretically, Impeller head is represented by a straight line equation: $H_e = k_1 + k_2 Q$ in which k_1 and k_2 are constants with the value of k_2 dependent on the value of the exit blade angle β_2. Figure 6.1 shows the Euler theoretical head-capacity characteristics for the three possible conditions on the blade angle at exit β_2 (<90°, 90°, and >90°). The effect of exit blade angle β_2 on figure 6.1 would be to change the value of $H_e = U_2^2/g$ at zero flow rate and the slopes of the lines but all head-capacity characteristics would remain straight lines. Maximum head, as an integral objective, has been applied. Figure 6.2a presents change of head, efficiency, and friction and total pressure losses with different mass flow rates.

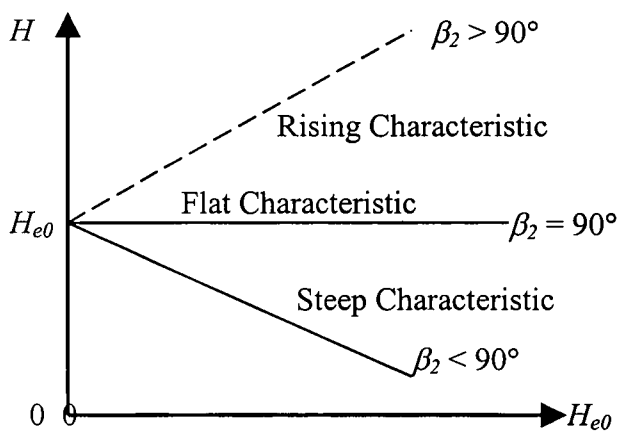


Figure 6.1, Euler's head-capacity characteristics.

6.3.2 Impeller Losses

On the other hand, impeller losses can be assumed as integral objective for the impeller design where good impeller performance needs minimising these losses. These losses are made up of impeller skin friction, and dynamic head losses. Impeller dynamic losses are entry shock loss; and hydraulic loss. Entry shock losses at the design point are small and can be neglected in this study, where optimisation procedure takes place at design conditions. In this work, hydraulic losses have been defined by total pressure losses. Where less change in blade angle along the blade passage means reduced total pressure

losses and longer blade length. Impeller skin friction losses are affected by changing blade length and relative velocity. Although flow patterns in the impeller flow passages are complex, the application of pipe friction formulae offers to date the best way of correlating design parameters with the magnitude of this loss, Neumann (1991). In the application to impeller passage the friction loss can be expressed by:

$$h_f = \lambda \frac{l}{D_{HEQ}} \frac{W_\infty^2}{2g} \tag{6.1}$$

where: λ friction coefficient.

W_∞ average relative velocity.

l blade length.

D_{HEQ} equivalent hydraulic diameter.

Colebrook's formula is recommended to evaluate the friction coefficient which is a function of Reynolds number (R_e) and of the relative surface roughness coefficient ($\overline{\Delta}$)

$$\lambda = f\left(R_e, \overline{\Delta}\right) = \frac{1}{4}\left[\log_{10}\left(\frac{4.52}{R_e}\log_{10}\frac{R_e}{7} + \frac{\overline{\Delta}}{3.7}\right)\right]^2 \tag{6.2}$$

where: $R_e = \frac{W_\infty D_{HEQ}}{\nu}$; $\overline{\Delta} = \frac{\Delta}{D_{HEQ}}$ and $\Delta = 0.00005$ m.

Minimum friction losses and total pressure losses, as integral objectives, have been applied, only one each time. Figures 6.2b&c present change of head, efficiency, and friction and total pressure losses with different mass flow coefficient.

From figures 6.2 a, b & c; it can be concluded that:

- Using the first objective "maximum head" offers better head and efficiency characteristics than those provided by the other objectives "minimum friction and total pressure losses", but there are a lot of parameters which lead the designer to avoid this objective as will be seen in subsection 6.3.3.
- The second objective "minimum friction losses" has better characteristics than the one provided by the third objective "minimum total pressure losses", where these better characteristics can be attributed in reduced friction losses which has affected both the head and efficiency characteristics. This excess in friction losses in the third objective "minimum total pressure losses" is due to more blade passage length, which will be discussed in next subsection 6.3.3.

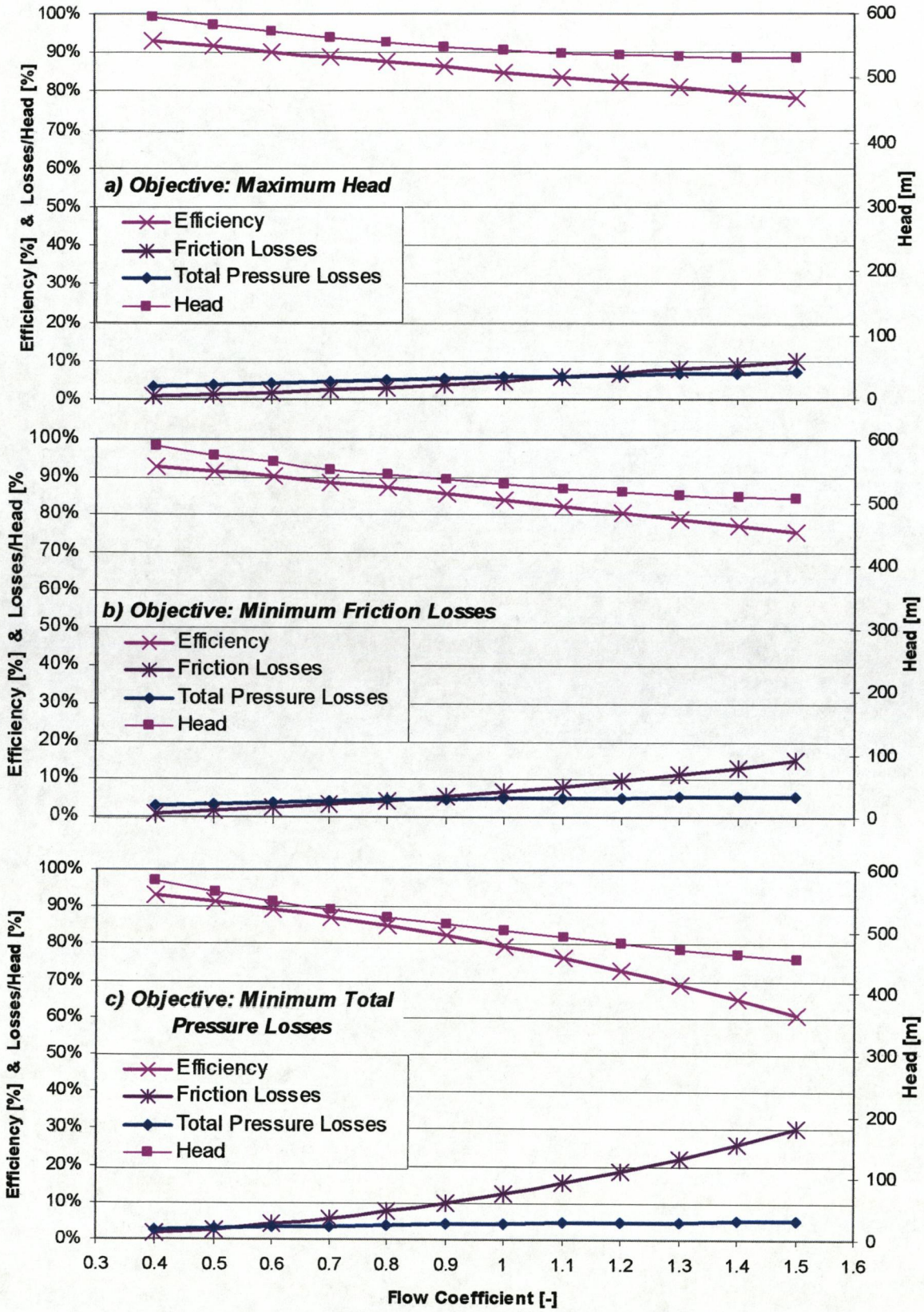


Figure 6.2, Impeller characteristics with mass flow coefficient for different integral objectives.

On the other hand, figure 6.3 presents another side of comparison between these three objective cases, which is the optimiser convergence criterion. Figures 6.3a&b show the convergence history of the two objective cases max. head and min. total pressure losses, respectively. In the case of max. head objective (a) convergence requested large number of generation if it is compared with the case of min. total pressure losses. This is because, there are many confusions which will be discussed in the multi-objective criteria, later. Otherwise, the two curves are quite reasonable form the convergence point of view.

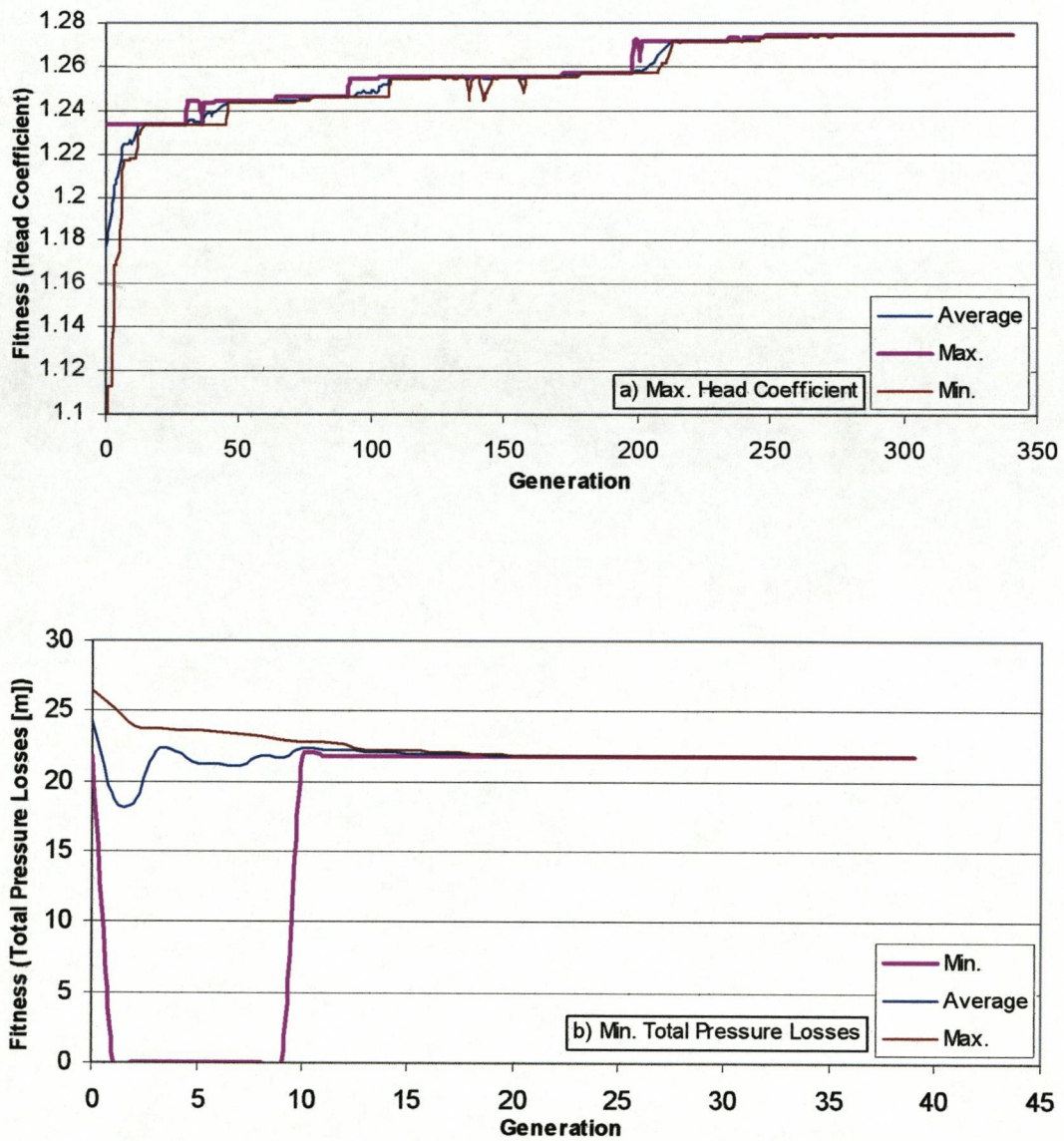
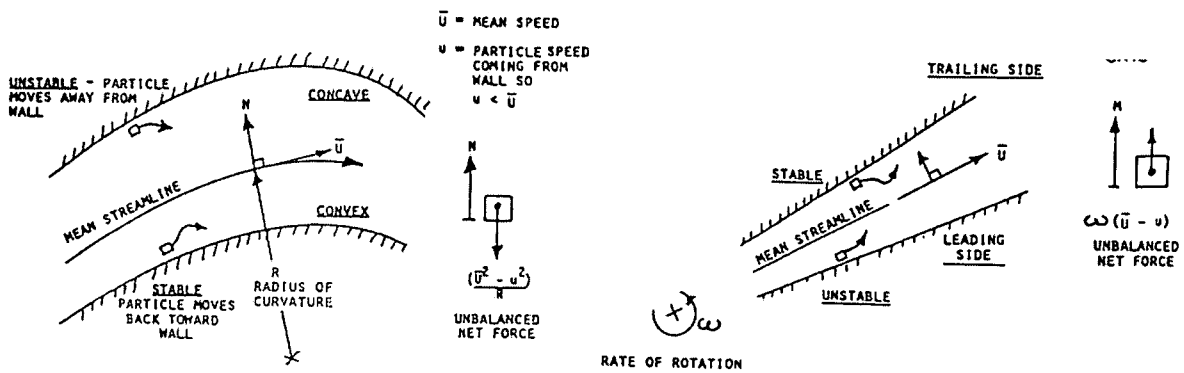


Figure 6.3, Genetic Algorithm Convergence History for single objective a) Max. Head b) Min. Total Pressure Losses.

6.3.3 Impeller Flow Separation

Flow detachment (separation) is an important term to judge between different geometric designs in centrifugal impellers. Centrifugal impeller has backswept curvature and rotational walls, and theory and experiments both show that a surface which is concave towards the flow tends to move low-speed fluid away from the wall. Conversely, the surface with a convex face toward the flow tends to resist flow detachment less than a flat surface with the same pressure gradient. The Coriolis forces, resulting from rotation, have a very similar effect, with the pressure surface resisting detachment more strongly and the suction surface resisting detachment less strongly than a flat surface with the same pressure gradient. For detailed information, see Kline and Johnston (1986). These effects are illustrated in figure 6.4.



Curvature Effect Coriolis Effects

Figure 6.4, Unbalanced net force due to curvature and Coriolis effects.

Figure 6.5 presents the relative flow velocity profile in blade-to-blade impeller for three differently designed geometries and same inlet and exit blade angles at design point. The objective functions "maximum head, minimum friction losses, and minimum total pressure losses" have been applied each time and the results are presented in a, b, and c, respectively. A quite large eddy flow area appears in the case of maximum head (a) and very smooth change of the blade angle appears in the case minimum total pressure losses (c), which means long blade profile and relatively large friction losses.

For maximum head design (a), it can be seen how the surface tends to be less backswept to give maximum head as it has been discussed in subsection 6.3.1, which means excessive separation and reverse flow. Reverse flow decays in the case of minimum

friction losses but is still detrimental. In other words, in cases a and b the blade profile becomes concave in the pressure side just after leading edge which means unstable (particle moves away from wall) for two reasons concave curvature and Coriolis effect. In the next subsection, the optimisation process will try to add some freedom to the blade shape by changing the inlet and exit blade angles. Also, in section 6.4 the optimisation will try to handle two or more objectives together in what is called multi-objective optimisation in order to improve these weak points.

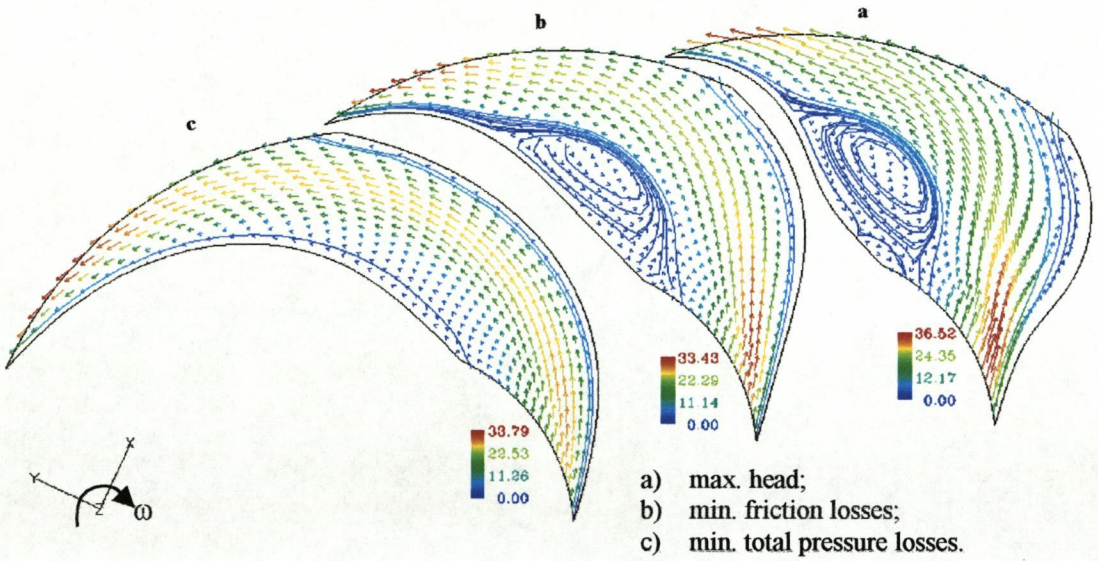


Figure 6.5, Relative flow velocity in blade-to-blade impeller profiles for three different objective functions for constant inlet and exit blade angles at the design point.

Also, from figure 6.6, it can be seen how the blade angle changes through the impeller passage and how the large change in it affects the flow through the passage. Where the flow separation dose not appear when the objectives is minimum total pressure losses where the change in the blade angle is quite small, and flow separation appears in the other two cases where there is a quite large change in the blade angle. For objective "minimum load", it looks that there is a quite small change in the blade angle which tends to be a straight line. This case will be discussed below, in the subsection 6.3.4. Friction losses and sudden change in the blade profile can also cause non-uniform flow at exit as in the case of a jet-wake flow pattern. This is apparent in figure 6.5a and starts to decay in b&c, as discussed later in subsection 6.3.6.

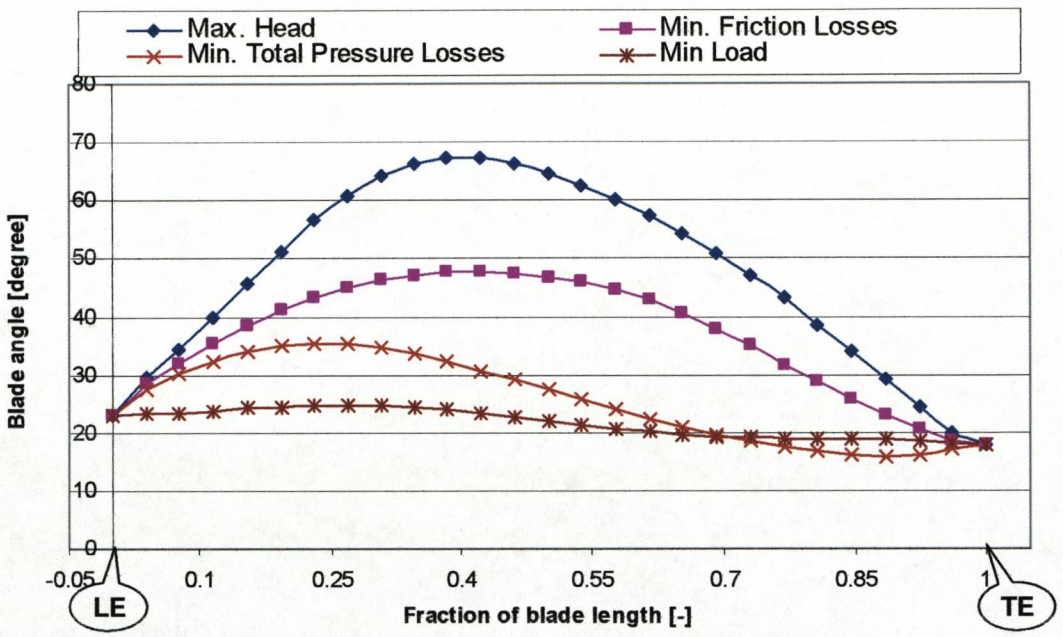


Figure 6.6, Blade angle changes through impeller passage for different objectives for constant inlet and exit blade angles at the design point.

6.3.4 Blade Loading

In figure 6.6, it appears that there is an objective "minimum load", that can change the blade angle more linearly than the objective function minimum total pressure losses. Blade loading is a very important parameter to select the proper impeller profiles. The majority of inverse design methods depend on how to find the geometry profile for a certain loading distribution. Because of the blade loading, the pressure difference across the blade, the pressure side velocity is lower than the suction side at the same fraction of the blade length. Most designers plot the relative velocity along the pressure and the suction sides of the blade, against the streamline length, measured from the leading edge. For more details see Tuzson (2000) and Japikse et al. (1997). The difference between the velocities on either side of the blade illustrates the pressure difference or blade loading, only qualitatively. Where, the rothalpy equation relates the pressure not to the velocity but to the square of the velocity, as shown in next equation:

$$P_{ps} - P_{ss} = \rho\left(\frac{W_{ss}^2 - W_{ps}^2}{2}\right) \tag{6.3}$$

From this point of view, the designer can depend on equation 6.3 as an objective to judge the blade profile. Figure 6.7 shows the relative velocity along the pressure and suction side of the blade for three objective functions which have been discussed above (max. head, min. friction and total pressure losses) and min. loading as a new integral objective. The plot shows that if the blade loading becomes too large, the velocity on the pressure side approaches zero, the flow separates, and this situation must be avoided. Also, it shows for the objective function of minimum total pressure losses and minimum loading, the loading is quite small, but for the objectives maximum head and minimum friction losses, it is quite large. In the later two objectives, the danger of flow separation is also presented because of big rapidly decreasing velocity and the pressure increasing.

In figure 6.7, relative velocity change starts to reverse its direction, in other words the pressure side relative velocity becomes higher than the suction side one, from around 60% or 75% of the blade fraction length to the impeller exit. This occurs because of the jet-wake flow pattern phenomena.

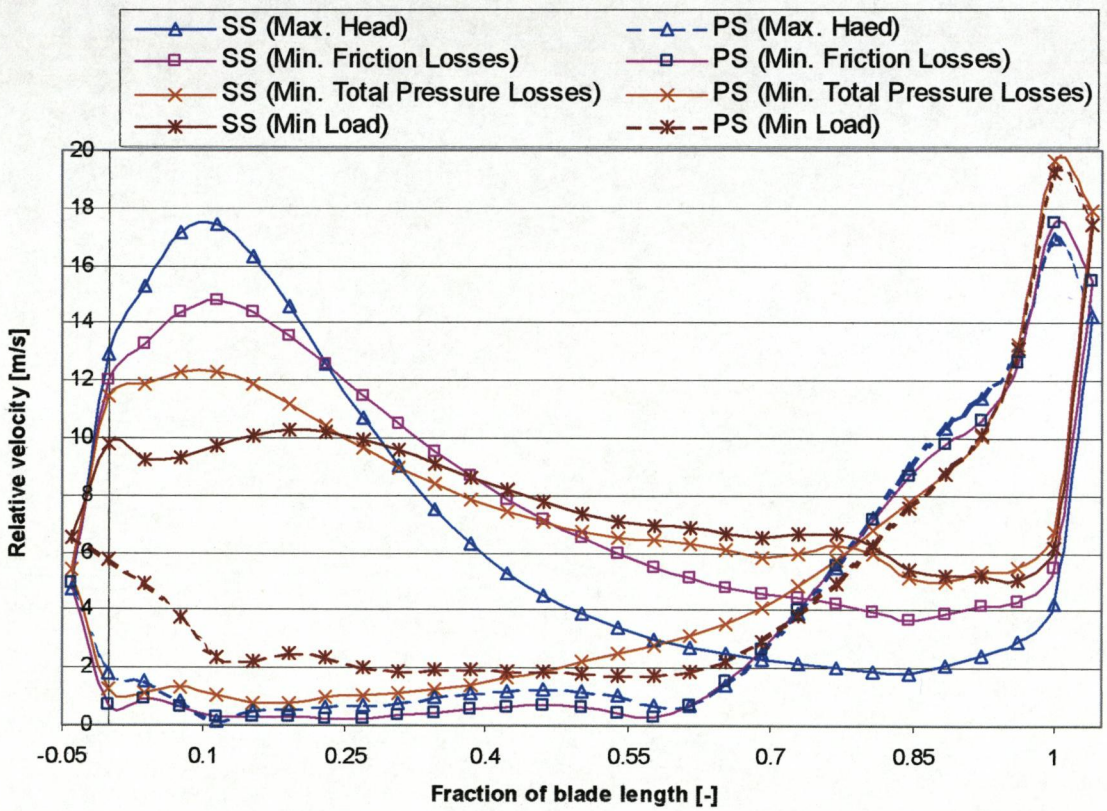


Figure 6.7, PS and SS relative velocity difference in the impeller as an approximate measure of blade loading for four different objectives for constant inlet and exit blade angles at the design condition.

6.3.5 Efficiency

The efficiency is an important performance parameter. It is a combination of two integral criteria head and losses. Figure 6.8 presents a comparison between all previous objectives in terms of efficiency and a new objective, which requires to accomplish max. efficiency. From this figure it can be observed that the max. efficiency objective behaves similarly to max. head objective. For min. total pressure losses and min. load objectives, the efficiency is reduced because of the extra frictional losses (longer blade length). In section 6.4, max. efficiency objective will be implemented for specified impeller head as a multi-objective criterion.

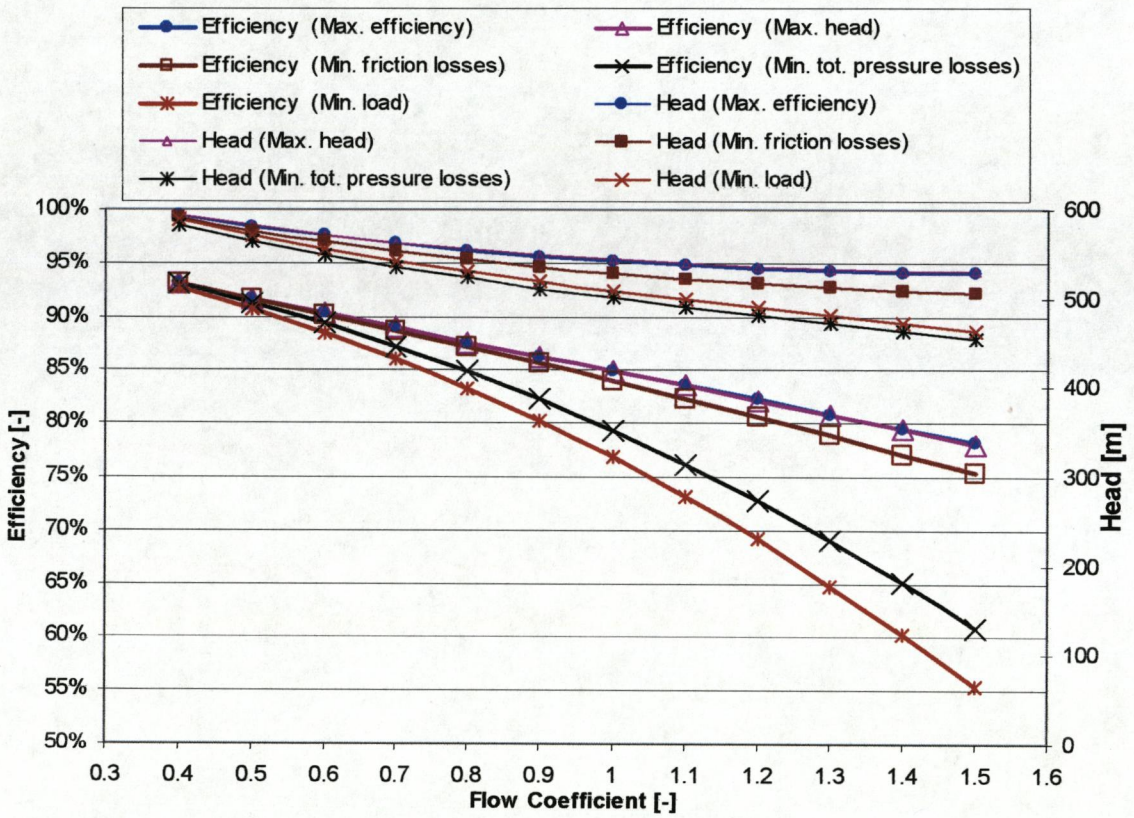


Figure 6.8, Efficiency and Head for different objectives for constant inlet and exit blade angles.

6.3.6 Local Criteria

All the previous objectives can be identified as integral criteria. In this subsection, local criteria can be tested as an objective. Local criteria are used to cover any sub properties of the flow-field or geometry. On the other hand, when separation appears in the centrifugal impeller, two flow regions of different energy can be distinguished: the separated region and the main stream, as shown in figure 6.9.

The relative velocity becomes different on either side of the streamline separating the two regions. The acceleration and corresponding pressure increase are directed from the suction side to the pressure side. A separated region of low energy will be stable and can then continue along the suction side. In case of heavy blade loading, the flow calculation may predict that the velocity slows to zero at the pressure side, as was shown above in figure 6.7 (in case of max. head and min. friction losses), and that flow separation will occur along the pressure side. However, since a separated region cannot continue on the pressure side, the flow switches to a jet-wake flow pattern. From this point of view, the designer can consider jet-wake flow pattern as a local criterion, which he/she can manage to minimise its effect. This objective "minimum jet-wake flow pattern" has been proposed in this work, where the difference between the relative flow velocity on the pressure side and suction side, respectively, has been applied to be minimum after the pressure side relative velocity becomes greater than the suction side one.

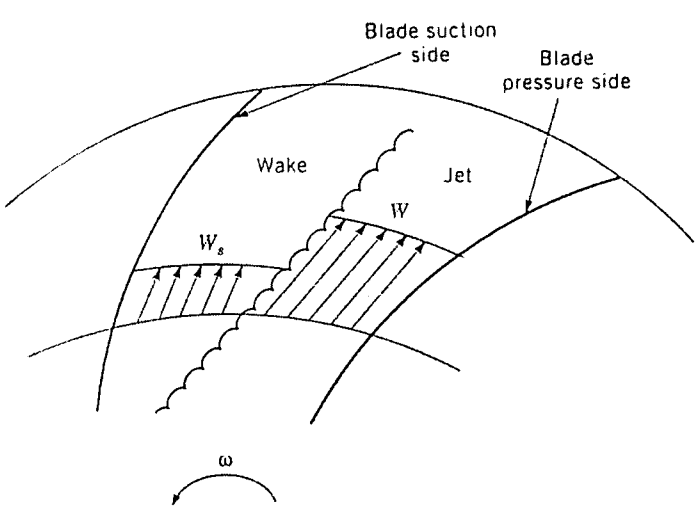


Figure 6.9, Separated jet-wake flow in impeller.

Figure 6.10 shows the pressure side and suction side relative velocity for this local criterion compared with some integral criteria for variable inlet and exit blade angles. In other words, for this time, instead of fixing inlet and exit angles (as shown before in figure 6.6) and change only Bézier's control lines in-between, inlet and exit angles have been released to change from one population to another. Hence the number of parameters has changed from five Bézier's control lines to seven. For the local criteria, minimum jet-wake flow pattern, the area between the PS and SS after 85% of the blade length looks slightly smaller than for the other cases which starts earlier (around 75% to 80% of the blade length).

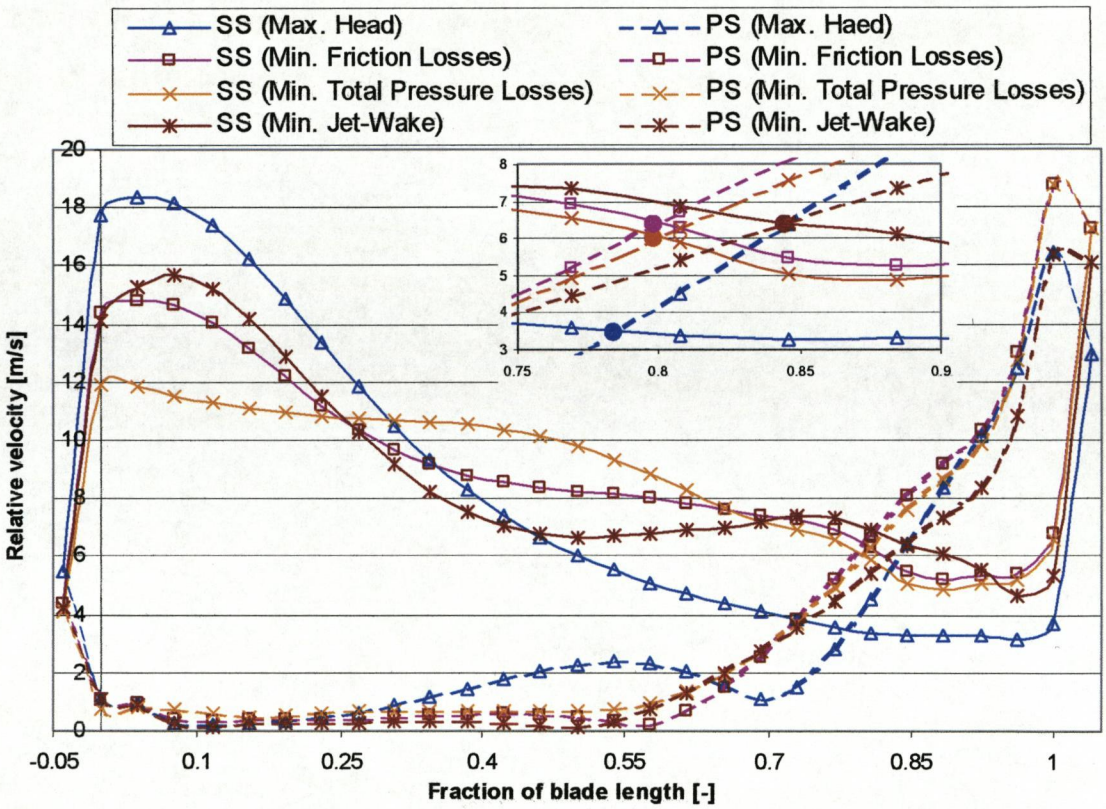


Figure 6.10, PS and SS relative velocity for three integral and one local objectives for variable inlet and exit blade angles at the design point.

Figure 6.11 shows how the blade angle changes along the impeller passage for variable inlet and exit blade angles. The change of the inlet blade angle, β_1, varies between 30° and 40° for all cases except that max. head objective (45°), tries to go towards high values (radial direction, figure 6.1). The lower and upper bounds of β_1, which were used

in max. head objective case were 20° and 45.10° respectively, as seen in Table 6.3. On the other hand, the exit blade angle, β_2, is bounded between $14.5° \leq \beta_2 \leq 26.10°$. Two objectives try to approach the upper bound and the other two the lower bound, as shown in Table 6.3 as well. Comparing figures 6.11 and 6.6, we can find there are large differences for the case of min. friction losses, where the change of the blade angle becomes distributed between inlet and exit (figure 6.11) instead of being maximum in the middle of the blade (figure 6.6). Quite small differences are observed in the others cases. This comparison will take place next in figure 6.14, using blade profiles and relative flow velocity for all flow passages obtained with different objectives.

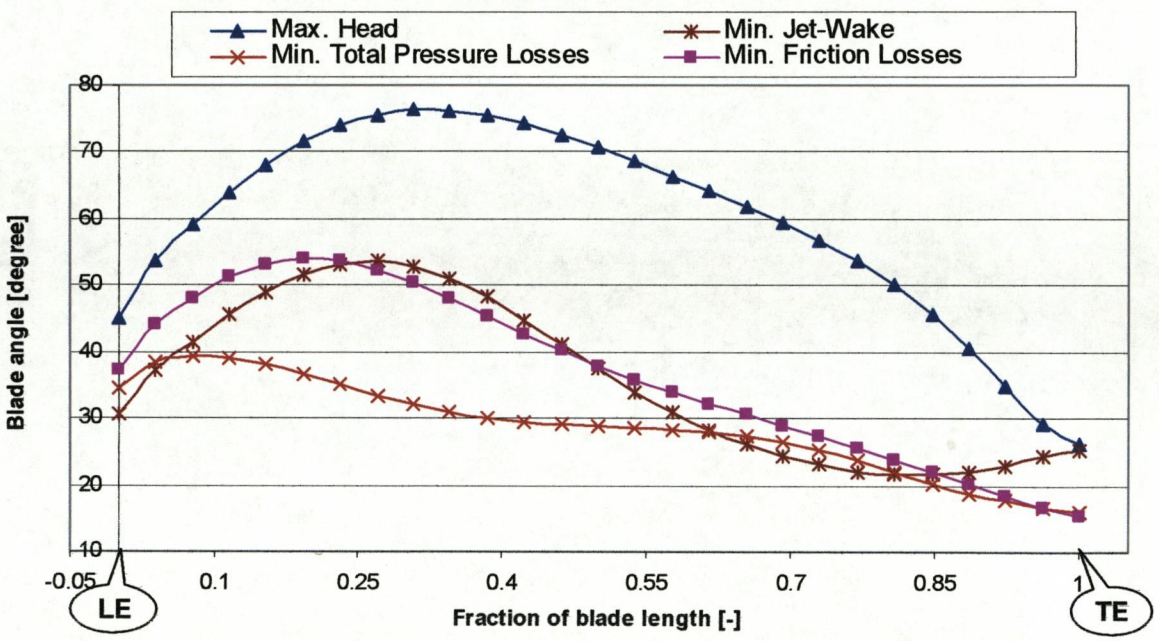


Figure 6.11, Blade angle changes through impeller passage for different objectives for variable inlet and exit blade angles at the design point.

Table 6.3, Comparison between different objectives.

Objective	β_1 & β_2 limits	β_1 [degree]	β_2 [degree]	**Head** [m]	**Friction Losses** [m]	**Tot. Pressure Losses** [m]	**Efficiency** [%]
Max. Head	$20.0 < \beta_1 < 45.1$ $14.5 < \beta_2 < 26.1$	44.90	26.09	514.39	13.42	27.17	87%
Min. friction losses	$20.0 < \beta_1 < 45.1$ $14.5 < \beta_2 < 23.1$	36.27	21.13	507.86	18.06	27.74	86%
Min. total pressure losses	$20.0 < \beta_1 < 40.1$ $14.5 < \beta_2 < 23.0$	34.40	16.09	447.35	40.63	22.62	79%
Min. jet-wake	$20.0 < \beta_1 < 40.1$ $14.5 < \beta_2 < 23.0$	38.71	17.74	470.97	42.86	22.79	81%
Min. PS relative velocity	$20.0 < \beta_1 < 40.1$ $14.5 < \beta_2 < 22.6$	43.32	24.96	483.79	28.19	25.15	84%

From figure 6.10, it can be seen that the PS relative velocity starts to rise not only after over-lapping between PS and SS relative velocity but before that (55% of the blade length) for all cases except max. head which is the earliest one. Consequently, it appears that, the last local criterion specified (minimise the difference between the relative velocity on PS and SS, after the PS relative velocity becomes greater than SS one) is not sufficient to decrease the jet-wake flow pattern. A new local objective, min. PS relative flow velocity, has been presented.

Figure 6.12 shows a comparison between these two local objectives, where quite small difference appears. But in case of min. PS relative velocity there is a quite uniform relative velocity along PS and SS than that in the min. jet-wake case. This leads to quite uniform pressure coefficient difference, as seen in figure 6.13.

A comparison of blade profile and relative velocity distribution between all different objective cases is presented in figure 6.14. Where, it can be seen that, output relative velocity (jet flow) depends on how eddy flow is big and how blade length is long.

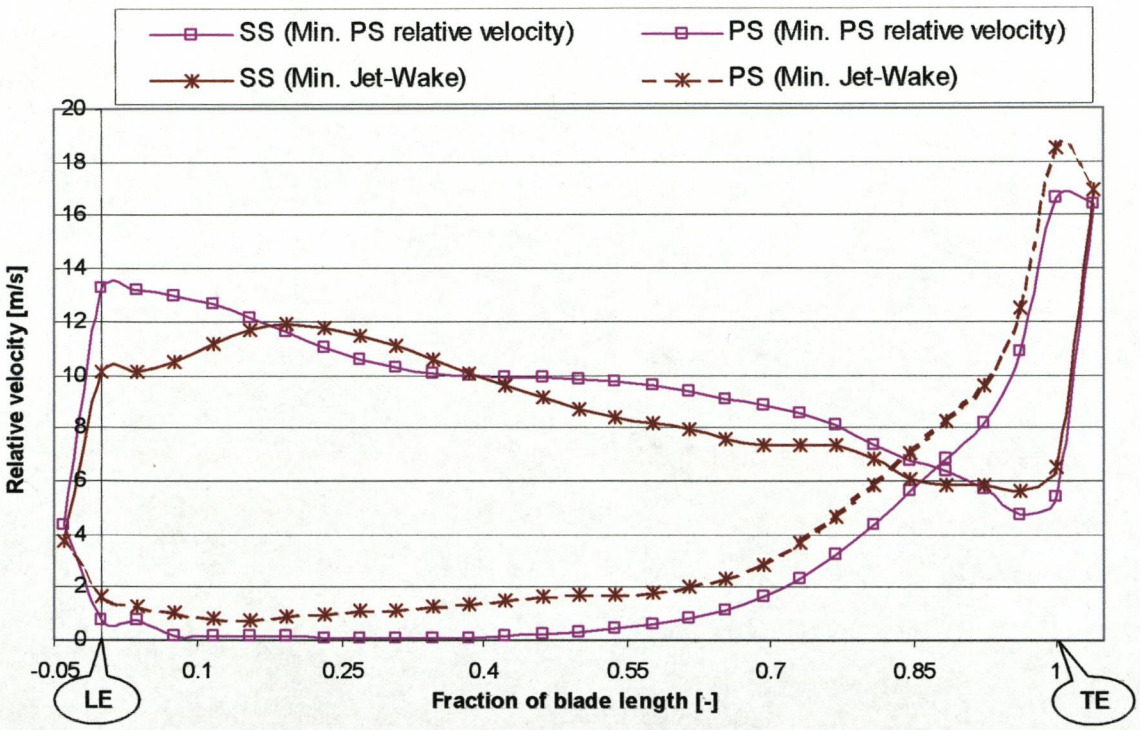


Figure 6.12, PS and SS relative velocity for two local objectives for variable inlet and exit blade angles at the design point.

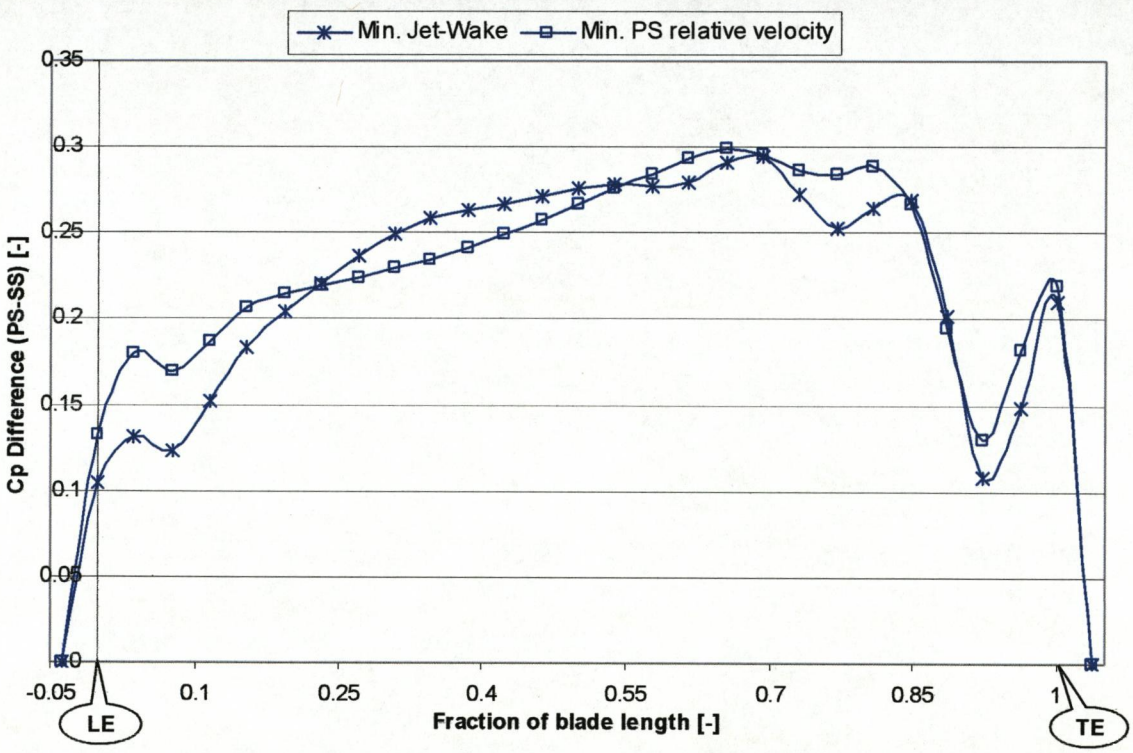


Figure 6.13, Impeller's profile loading for two local objectives for variable inlet and exit blade angles at the design point.

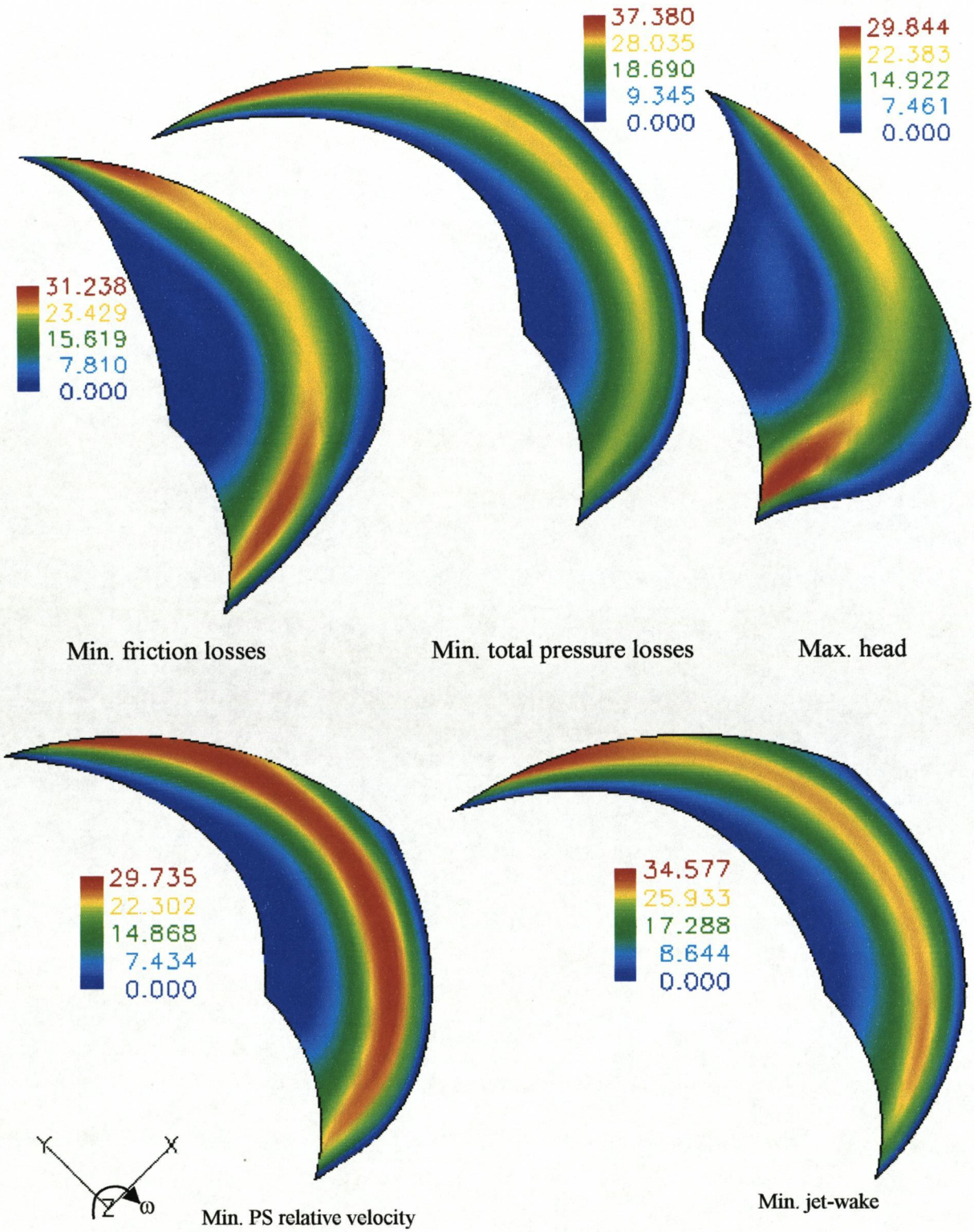


Figure 6.14, Comparison of relative velocity [m/s] distribution for different objectives and variable inlet and exit blade angles at the design point.

6.4 MULTI-OBJECTIVE DESIGN

The majority of engineering design problems requires the simultaneous optimisation of more than one objective function (multi-objectives), in which there are several conflicting design aims, which need to be simultaneously achieved. In the previous section, some of objectives have been applied individually. With multi-objective problems, every solution has a number of fitness values, one for each objective. This presents a problem in judging the overall fitness of the solutions. For example, one solution could have excellent fitness values for some objectives and poor values for other objectives, whilst another solution could have average fitness values for all of the objectives. The question arises: which of the two solutions is the fittest? This is a major problem, for if there is no clear way to compare the quality of different solutions, then there can be no clear way for the GA to allocate more off-springs to the fitter solutions. So, this section will present some objectives combinations, which can be applied to pump impeller blade profile, for example:

- Maximise impeller head and minimise total pressure losses;
- Minimise total pressure losses and minimise friction losses;
- In some cases, it is needed to fix the head and minimise the losses of the impeller.

In these and most other cases, it is unlikely that the different objectives would be optimised by the same alternative parameter choices. Hence, some trade-off between the criteria is needed to ensure a satisfactory design. Two methods are used in this study, hierarchical method and weighted sum method. Also, a comparison between these two methods will take place.

6.4.1 Max. Head and Min. Total Pressure Losses

Maximum Head does not mean minimum total pressure losses, so; designer has to find other terms or mix between head and total pressure losses (multi-objective) to design the centrifugal impeller blade profile.

The multi-objective optimisation method used here depends on the ***hierarchical*** method, subsection 3.3.2, which is carried out in four steps:

1. Rank our two objectives in order of importance (a: min. total pressure losses and b: max. head).
2. Find the optimum solution for single object (min. total pressure losses).
3. Find the optimum solution for single object (max. head) with one extra constraint:

$$\text{New Losses} \le \left(1+\frac{\varepsilon-1}{100}\right)\text{Previous Losses}, \qquad \text{for} \quad \varepsilon = 5,10,15...\% \tag{6.4}$$

   From Table 6.3, total pressure losses varies up and down within 20% between the case max. head objective and the min. total pressure losses objective case. So, ε can be less than or equal to 20%.
4. Penalty method based on feasibility technique, Deb (2000), is used to handle this extra constraint, as seen in section 3.4.2.

Figures 6.15 to 6.18 show results of this multi-objective method, and how it can find alternatives from which the decision-maker can select the proper design.

It can be noticed that, the single objective case (SO: max. head) has some oscillation in off-design (figures 6.16 and 6.17), this is for a poor blade profile as shown in figure 6.15. Also, these four figures show how to find better head even than that obtained in SO: max. head when fixing more constraints guide the optimiser.

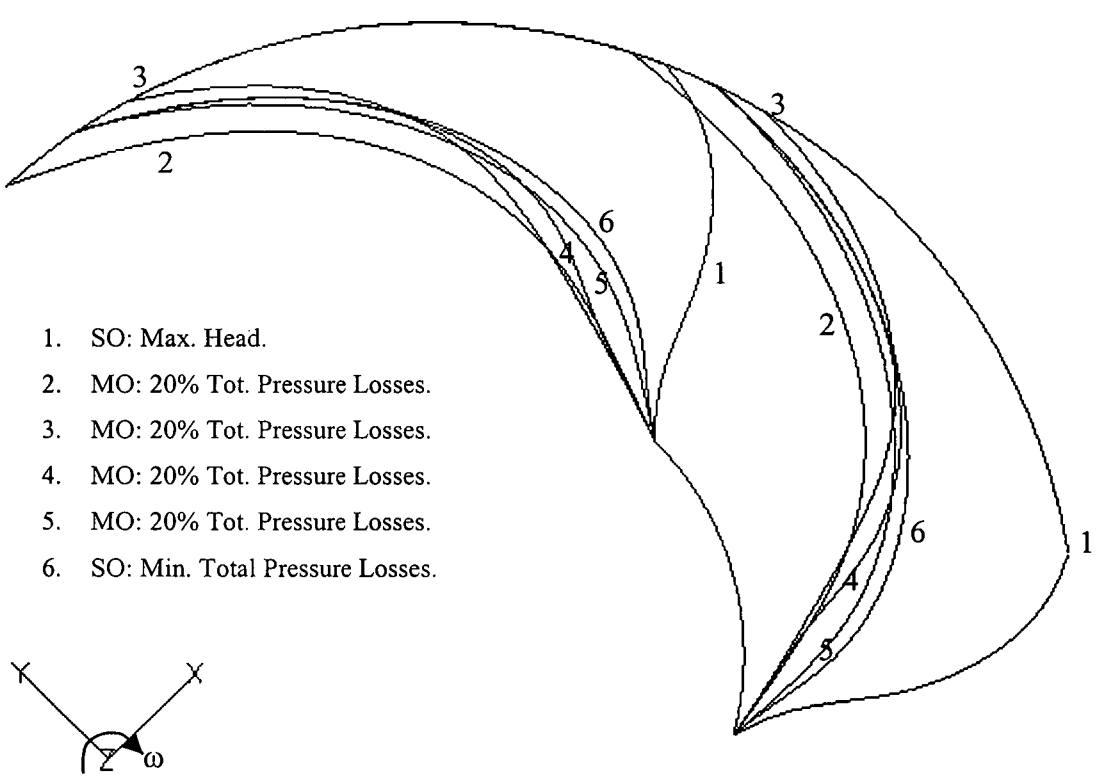


Figure 6.15, Impeller's profile geometry for these six design cases for variable inlet and exit blade angles at the design point (Multi-Objectives "hierarchical method")

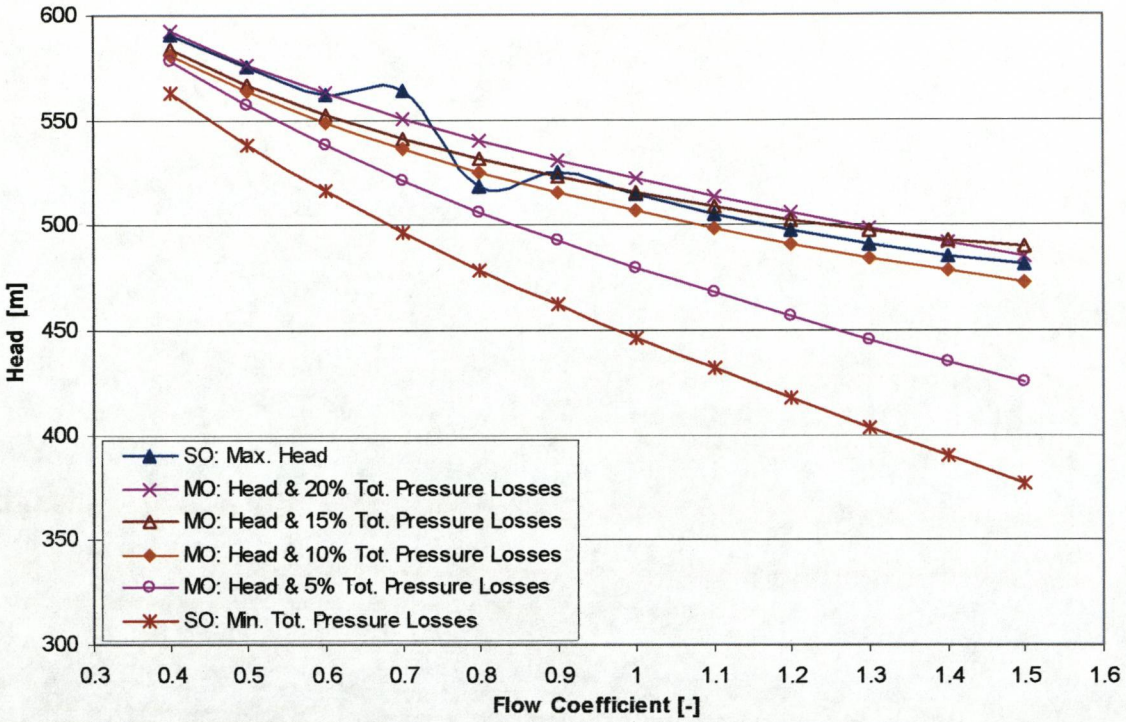


Figure 6.16, Comparison between different design Head for variable inlet and exit blade angles (Multi-Objectives "hierarchical method").

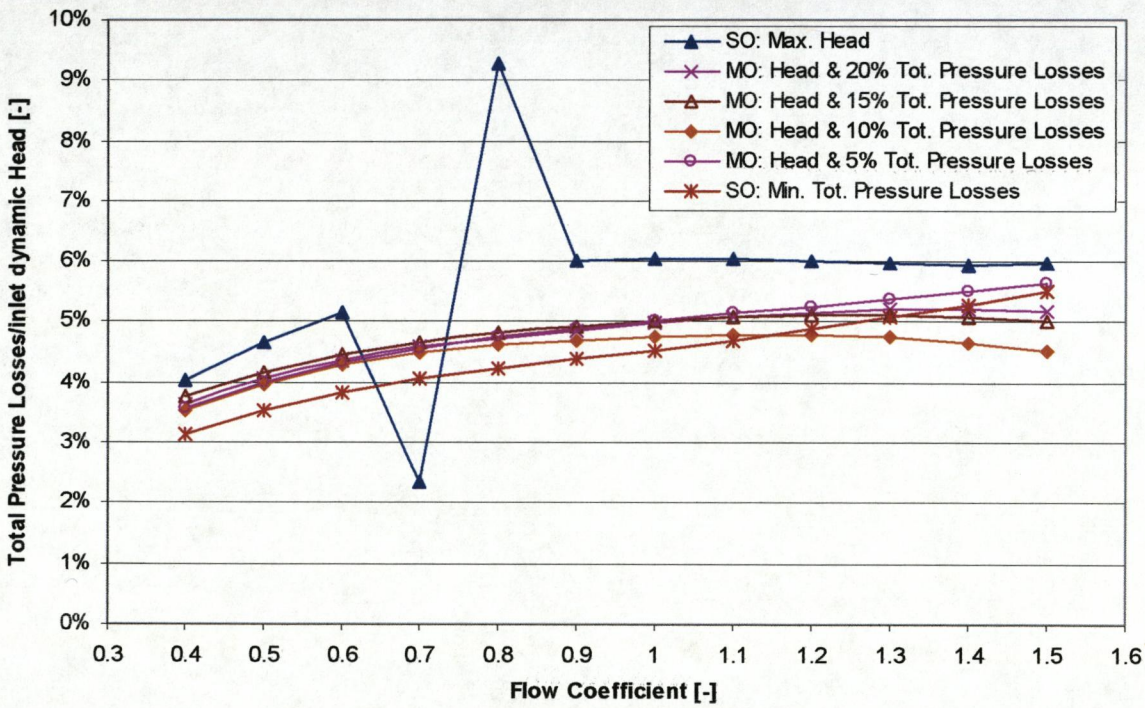


Figure 6.17, Comparison between different design total pressure Losses/dynamic head for variable inlet and exit blade angles (Multi-Objectives "hierarchical method").

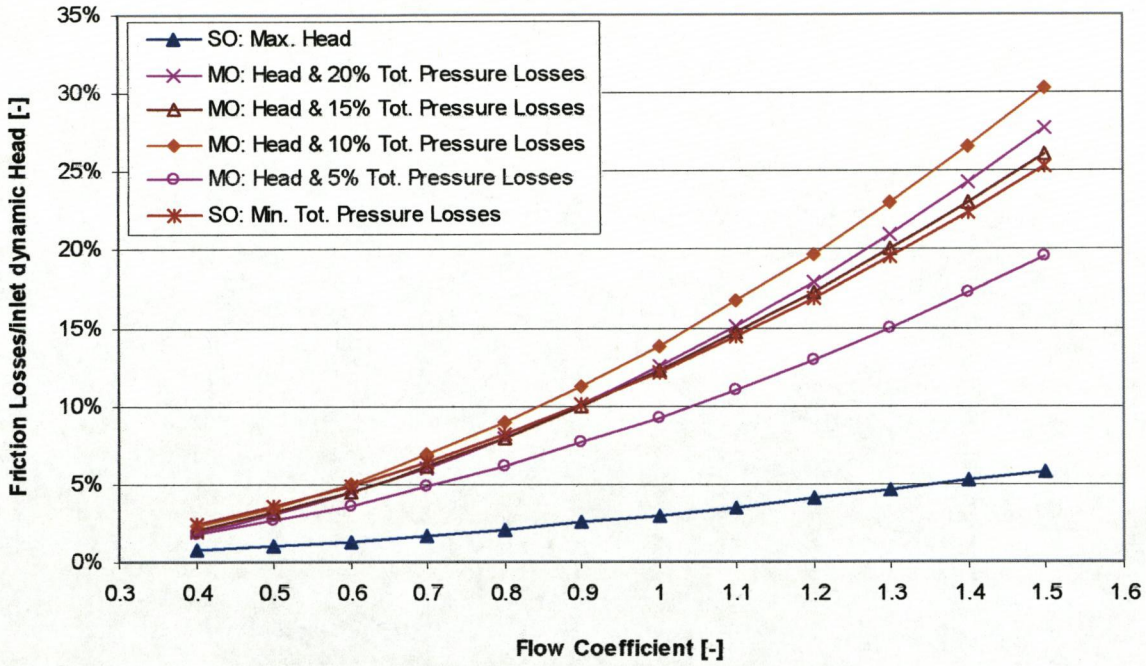


Figure 6.18, Comparison between different design friction losses/dynamic head for variable inlet and exit blade angles (Multi-Objectives "hierarchical method").

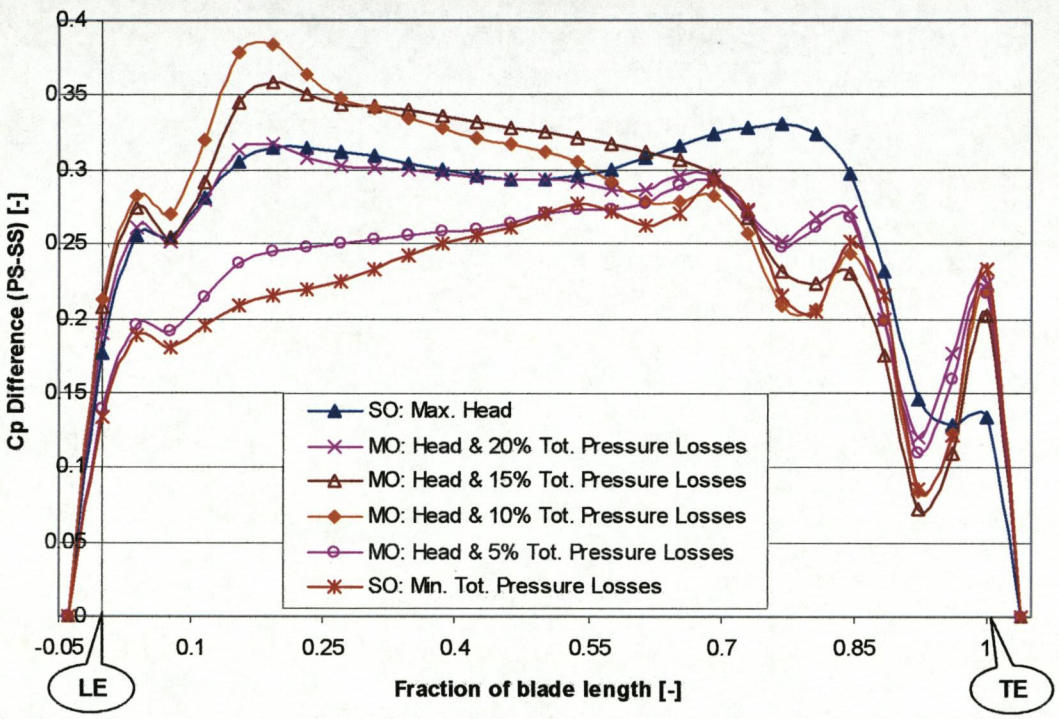


Figure 6.19, Impeller's profile loading for constant inlet and exit blade angles at the design point (Multi-Objectives "hierarchical method").

For these design cases, all multi-objective cases look to be very close to the single objective one, min. total pressure losses. For this reason, the same case is applied with constant inlet and exit angles, and this may show some differences in the loading and relative velocity distributions (figures 6.19 & 6.20), Wahba and Tourlidakis (2001a). For constant inlet and exit angles, figure 6.19 shows how loading coefficient changes within 10% from max. head design to min. total pressure losses design. Moreover, it shows how multi-objectives (mix between the two deigns) is needed to obtain better profile loading, which looks like the one obtained with min. total pressure losses, and is better than the one obtained with min. total pressure losses.

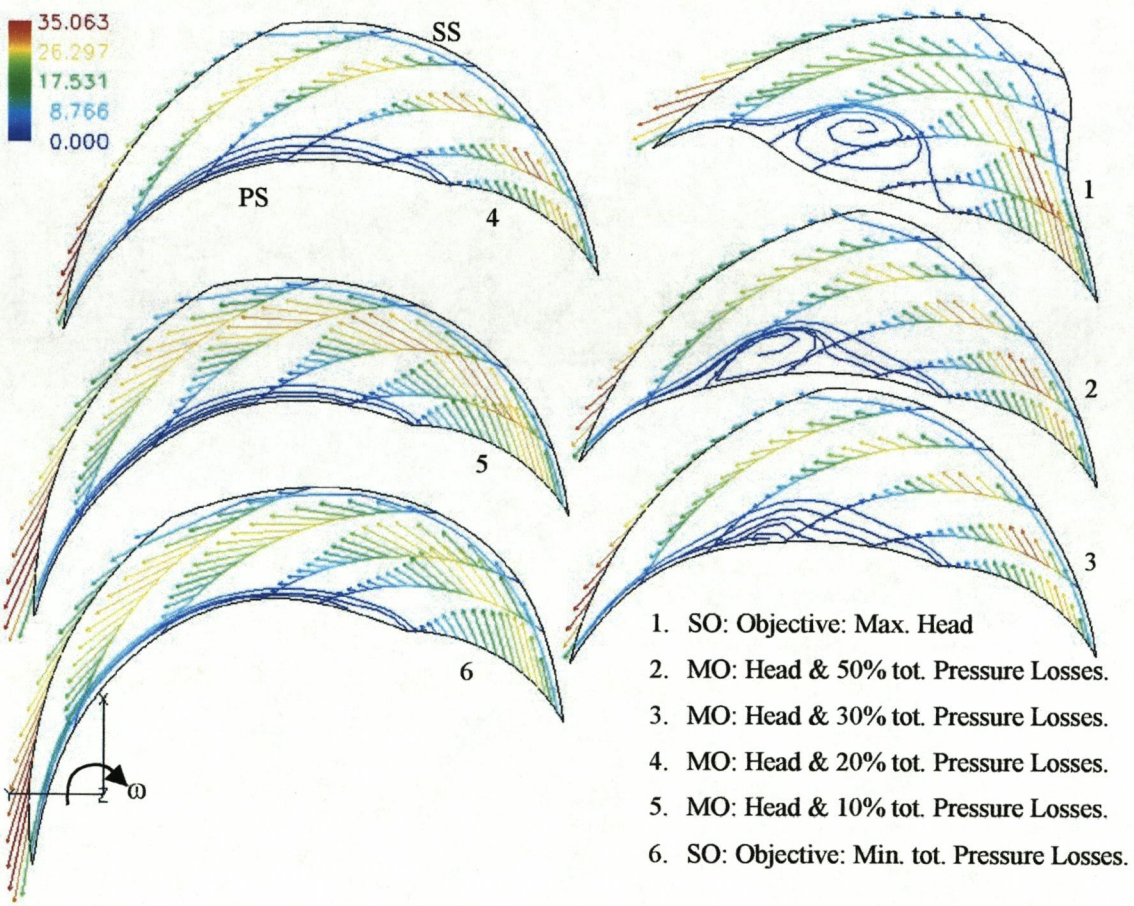


Figure 6.20, Relative flow velocity [m/s] in impeller profiles for variable inlet and exit blade angles at the design point (Multi-Objectives "hierarchical method").

Figure 6.20 presents the relative flow velocity distributions in blade-to-blade planes for six selected radii and six-designed geometry for constant inlet and exit angles at design point. One objective function (maximum head and minimum total pressure losses) is applied and represented in a and f, respectively. Back flow appears in the case of max. head and long profile length, (enhanced friction losses), appears in the min. total pressure losses case. For max. head design (1), it can be seen how the surface turns to be more radial in order to give max. head, which means separation and reverse flow. Reverse flow decays gradually when looking for less head, see (2) and (3), and disappears for (4) and (5). For min. total pressure losses (6), it can be seen how smooth is the change of blade angle, which means long blade profile and relatively large friction losses.

Figures 6.21a&b present optimisation convergence history for two multi-objective cases, which are MO: max. head subjected to 5% and 10% of the case of min. total pressure losses, respectively. Hierarchical approach using penalty method based on feasibility has been used in these two cases, where this approach considers 5% or 10% as a constraint. If this constraint is not feasible, it rejects the fitness value, by assigning the fitness value with a value less than the lowest fitness value by constraint violation value. So, in case of 5% the constrain violations still appear in the end of convergence history, which does not appears in the 10% case.

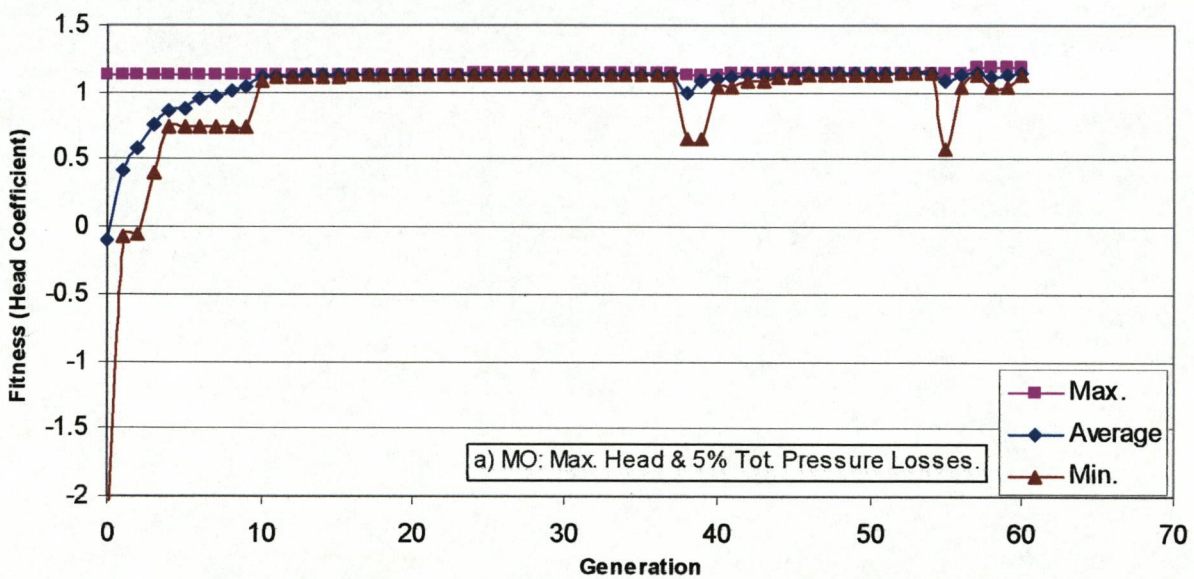


Figure 6.21, Genetic Algorith Convergence History for multi-objective a) MO: Max. Head & 5% Tot. Pressure Losses.

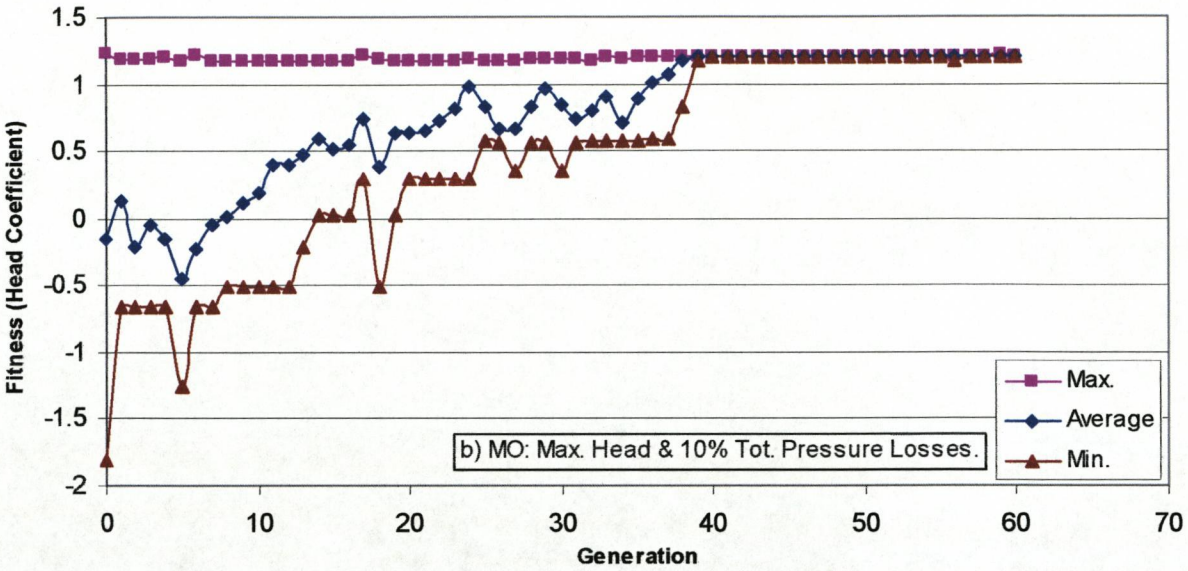


Figure 6.21, Genetic Algorith Convergence History for multi-objective b) MO: Max. Head & 10% Tot. Pressure Losses.

6.4.2 Total Pressure Losses and Minimise Friction Losses

As seen in subsection 3.3.3, the approach most users of GAs favour to the problem of multi-objective, is to weight and sum the separate fitness values in order to produce just a single fitness value for every individual. This is allowing the GA to determine which solutions are fittest as in the single objective case. In this section a design using sum of weighted objectives by applying two objectives, total pressure and friction losses (minimum losses) will take place. In other words, trying to minimise the summation of the two losses, where weighting factors, R_1 and R_2 are multiplied to each one, respectively. And as in the penalty function method, users usually have to try different values of penalty parameter, to find what value would be the best. Here also, these two weighting factors will start equal with values (0.5 & 0.5) and change slightly. This change will be in one direction only, increasing the total pressure case weighting factor and decreasing the friction losses one. Figure 6.22 presents the impeller's profile loading for this case. It can be seen that the sum of the weighted objectives method works well for multi-objective functions where some intermediate design cases have been obtained. Also, figure 6.23 shows the impeller's profile geometry for these six design cases, where it appears that the decision-maker can find a never-ending range

and discover more design profiles. All new four-design cases provided in this subsection are quite acceptable.

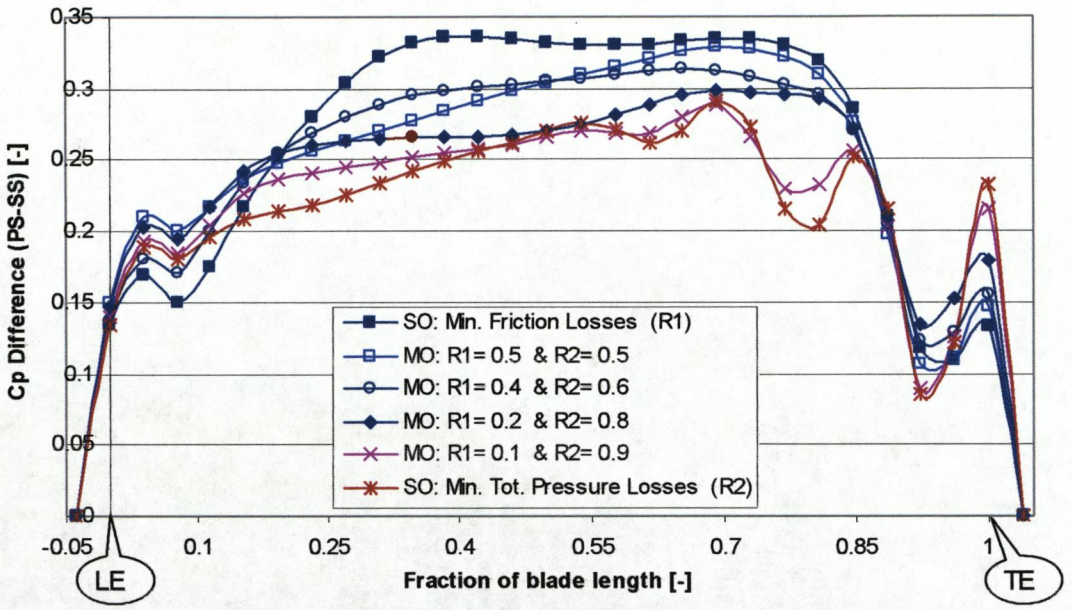


Figure 6.22, Impeller's profile loading for variable inlet and exit blade angles at the design point (Multi-Objectives "weighted sum").

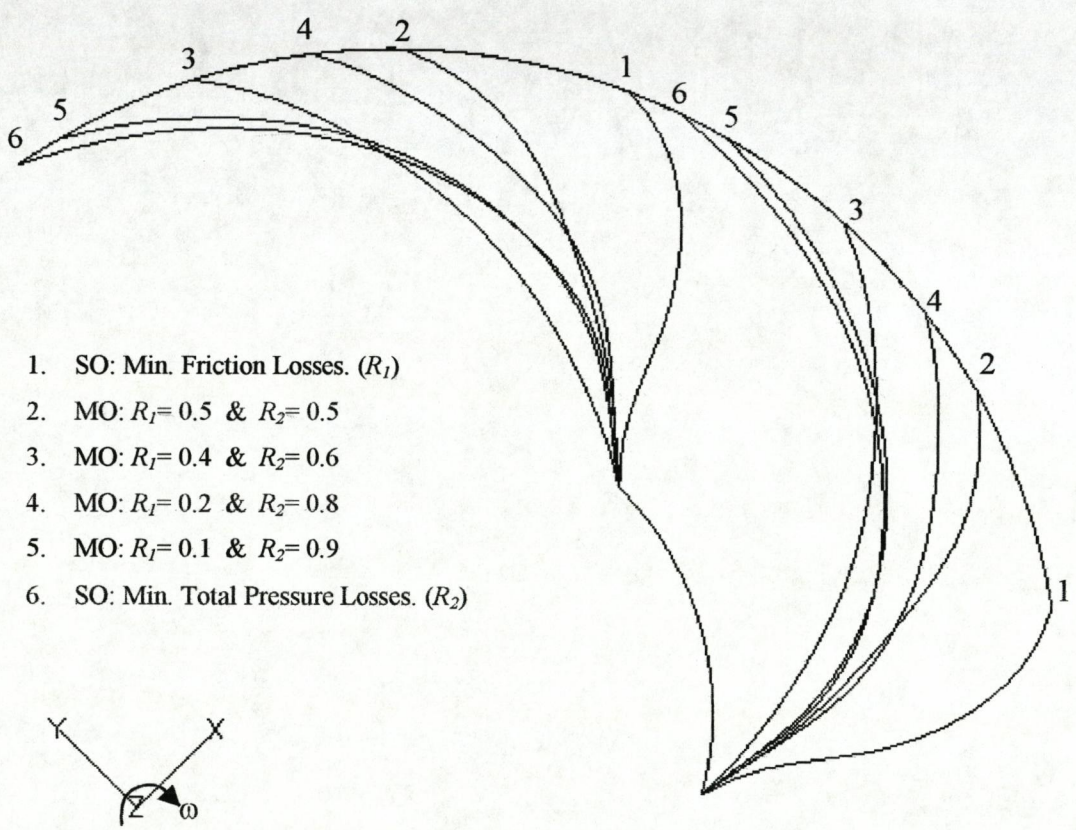


Figure 6.23, Impeller's profile geometry for these six design cases for variable inlet and exit blade angles at the design point (Multi-Objectives "weighted sum")

6.4.3 Comparison Between Multi-Objectives Handling

Table 6.4 can assist in comparing among the method of weighted objectives sum and hierarchical approach using penalty method based on feasibility. Table 6.4 presents some summarised characteristic values, which have been obtained from the cases discussed above in subsections 6.4.1 and 6.4.2. The weighted sum method, has been applied for a simple case where the two objectives are quite close to each other in value. The dependency of GA's performance on weighting factors became very small. The results, which have been obtained using this method, are quite good without a lot of try-and-error in order to find the best weight factors, as seen in Table 6.4.

Table 6.4, Comparison between different objectives at design point (Single and Multi-Objectives criteria).

Objective	β_1 [degree]	β_2 [degree]	Head [m]	Efficiency [%]	Friction Losses [m]	Tot. Pressure Losses [m]
SO: Max. Head	44.90	26.09	514.39	87%	13.42	27.17
MO: Max. Head & 50% Tot. Pressure Losses	27.75	14.83	467.29	76%	67.05	22.21
MO: Max. Head & 30% Tot. Pressure Losses	23.61	18.58	510.14	81%	51.28	22.65
MO: Max. Head & 20% Tot. Pressure Losses	21.94	22.51	522.20	81%	56.02	22.37
MO: Max. Head & 15% Tot. Pressure Losses	20.66	14.69	515.49	81%	54.96	22.55
MO: Max. Head & 10% Tot. Pressure Losses	20.36	15.58	506.83	80%	62.26	21.36
MO: Max. Head & 5% Tot. Pressure Losses	33.82	19.52	479.62	82%	41.57	22.51
SO: Min. Total pressure losses	34.40	16.09	447.35	79%	40.63	22.62
MO: 0.1 Friction Losses + 0.9 Tot. Pressure Losses	37.48	14.97	457.66	82%	43.30	20.97
MO: 0.2 Friction Losses + 0.8 Tot. Pressure Losses	40.05	22.26	493.24	85%	23.81	24.99
MO: 0.4 Friction Losses + 0.6 Tot. Pressure Losses	32.45	22.53	515.48	85%	27.56	26.19
MO: 0.5 Friction Losses + 0.5 Tot. Pressure Losses	39.09	21.21	502.04	86%	19.99	26.63
Min. friction losses	36.27	21.13	507.86	86%	18.06	27.74

The hierarchical approach using penalty method based on feasibility gave good results easily as seen before, and discovered new blades profiles with a better head than in the case of single objective (min. total pressure losses). Table 6.4 presents these results where less total pressure losses can be observed with higher head than not only the case of min. total pressure losses but also the case of max. head objectives. It can be

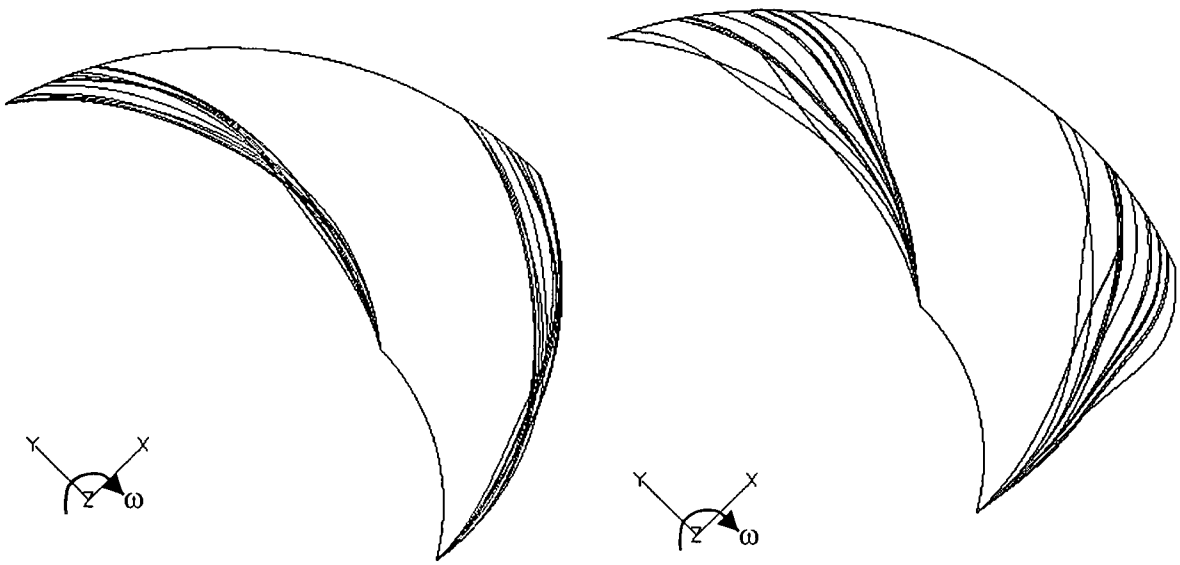
concluded that more specification means more discoveries in the proper place and good results.

This confusion between the integral objectives, SO, and multi-objective, MO, results which have been obtained is not a weak point in using GAs and it cannot be said that GAs are local optimisations algorithms, because these results are obtained after small number of generations. Also, the presented data (figures) are only for one individual from a population of solutions. More details are presented in the next subsection.

6.4.4 Specified head

Pump users asks for a certain pumping head even it is clear from above discussions, figures and tables impeller head varies up and down within 20%, where the main two factors affecting pump head are its impeller speed and size. From this point of view, more than two new multi-objective cases can be constructed where one or two of integral/local criteria can be efficiency linked with specified head. However the question still remains which value for specified head can be selected. From figures 6.15 to 6.22 and table 6.4, it can be seen that case 4 (figure 6.23, MO: 0.4 Friction Losses + 0.6 Tot. Pressure Losses) is quite good design where it provides max. head within reasonable losses and blade profile, and it is not strange if compared with those obtained by the first multi-objective case "max. head and min. total pressure losses". For this case, this head can be selected to be the specified head in the new multi-objective cases. Figure 6.24 shows one of these new multi-objective cases, where one integral objective, max. efficiency, has been applied linked with 5% and 10% of the specified head, 515m, respectively.

Figure 6.24 does not present only the final optimum profile but population of profiles, which are presented in the last generation. This clears the confusion appeared before in subsection 6.4.3, "Are GAs local optimisers?", and the answer is NO. As seen in this figure, the decision-maker has not only one optimum solution but also trade-off of optimum solutions, where they are not in only one local area. This leads to that "GAs are an *exploration* and *exploitation*" as mentioned in chapter three and proven in chapter five.



a) 5% of the specified head, b) 10% of the specified head,

Figure 6.24, Population of impeller's profiles for two Multi-Objective design cases for variable inlet and exit blade angles at the design point.

6.5 CONCLUSION

This work presented a novel way to design the pump impeller blade profile using parallel GAs and CFD. When GAs require only objective values as feedback from the solver, the use of CFD as solver is not necessarily the most cost-effective approach. Nevertheless the coupling of CFD with GAs, as it is shown by the results of the present study, leads to an extension of the power of CFD as a contributor towards a fully automated design procedure.

The chapter works step by step from how design variables are selected to how a number of objectives can be tested and how each objective behaves. Individual and multi-objectives are tested. Two different kinds of single objective, integral and local, have been used. Two methods are used to handle the multi-objective criteria, weighted sum method and hierarchical approach using penalty method based on feasibility. Comparison between these two methods has been presented.

Setting multi-objective criteria is very essential in order to find a good design, where focuses in the zone of good deign. Also, it is shown that GAs are an *exploration* and *exploitation* optimiser, as mentioned before.

Chapter 7

Conclusions and Future Work

7.1 CONCLUSIONS

The main aim of the research project documented in this thesis has been to develop and integrate methods of analysis, design, and optimisation of viscous flows governed by the Navier-Stokes equations and how to apply this work to produce a novel method for the design of centrifugal pump impeller blade profiles. The new method depends on the link between CFD and optimisation techniques. This has led to general design tools for shape optimisation. The motivation for the present work has been the growing industrial use of CFD for design purposes, accompanied by an increase in size and complexity of practical solvable CFD problems. At the same time, the general demand for rapid product development and fast response to market changes means that it becomes gradually more difficult to base the design of new and better centrifugal pump impeller on quick copies of existing pumps, or fall back on empirical design formulas established by statistical surveys of existing pumps. In the author's opinion, it is a logical step to boost the power of CFD by integration with optimisation tools for automated design procedures. And, this has been the main emphasis of this study. The main conclusions of this work can be summarised as follows:

Literature review

In chapter two, an extensive literature review has been presented. It focused on the existing pump design procedures and how these methods need to be automated. It is concluded that, CFD can and should be used for purpose of shape design in order to cut down the cycle time for a new or improved product. Consequently, the shape design methods, inverse and direct (optimisation) design, are reviewed. The evolutionary algorithms are discussed with main focus placed on genetic algorithms. Parallelisation of GAs and constraints handling methods are also discussed.

GAs

GAs are original systems based on the supposed functioning of the living. They are very different from classical optimisation algorithms. Chapter three handled the working principles of GAs and MOO. Two simple methods of dealing with MOO, are selected to be applied in the current work, weighted sum method and hierarchical approach using penalty method based on feasibility, for their efficiently and simplicity. A comparison between these two methods is presented in Chapter six. Also, parallelising methods for GAs are presented in Chapter three together with reasons for why the micro-Grain (master/slave) GA method has been selected and used in this work for its simplicity. GAlib is a GA library, which has been selected and used to present the optimisation cornerstone of this work. GAlib was modified to run as a parallel algorithm using Message Passing Interface (MPI). It is indicated that parallelisation using MPI is a good technique to overcome the time taken by GA and CFD, and quite good optimisation convergence criteria obtained by using parallelisation. GA parameters are very important to achieve the optimisation work successfully, and hence Chapter five presented how these parameters have been selected. The crossover rate recommended is relatively high. On the other hand, very low mutation rate is also recommended, where there is a relationship between mutation rate, population size and chromosome length. The performance of GA was found not to be affected by the population size.

Design procedures

Chapter four presented full details of how the GA optimisation procedures linked to CFD to find the optimum shape of a centrifugal pump impeller blade profile. The CFD solver used, Mac_LNS, is quite good to achieve the required work. Mac_LNS is based on finite difference, viscous, laminar, incompressible Navier-Stokes equations and uses the MacCormack scheme (2^{nd} order of accuracy). Also, Bézier curves are used in the current study and proved to be a quite good tool to describe the impeller profile where a few control points are enough to represent the whole shape.

The coupling of CFD with GAs, as it is shown by the results of the present study, leads to an extension of the power of CFD as a contributor towards a fully automated design procedure.

Finally, the optimum design tool has been implemented to optimise flow problems concerned with centrifugal pump impellers with respect to different types of criteria. Some integral criteria head, friction losses, total pressure losses and load are presented in Chapter six. On the other hand, local criteria and multi-objective criteria have been presented as well. There are several flow phenomena inside the centrifugal pump impeller, which were discussed, for example, losses, blade loading, jet-wake flow and interaction between flow separation and curvature of the blade and Coriolis effects. Quite good results have been obtained with lesser effort and no need to experience and experimental work.

7.2 FUTURE WORK

The work on design optimisation in turbomachinery is a recently initiated research area, which leaves many interesting possibilities for future work. First of all, there is an evident need to link further CFD and optimisation procedures, where some of CFD commercial codes already starts to add optimisation modules attached to their algorithms.

For centrifugal pump design, it is important to apply this procedure to the whole pump either using CFD or other preliminary design tools. Where, it is necessary at the outset to establish quite clearly the designers' objectives and try to go outside of the closed loop "quickly copy of existing pumps", or fall back on empirical design formulas established by statistical surveys of existing pumps. Where, the perfect hydraulic pump design is of no use unless it can be made and put into service, economically and the recent design procedures depend on try and error and experimental work, which are expensive.

For the pump flow solver, it is important to have a very good flow solver regardless of any applied methodology, which imposes two kinds of requirements:

Flow Requirements:

- Good turbulence model to simulate turbulent flow near walls, tip leakage, secondary flow, etc.
- Powerful curvilinear grid generator to capture boundary layers;
- Taking into consideration blade thickness.
- Taking into consideration cavitating flows by two-phase flow which is the real case of high-speed pumps.

Optimisation Requirements:

- Computational efficiency to handle a considerable number of analyses.
- Robustness to handle the numerical difficulties that may arise from the analysis of extreme geometrical variants (turbomachinery has a very complicated geometry).
- Good interface capabilities to be clear and easy to use.
- Code transparency to utilise quantities from the code, which the designer needs as objectives for his/her work.

References

Adby P.R. and Dempster M.A.H., 1974, *Introduction to Optimization Methods*, Chapman and Hall, London.

Adelman H.M. and Haftka R.T., 1986a, "Sensitivity Analysis in Engineering Symposium", *NASA/CP-2457.*

Adelman H.M. and Haftka R.T., 1986b, "Sensitivity Analysis of Discrete Structural Systems", *AIAA Journal 24(5),* 823-832.

Aidala P.V., Davis W.H. and Mason W.H., 1983, "Smart Aerodynamic Optimization", *AIAA 83-1863.*

Alonso J., Jameson A., et.al., 1997, "An Efficient Multi-Block Method for Aerodynamic Analysis and Design on Distributed Memory Systems," *AIAA Paper 97-1893.*

Al-Zubaidy S.N., 1995, "A Proposed Design Package for Centrifugal Impellers", *Computers & Structures*, Vol. 55, No. 2, pp. 347-356.

Anderson H.H., 1938, "Mine Pumps", *Journal of the Mining Society*, Durham.

Anderson H.H., 1955, "Modern Developments in the use of Large Single Entry Centrifugal Pumps", Proceeding of I.Mech.E., Vol. 169, No. 6.

Appa K., Argyris J., et.al., 1998, "Synergistic Aircraft Design using CFD Air Loads", *Computer Methods in Applied Mechanics and Engineering,* 166, Pages 247-259.

Bäck Th., 1996, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.

Baysal O. and Eleshaky M.E., 1991 "Aerodynamic Sensitivity Analysis Methods for the Compressible Euler Equations", *J.Fluids Engineering*, 113, 681-688.

Baysal O. and Eleshaky M.E., 1992, "Aerodynamic Design Optimization Using Sensitivity Analysis and Computational Fluid Dynamics", *AIAA Journal*, Vol.30, No.3, March.

Bazaroo M.S. and Shetly C.M., 1979, *Non-linear Programming, Theory and Algorithms*, John Wiley, New York.

Benini E. and Tourlidakis A., 2001, "Design Optimization of Vand Diffusers for Centrifugal Compressors using Genetic Algorithms", 15th AIAA Computational Fluid Dynamics Conference, 11-14 June, Anaheim, CA, USA.

Bogers P.F., Van den Braembussche R.A. and Breugelmans F.A.E., 1998, "Design and Experimental Verification of an Optimised Compressor Blade", *ASME 98-GT-193.*

Bouzida S. and Mignot C., 1997, "Optimization of Fin Louver Design Based on CFD", *SAE Technical Paper Series*, Pages 43-53.

Bradshaw P. (1971), *An Introduction to Turbulence and its Measurement*, Pergamon Press, Oxford, UK.

Çabuk H. and Modi V., 1991, "Optimum Design of Oblique Flow Headers", *ASHRAE Transactions, NY-91-13-2,* Pages 803-813.

Çabuk H. and Modi V., 1992, "Optimum Plane Diffusers in Laminar Flow", *J.Fluids Mechanics 237,* 373-393.

Coello C.A.C., 1999, "A comprehensive survey of evolutionary-based Multiobjective Optimization techniques", *Knowledge and Information Systems: An International Journal*;1(3):269-308.

Deb K., 1995, *Optimization for Engineering Design: Algorithms and Examples*, Prentice-Hall, New Delhi.

Deb K., 2000, "An efficient constraint Handling Method for Genetic Algorithms", *Computer Methods in Applied Mechanics and Engineering*, 186, 311-338.

DeJong K.A., 1975, "An Analysis of the Behaviour of a Class of Genetic Adaptive Systems", Ph.D. thesis, University of Michigan.

Demeulenaere A. 1997, "An Euler/Navier-Stokes Inverse Method for Compressor and Turbine Blade Design", Von Karman Institute for Fluid Dynamics, Inverse Design and Optimisation Methods, April 21-25 Lecture Series 1997-05, 1-45.

Demeulenaere A. and Van den Braembussche R.A., 1998a, "Three-dimensional inverse method for turbomachinery blading design", *ASME J. of Turbomachinery.*

Demeulenaere A., Leonard O. and Van den Braembussche R., 1998b, "Application of a Three-dimensional inverse Method to the Design of a Centrifugal Compressor Impeller", *ASME 98-GT-127.*

Doorly D.J. and Peirò J., 1997, "Supervised Parallel Genetic Algorithms in Aerodynamic Optimisation," *AIAA Paper 97-1852.*

Dorigo M. and Maniezzo V., 1993, "Parallel Genetic Algorithms: Introduction and Overview of Current research", in: J. Stender (Ed.), Parallel Genetic Algorithms: Theory & Applications Frontiers in Artificial Intelligence and Applications, IOS Press, Amsterdam, 1993, pp. 5-42.

Dulikravich G.S., 1991, "Aerodynamic Shape Design and Optimization", *AIAA Paper 91-0476.*

Duxbury P.M. and Dobrin R., 1999, "Greedy algorithms in disordered systems", Physica A: Statistical Mechanics and its Applications, Volume 270, Issues 1-2, 1 August 1999, Pages 263-269

Elbanna H.M. and Carlson L.A., 1990, "Determination of Aerodynamic Sensitivity Coefficients Based on Transonic Small Perturbation Formulation", *J.Aircraft 27(6)*, 507-515.

Eleshaky M.E. and Baysal O., 1993, "Airofoil Shape Optimization Using Sensitivity Analysis on Viscous Flow Equations", *J. Fluids Engineering 115*, 75-84.

Eshelman L.J. and Schaffer J.D., 1993, "Crossover's Niche", Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 9-14.

Fay A., 1976, "On the High Accuracy Scaling up for Pumps and Turbines", Conference Proceeding of Pumps/Turbines, NEL, E.Kilbride.

Fogel D.B., 1995, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ.

Fogel L.J., Owens A.J. and Walsh M.J., 1966, *Artificial Intelligence Through Simulated Evolution*, Wiley, New York.

Fonseca C.M. and Fleming P.J., 1996, "An Overview of Evolutionary Algorithms in Multi-Objective Optimization", *Evolutionary Computation*, 3(1): 1-16, 1996.

Fonseca C.M. and Fleming PJ., 1993, "Genetic Algorithms for Multi-Objective Optimization: Formulation, Discussion and Generalization", Forrest S, editor. Proceedings of the Fifth International Conference on Genetic Algorithms. Los Altos, CA: University of Illinois at Urbana-Champaign, Morgan Kauffman, pp. 416-423.

Fourman M.P., 1985, "Compaction of Symbolic layout using genetic algorithms", Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms. London: Lawrence Erlbaum, pp. 141-153.

Gage P. and Kroo I., 1992, "Development of the Quasi-Procedural Method for use in Aircraft Configuration Optimization," Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, September 21-23, Cleveland, OH, *AIAA 92-4693*.

Giles M. and Drela M., 1987, "Two-Dimensional Transonic Aerodynamic Design Method", *AIAA Journal 25(9)*, 1199-1206.

Glowinski R. and Pironneau O., 1975, "On the Numerical Computation of the Minimum-Drag Profile in Laminar Flow", *J.Fluid Mechanics 72(2)*, 385-389.

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

Goldberg D.E. and Richardson J., 1987, "Genetic Algorithm with Sharing for Multimodal Function Optimization", Grefemstette JJ, editor. Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms. London: Lawrence Erlbaum, 1987. p. 41-9.

Goldberg D.E., Deb K. and Clark J.H., 1992, "Genetic Algorithms, Noise, and the Sizing of Populations", *Complex Systems*, 6, pp. 333-362.

Goldberg D.E., Deb K. and Thierens D., 1993, "Toward a Better Understanding of Mixing in Genetic Algorithms", Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 190-195.

Goodman E.D., Punch W.F. and Lin S., 1994, "Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach", Sixth IEEE Parallel and Distributed Processing, pages 28-37.

Gorges-Schleuter M., 1992, "Comparison of Local Mating Strategies in Massively Parallel Genetic Algorithms", in: R. Männer, B. Manderick (Eds.), Parallel Problem Solving from Nature 2, Elsevier, Amsterdam, 1992, pp. 553-562.

Greenwell R. N., Angus J. E. and Finck M., 1995, "Optimal Mutation Probability for Genetic Algorithms", Mathematical and Computer Modelling, Volume 21, Issue 8, pp 1-11.

Hart W.E. and Belew R.K., 1991, "Optimizing an Arbitrary Function is Hard for the Genetic Algorithm", Proceedings of the Fourth International Conference on Genetic Algorithms. pp. 190-195.

Hajela P. and Lin CY., 1992, "Genetic Search strategies in Multicriterion Optimal Design", Structural Optimization 1992;4:99-107.

Hessner J. and Männer R., 1991, "Choosing Optimal Mutation Rates", in Proceedings of the First Workshop on Parallel Problem Solving from Nature (Lecture Notes in Computer Science, Vol. 496); P. Schwefel and R. Männer (Eds.); Springer-Verlag: Berlin, 1991, pp 23-31.

Hicks R.M., Murman E.M. and Vanderplaats G.N., 1974, "An Assessment of Airfoil Design by Numerical Optimization", *NASA TM X-3092.*

Hicks R.M., Murman E.M. and Vanderplaats G.N., 1974, "An Assessment of Airfoil Design by Numerical Optimization", *NASA TM X-3092.*

Holland J.H., (1975/1992), *Adaptation in Natural and Artificial Systems*, MIT Press, (First Edition 1975, Ann Arbor: University of MichiganPress).

Hollstien R.B., 1971, "Artificial Genetic Adaptation in Computer Control Systems", Ph.D. thesis, University of Michigan.

Homaifar A., Lai S.H.-V. and Qi X., 1994, "Constrained Optimisation Via Genetic Algorithms", *Simulation* 62 (4) 242-254.

Hong Hu, 1999, "Development of a Sensitivity Derivative Version of the FPX CFD Code Via Automatic Differentiation", *Advances in Engineering Software,* Volume 30, Issue 4, April, pp. 273-279.

Horn J. and Nafpliotis N., 1993, "Multiobjective Optimization Using the Niched Pareto Genetic Algorithm", Technical Report IlliGAL report 93005, University of Illinois at Urbana-Champaign, Urbana, IL, USA.

Jameson A., 1995, "Optimum Aerodynamic Design Using CFD and Control Theory", *AIAA Paper 95-1729.*

Jansen W., 1995, "Computer-Aided Design of Pumps", NREC Internal Report.

Japikse D., Marscher W.D. and Furst R.B., 1997, *Centrifugal Pump Design and Performance*, Concepts EDI. Inc., Wilder, Vermont, USA.

Johansen M.D., Lauridsen S., Risager L., Madsen J.I. and Lundgren K.D., 1997, "Response Surface Approximations for Shape Optimization of a 3D-Manifold", Proceedings of Fourth CFX International Users Conference, Chicago, IL, October 6-10, pp. 222-231.

Joines J.A. and Houck C.R., 1994, "On the Use of Nonstationary Penalty Functions to Solve Nonlinear Constrained Optimisation Problems with GAs", in: Z. Michalewicz (Ed.), Proceedings of the International Conference on Evolutionary Computation, IEEE Press, Piscataway, 1994, pp. 579-584.

Jones W.P. and Launder B.E., 1972, "The Prediction of Laminarization with a Two-Equation Model of Turbulence", *International Journal of Heat and Mass Transfer, Vol. 15*, pp. 301-314.

Jones W.P. and Launder B.E., 1973, "The Calculation of Low-Reynolds-Number Phenomena with a Two-Equation Model of Turbulence", *International Journal of Heat and Mass Transfer, Vol. 16*, pp. 1119-1130.

Karassik, 1986, *Pump Handbook*, 2nd edition, McGraw-Hill, New York.

Kline S.J. and Johnston J.P., 1986, "Diffussers – Flow Phenomena and Design", in: David Japikse (Ed.), Advanced Topics in Turbomachinery Technology, Principal Lecture Series No.2, Concepts ETI, Inc., Norwich, Verment 05055, USA.

Kondo Y., Behnia M., et.al., 1998, "Optimization of Finned Heat Sinks for Impingement Cooling of Electronic Packages", *J. of Electronic Packaging 120*, pp. 259-266.

Kroo I. and Takai M., 1988, "A Quasi- Procedural, Knowledge-Based System for Aircraft Design," *AIAA Paper No. 88-4428.*

Kroo I. and Takai M., 1990, "Aircraft Design Optimization Using a Quasi- Procedural Method and Expert System", Third Air Force/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, San Francisco, September 24-26.

Launder B.E. and Spalding D.B., 1974, "The Numerical Computation of Turbulent Flows", *Computer Methods in Applied Mechanics and Engineering, Vol.3*, 269-289.

Lazarkiewicz S. and Troskolanski A.T., 1965, *Impeller pumps*, Pergamon Press Ltd, London.

Lighthill M.J., 1945, "A Method of Two-Dimensional Aerodynamic Design", *R&M 2112, Aeronautical Research Council, London.*

Madsen J., Shyy W., Haftka R. and Liu J., 1997, "Response Surface Techniques for Diffuser Shape Optimization", *AIAA Paper 97-1801.*

Makinen R.A.E., Periaux J. and Toivanen J., 1999, "Multidisciplinary Shape Optimization in Aerodynamics and Electromagnetics Using Generic Algorithms", *International J. for Numerical Methods in Fluids 30* (2), Pages 149-159.

Michalewicz Z. and Attia N., 1994, "Evolutionary Optimisation of Constrained Problems", in: A.V. Sebald, L.J. Fogel (Eds.), Proceedings of the Third Annual Conference on Evolutionary Programming, World Scientific, Singapore, pp. 98-108.

Michalewicz Z., 1995, "Genetic Algorithms, Numerical Optimisation, and Constraints", in: L. Eshelman (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kauffman, San Mateo, pp. 151-158.

Michalewicz Z. and Schoenauer M., 1996, "Evolutionary Algorithms for Constrained Parameter Optimisation Problems", *Evolutionary Computation* 4 (1) 1-32.

Mital K.V., 1977, *Optimization Methods in Operation Research and System Analysis*, Wiley Eastern Limited, New Delhi.

Narducci R., Grossman B., Valorani M., Dadone A. and Haftka R.T., 1995, "Optimization Methods for Non-smooth or Noisy Objective Functions in Fluid Design Problems", *AIAA Paper 95-1648.*

Neumann B., 1991, *The Interaction Between Geometry and Performance of a Centrifugal Pump*, Mechanical Engineering Publications Limited, London.

Newman J.C., Taylor A.C., et.al., 1999, "Overview of Sensitivity Analysis and Shape Optimization for Complex Aerodynamic Configurations", *J. of Aircraft 36* (1) Pages 87-96.

Nielsen K.K. and Myllerup C.M., 1998, "Optimization of Swirl Brakes", *ASME 98-GT-328.*

Nixon R.A., 1966, "Examination of the Problem of Pump Scale Laws", Conference Proceeding of Pump Design Testing and Operation, NEL., East Kilbride, Paper D2-1.

Nixon R.A. and Cairney W.D., 1972, "Scale Effects in Centrifugal Cooling Water Pumps for Thermal Power Stations", NEL Report No. 505/1972.

Oksuz O. and Akmandor I., 2001, "Aerodynamic Optimization of Turbomachinery Cascades using Euler/Boundary-Layer Coupled Genetic Algorithms", 15th AIAA Computational Fluid Dynamics Conference, 11-14 June, Anaheim, CA, USA.

Orszag S.A. and Staroselsky I., 2000, "CFD: Progress and Problems", *Computer Physics Communications* 127, pp. 165-171.

Osterwalder J. and Ettig C., 1977, "Determination of Individual Losses and Scale Effect by Model Tests with a Radial Pump", Conference Proceding of Scaling for Performance Prediction in Rotodynamic Machines, I.Mech.E., Stirling, Paper C185/77.

Oyama A. and Liou M.-S., 2001, "Optimization of Rocket Engine Pumps using Evolutionary Algorithm", 15th AIAA Computational Fluid Dynamics Conference, 11-14 June, Anaheim, CA, USA.

Pandya M. and Baysal O., 1997, "Gradient-based Aerodynamic Shape Optimization Using Alternating Direction Implicit Method," *J. of Aircraft*, 34(3), pp.346-352.

Patankar S.V., 1980, *Numerical Heat Transfer and Fluid Flow*, Series in Computational Methods in Mechanics and Thermal Sciences, Washington: Hemisphere Pub. Corp, New York: McGraw- Hill

Patankar S.V. and Spalding D.B., 1972, "A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flows", *Int.J. Heat & Mass Transfer 15*, 1787

Pearsall I.S., 1966, "The Design and Performance of Supercavitating Pumps", Conference Proceeeding of Pump Design Testing and Operation, NEL, East Kilbride, Paper C2-2.

Pearsall I.S., 1973, "Design of Pump Impellers for Optimum Cavitation Performance", Proceeding of I.Mech.E., Vol. 187, No. 55/73.

Périaux J., Sefrioui M., Stoufflet B., Mantel B. and Laporte E., 1995, "Robust genetic algorithms for optimization problems in aerodynamic design", in: M. Galàn, G. Winter, J. Périaux, P. Cuesta (Eds.), *Genetic algorithms in engineering and computer science*, Wiley, New York, pp. 371-396.

Pironneau O., 1973, "On Optimum Design in Stokes Flow", *J.Fluid Mechanics 59*, pp. 117-128.

Pironneau, O., 1974, "On Optimum Design in Fluid Mechanics", *J.Fluid Mechanics 64*, 97-110.

Poloni C., 1995, "Hybrid GA for multi-objective aerodynamic shape optimization", in: M. Galàn, G. Winter, J. Périaux, P. Cuesta (Eds.), *Genetic Algorithms in Engineering and Computer Science*, Wiley, New York, pp. 397-415.

Qiu-Lin Ding and Davies B.J., 1987, *Surface Engineering Geometry for Computer-Aided Design and Manufacture*, Ellis Horwood series in mechanical engineering, Chichester: Horwood.

Quagliarella D., 1995, "Genetic algorithms applications in computational fluid dynamics", in: M. Galàn, G. Winter, J. Périaux, P. Cuesta (Eds.), *Genetic algorithms in engineering and computer science*, Wiley, New York, pp. 417-442.

Rao SS., 1984, "Multiobjecive Optimization in Structural Design with Uncertain Parameters and Stochastic Processes", *AIAA Journal*; 22(11):1670-8.

Reuther J., Alonso J.J., Rimlinger M.J. and Jameson A., 1999, "Aerodynamic Shape Optimization of Supersonic Aircraft Configurations Via an Adjoint Formulation on Distributed Memory Parallel Computers", Computers & Fluids, Volume 28, Issues 4-5, 6 May 1999, Pages 675-700.

Reneaux J. and Thibert J.J., 1985, "The use of Numerical Optimization for Airfoil Design", *AIAA Paper* 85-5026.

Richardson J.T., Palmer M.R., Liepins G. and Hilliard M., 1989. "Some Guidelines for Genetic Algorithms with Penalty Functions", Proceedings of third International Conf. Genetic Algorithms, pp. 191-197.

Rudolph G., 1991, "Global optimization by means of distributed evolution strategies", in: H.-P. Schwefel, R.Männer (Eds.), Parallel Problem Solving from Nature, Proceedings of the First Workshop PPSNI, Lecture Notes in Computer Science, vol. 496, Springer, Berlin, 1991, pp. 209-213.

Schaffer J., Caruna R. A., Eshelman L. J. and Das R., 1989, "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimisation", Proceedings of the third International Conference on Genetic Algorithms and Their Applications, pages 51-60, San Mateo, California.

Schaffer J.D., 1985, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms", Genetic algorithms and their applications: Proceedings of the First International Conference on Genetic Algorithms. London: Lawrence Erlbaum, p. 93-100.

Schwefel H.P., 1995, *Evolution and Optimum Seeking Sixth-Generation Computer Technology Series*, Wiley, New York.

Schweiger F., 1969, "The Shift in Best Efficiency Point on Changing the Geometry of Pumps", Conference Proceeding of Fluid Mechanics and Fluid Machinery, Academia Kaido, Budapest, Hungary, Page 547.

Siegel J.M. and Makhijani V.B., 1998, "Computational Optimization of Biomedical Devices", ASME Proceedings for IMECE'98 39, Pages 85-86.

Spiessens P. and Manderick B., 1989, "Fine-grained parallel genetic algorithms", in: J.D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, CA, 1989, pp. 428-433.

Srinivas N. and Deb K., 1993, Multiobjective Optimization Using Non-Dominated Sorting in Genetic Algorithms. Technical Report, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India, 1993.

Stamatiadis S., Prosmiti R. and Farantos S.C., 2000, "AUTO_DERIV: Tool for Automatic Differentiation of a Code", Computer Physics Communications, Volume 127, Issues 2-3, 10 May 2000, Pages 343-355.

Stepanoff A.J., 1948, *Centrifugal and Axial Flow Pumps: theory, design, and application*, J.Wiley, New York.

Stepanoff A.J., 1976, *Centrifugal and Axial Flow Pumps*, J.Wiley, New York.

Suzuki J., 1993, "A Markov Chain Analysis on a Genetic Algorithm", Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 146-153.

Tauber M.E. 1989, "A Review of High-Speed, Convective, Heat-Transfer Computation Methods", *NASA/TP-2914.*

Taylor A.C., Hou G.W. and Korivi V.M., 1991, "Sensitivity Analysis", *AIAA Paper 91-3083.*

Taylor A.C., Korivi V.M. and Hou G.W., 1992, "Taylor Series Approximation of Geometric Shape Variation for the Euler Equations", *AIAA Journal* 30 (8), pp. 2163-2165.

Thompson J.F., Warsi Z.U.A. and Mastin C.W., 1985, *Numerical Grid Generation – Foundations and Applications*, Elsevier Publishing Company.

Thorne E.W., 1979, "Design by the Area Ratio Method", Conference Proceeding of Pumps, Sixty Tech. Conf., BPMA, Canterbury, Paper C2.

Turton R.K., 1994, *Rotordynamic pump design*, Cambridge University Press, Cambridge, UK.

Tuzson J., 2000, *Centrifugal Pump Design*, John Wiley & Sons, Inc., New York, USA.

Van den Braembussche R.A., Pierret S. and Demeulenaere A., 1998, "Modern Turbo-Machinery Blade Design", VKI/Reprint-1998/21,

Varley F.A., 1961, "Effects of Impeller Design and Surface Roughness on the performance of Centrifugal Pumps", Proceeding of I Mech E, Vol. 175, No. 21.

Wahba W.A. and Tourlidakis A., 2001a, "A Genetic Algorithm Applied to the Design of Blade Profiles for Centrifugal Pump Impellers", 15th AIAA Computational Fluid Dynamics Conference, 11-14 June, Anaheim, CA, USA.

Wahba W.A. and Tourlidakis A., 2001b, "A Parallel Genetic Algorithm Applied to the Design of Blade Profiles for Centrifugal Pump Impellers", Seventh International Conference of Fluid Dynamics and Propulsion, 19-21 December, Sharm El-sheik, Egypt.

Wahba W.A., 1997, "Flow Pattern in A Centrifugal Pump used in Turbo-Pump Feed System", M.Sc. Thesis, Military Technical College, Cairo, Egypt.

Wahba W.A., Abdallh H. and Allam M., 1998a, "Flow Pattern Solution for a Radial Blade Centrifugal Pump", Seventh International Conference on Aerospace Sciences & Aviation Technology, 13 - 15 May, MTC, Cairo, Egypt.

Wahba W.A., Abdallh H. and Allam M., 1998b, "Flow Pattern Solution for a Backward Blade Centrifugal Pump", Seventh International Conference on Aerospace Sciences & Aviation Technology, 13 - 15 May, MTC, Cairo, Egypt.

Wall M., 1996, "GAlib: A C++ Library of Genetic Algorithm Components", version 2.4, Massachusetts Institute of Technology, USA.

Whitfield A. and Baines N.C., 1990, *Design of Radial Turbomachines*, Longman Group UK Ltd., England.

Worster R.C, 1963, "The Flow in Volutes and its Effect on Centrifugal Pump Performance", Proceeding of I Mech E, Vol. 177, No. 31.

Zangeneh M., Goto A. and Takemura T., 1996, "Suppression of Secondary Flows in a Mixed-Flow Pump Impellers by Application of Three-Dimensional Inverse Design Method: Part 1- Design and Numerical Validation", *J. Turbomachinery 118*, 536-543.

Appendix

A

Mac_LNS CFD Code for Pump Impeller

In the following appendix a summary of the Mac_LNS CFD code, Wahba (1997), will be provided in order to give a general understanding and form the necessary background for how to use this code and CFD in general in combination with an optimisation algorithms.

Firstly, a discussion of the flow in turbomachinery devices will take place, where a number of simplifying assumptions used in this code is provided. The governing differential equations of steady, incompressible, viscous flows are presented for a two-dimensional, polar-coordinate system. An explicit, finite difference scheme used to solve these governing equations has been described.

A.1 INTRODUCTION

Turbomachinery flows are among the most complex flows encountered in fluid dynamic practice. In most instances, they are three-dimensional, with laminar, transitional and turbulent flow; separated flows are frequently encountered. The flow may be incompressible, subsonic, transonic, or supersonic; some turbomachinery flows include all of these flow regimes. The flow may be single-phase or two-phase (liquid-solid or liquid-gas).

The viscous and turbulent regions encounter complex stress and strain due to three-dimensionality, appreciable pressure gradients in all directions, rotation, curvature, shock-boundary layer interaction, heat transfer and interacting boundary layers. The flow is dominated by vortical flows: secondary, leakage, trailing, horseshoe and scraping vortices.

The absolute flow is always unsteady in a rotor, and both the relative and absolute flows are unsteady in a multi-stage environment. The equations are strongly coupled and complex boundary conditions are often encountered (transpiration, periodicity, etc.). The free stream turbulence is usually high. However, not all these effects are present in any given blade row. The flow and geometrical variable direct the nature of governing equations and flow solvers to be used (Lakshminarayana B., 1991).

The present Appendix presents the development of a steady two-dimensional, incompressible N-S equations solved in a polar-coordinate system using primitive variables. The method was applied to centrifugal pumps with a radial and a backward-curved blade, Wahba et.al. (1998a&b). The difficult problem of calculating the flow pattern can be tackled only after making a number of simplifying assumptions.

1. The relative flow through the impeller passage is laminar, incompressible, and steady.
2. The impeller is assumed to be rotating in an infinite field.
3. The thickness of the impeller blades is neglected.
4. The effect of the turning from the axial to radial direction is neglected.
5. The width of the impeller passage is constant.
6. The velocity across the axial direction is constant and hence the flow can be considered as a two-dimensional flow.

A.2 MAIN GOVERNING EQUATIONS

Consider the polar-coordinate system (r, θ), where r denotes the radial direction and θ denotes the tangential direction. Consider the two-dimensional, incompressible N-S equations for a constant property flow without body forces or external heat addition. The continuity equation, written in the system relative to a blade row, is:

$$\frac{1}{\rho}\frac{\partial}{\partial r}(ru_r)+\frac{1}{\rho}\frac{\partial}{\partial \theta}(u_\theta)=0 \qquad \text{(B.1)}$$

where:

u_r, u_θ are the relative flow velocity components in r and θ directions respectively.

ρ is the operating fluid density.

After the above assumptions, the momentum conservation law for a rotating blade is the N-S equations written as:

$$\frac{\partial}{\partial t}(u_r)+\frac{1}{r}\frac{\partial}{\partial r}(ru_r^2)+\frac{1}{r}\frac{\partial}{\partial \theta}(u_r u_\theta)-\frac{u_\theta^2}{r}-r\omega^2-2\omega u_\theta=-\frac{1}{\rho}\frac{\partial p}{\partial r}+\nu\left[\nabla^2 u_r-\frac{u_r}{r^2}-\frac{2}{r^2}\frac{\partial u_\theta}{\partial \theta}\right]$$

$$\frac{\partial}{\partial t}(u_\theta)+\frac{1}{r}\frac{\partial}{\partial r}(ru_r u_\theta)+\frac{1}{r}\frac{\partial}{\partial \theta}(u_\theta^2)+\frac{u_r u_\theta}{r}+2\omega u_r=-\frac{1}{\rho r}\frac{\partial p}{\partial \theta}+\nu\left[\nabla^2 u_\theta-\frac{u_\theta}{r^2}+\frac{2}{r^2}\frac{\partial u_r}{\partial \theta}\right]$$

where:

p is the static pressure.

ω is the angular velocity of the rotating blade.

ν is the kinematic viscosity.

These equations are written in the so-called primitive-variable form where p, u_r, and u_θ are the primitive-variables (Hirsch, 1976). The computation of incompressible flows is not straightforward as in the case of computation of compressible flow. The most characteristic aspect of the computation of incompressible flow is the difficulty of extracting the pressure from the combined continuity and momentum equations. One common procedure is to define a Poisson equation or an especially formulated correction equation for the pressure. Application of compressible algorithms to the incompressible equations is accomplished by adding a time derivative term to the continuity equation in a manner analogous to that originally suggested by Chorin (1967).

The artificial density is related to the pressure by the artificial equation of state:

$$p=\delta\tilde{\rho} \tag{B.4}$$

where:

$\tilde{\rho}$ is the artificial density (variable).

δ is an artificial compressibility factor (constant).

Considering constant density ρ, the continuity equation (B.1) can be rewritten in the form:

$$-\frac{\partial}{\partial t}(\tilde{\rho}) = \rho\left[\frac{1}{r}\frac{\partial}{\partial r}(ru_r) + \frac{1}{r}\frac{\partial}{\partial \theta}(u_\theta)\right] \tag{B.5}$$

Using equation (B.4).

$$-\frac{\partial}{\partial t}(p) = \rho\delta\left[\frac{1}{r}\frac{\partial}{\partial r}(ru_r) + \frac{1}{r}\frac{\partial}{\partial \theta}(u_\theta)\right] \tag{B.6}$$

The following is a trial to investigate the artificial compressibility factor δ. An artificial equation of state implies the existence of an artificial sound speed $(\tilde{a}^*)$ given by:

$$\tilde{a}^* = \sqrt{\frac{\partial p}{\partial \tilde{\rho}}} = \sqrt{\delta} \tag{B.7}$$

The maximum artificial Mach number $(\tilde{M}_{\max})$ based on this artificial sound speed is required to be less than unity.

$$\tilde{M} = \frac{\sqrt{(u_r^2 + u_\theta^2)_{\max}}}{\tilde{a}^*} = \frac{\sqrt{(u_r^2 + u_\theta^2)_{\max}}}{\sqrt{\delta}} \leq 1 \tag{B.8}$$

The following condition is obtained:

$$\delta \geq (u_r^2 + u_\theta^2)_{\max} \tag{B.9}$$

A.3 NUMERICAL ALGORITHM

A.3.1 Computational Grid

The computational domain is discretized into mesh points, figure B.1, to enable discretization of the equations. The present scheme calculates the flow through one blade passage. The computational boundaries comprise from the upstream inlet, the downstream exit, the pressure and suction side blade surfaces. The blade surfaces were extended along a surface of grid points in the upstream and downstream direction. These form permeable periodic boundaries. A polar grid is used in the present method.

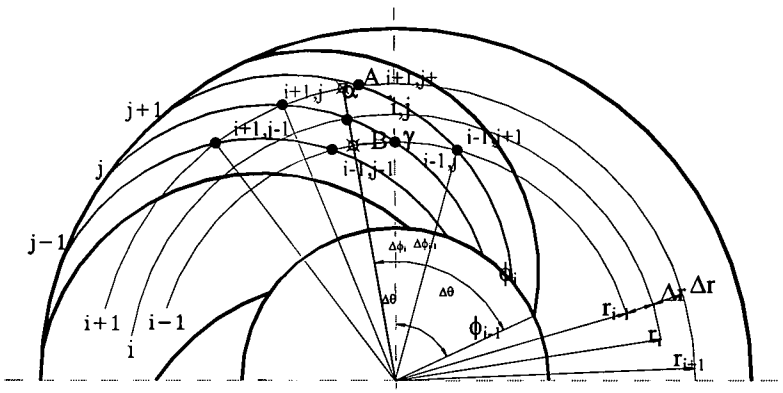


Fig. B.1 Grid geometry selected for solution.

Where:

$$\Delta\phi_i = \phi_{i+1} - \phi_i$$
$$\alpha_{i,j} = -r_{i+1}\Delta\phi_i$$
$$\gamma_{i,j} = r_{i-1}\Delta\phi_{i-1}$$
$$S_{i-1} = r_{i-1}\Delta\theta$$
$$S_{i+1} = r_{i+1}\Delta\theta$$

A.3.2 Finite Difference Scheme

There are two approaches to calculate incompressible N-S equations: implicit and explicit techniques. We will use the explicit one for which the computational procedures are relatively easy to implement. The most appropriate system of equations in differential form is the Reynolds averaged N-S equations in a rotating polar-coordinate system given by Lakshminarayana (1991).

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{r}\frac{\partial (r\mathbf{E})}{\partial r} + \frac{1}{r}\frac{\partial \mathbf{F}}{\partial \theta} = \frac{1}{r}\mathbf{S} + ViscousTerm \qquad \text{(B.10)}$$

where:

Q is the conservation variable.

E, F are the flux vectors.

S is the source term.

The viscous term will be discarded for its later use as central difference form in r and θ directions. Equation (B.10) becomes:

$$\frac{\partial \mathbf{Q}}{\partial t} = -\frac{1}{r}\frac{\partial (r\mathbf{E})}{\partial r} - \frac{1}{r}\frac{\partial (\mathbf{F})}{\partial \theta} + \frac{1}{r}\mathbf{S} \tag{B.11}$$

Using equations (B.2, B.3, B.6)

$$\mathbf{Q} = \begin{bmatrix} \mathrm{P} \\ \mathrm{u_r} \\ \mathrm{u_\theta} \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \rho\delta \mathrm{u_r} \\ \mathrm{u_r^2 + p/\rho} \\ \mathrm{u_r u_\theta} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho\delta \mathrm{u_\theta} \\ \mathrm{u_r u_\theta} \\ \mathrm{u_\theta^2 + p/\rho} \end{bmatrix}, \text{ and}$$

$$\mathbf{S} = \begin{bmatrix} 0 \\ \mathrm{u_\theta^2 + p/\rho + \omega^2 r^2 + 2\omega r u_\theta} \\ \mathrm{-u_r u_\theta - 2\omega r u_r} \end{bmatrix} \tag{B.12}$$

where the element $[p/\rho]$ in the flux vector **E** and source term **S** respectively are obtained from substituting $\left[-\frac{1}{\rho}\frac{\partial p}{\partial r}\right]$ in equation (B.2) with $\left[\frac{1}{r}\frac{p}{\rho} - \frac{1}{r}\frac{\partial}{\partial r}(r\, p/\rho)\right]$.

A.3.3 Predictor-Corrector Method

The Predictor-Corrector method proposed by MacCormack (Anderson 1984) is a two-step procedure based on the Lax-Wendroff scheme, and is widely used for both internal and external flows. The method is second-order accurate in both time and space. It can be used for both steady and unsteady compressible flow, as well as viscous and inviscid flows. For the inviscid flow, in the procedure suggested by MacCormack, an iterative approach and intermediate value $\overline{\mathbf{Q}_{i,j}^{n+1}}$ is obtained by a predictor step, and $\mathbf{Q}_{i,j}^{n+1}$ is obtained by a corrector step, where n is the time step and n+1 is the next one. The predictor step written for the two dimensional (2D) inviscid equations in polar (r, θ) system is given by:

$$\overline{\mathbf{Q}_{i,j}^{n+1}} = \mathbf{Q}_{i,j}^{n} - \frac{\Delta t}{\Delta r}\left[\mathbf{E}_{i+1,j}^{n} - \mathbf{E}_{i,j}^{n}\right] - \frac{\Delta t}{r_i \Delta\theta}\left[\mathbf{F}_{i,j+1}^{n} - \mathbf{F}_{i,j}^{n}\right] + \frac{\mathbf{S}_{i,j}\Delta t}{r_i} \tag{B.13}$$

It should be highlighted here that this step provides only an approximate value for $\mathbf{Q}_{i,j}^{n+1}$, and this can be corrected or updated using the following corrector step:

$$\mathbf{Q}_{i,j}^{n+1} = \frac{1}{2}\left\{\mathbf{Q}_{i,j}^{n} + \overline{\mathbf{Q}_{i,j}^{n+1}} - \frac{\Delta t}{\Delta r}\left[\overline{\mathbf{E}_{i,j}^{n+1}} - \overline{\mathbf{E}_{i-1,j}^{n+1}}\right] - \frac{\Delta t}{r_i \Delta\theta}\left[\overline{\mathbf{F}_{i,j}^{n+1}} - \overline{\mathbf{F}_{i,j-1}^{n+1}}\right] + \frac{S_{i,j}\Delta t}{r_i}\right\} \tag{B.14}$$

where:

Δr is the element length in r direction.

$\Delta\theta$ is the element angle in θ direction.

The forward and backward differencing can be alternated between predictor and corrector steps as well as between the two spatial derivatives in a sequential fashion. The sequence is given in four time steps as shown in Table 1, in such a way to eliminate any bias due to the one-sided differencing.

Table B.1. Differencing sequence for the MacCormack scheme*

Predictor		**Corrector**	
$\frac{\partial}{\partial r}$	$\frac{\partial}{\partial \theta}$	$\frac{\partial}{\partial r}$	$\frac{\partial}{\partial \theta}$
F	F	B	B
F	B	B	F
B	F	F	B
B	B	F	F

* (F = Forward, B = Backward)

This procedure will be repeatedly carried out as required for the next four time steps.

A.3.4 Time Marching Procedure

The equations are solved using a time-marching procedure as is described in the following steps:

1) Specify initial values for u_r, u_θ, p at time $t = 0$.
2) Get stability condition, which is used to converge the solution. In other words, calculate time steps Δt for all the grid points.
3) Solve the continuity equation and N-S equation at each interior grid point (predictor and corrector).
4) Perform the appropriate smoothing for the primitive variables to maintain stability in the numerical solution.
5) Find the primitive variables at the boundary using their values at the interior points.
6) Return to step 3 if the solution is not converged.

Practically, it was found unnecessary to recalculate the stability condition (step 2) after performing each time step computation.

A.3.5 Smoothing Terms

For algorithms of the present type, it is often necessary to add smoothing terms in order to suppress high frequency oscillations. This can easily be accomplished by adding a fourth-order explicit dissipation term to the primitive variables in the two-directions of flow (r,θ) for interior points. The added term has the form:

$$-\varepsilon_e\left[(\Delta r)^4\frac{\partial^4}{\partial r^4}(\mathbf{Q})+(\Delta\theta)^4\frac{\partial^4}{\partial\theta^4}(\mathbf{Q})\right] \tag{B.15}$$

where ε_e is the explicit smoothing coefficient.

Since this is a fourth-order term it does not affect the formal accuracy of the algorithm. The negative sign is required in order to produce positive damping. The smoothing coefficient ε_e should be less than approximately 1/16 for stability (Anderson 1984). A value of 0.05 is used in the present work. The fourth-derivative terms are evaluated using finite-difference approximations.

A.3.6 Stability Condition and Convergence Criteria

It is necessary to find a value of the time step Δt for which the stability of the solution could be maintained. For the inviscid and incompressible time marching method, a complete stability analysis is reported by Abdalla (1981), who could express the maximum possible time step for the 2D problem. Because of the complexity of N-S equations, it is not possible to obtain a closed form stability expression for the MacCormack scheme applied to the governing equations. Anderson (1984) used an empirical formula in case of one-dimensional N-S equations. The expressions of Abdalla and Anderson have been combined into one empirical formula, suitable for 2D incompressible N-S equations, in the form (Wahba, 1997):

$$\Delta t=\left(\frac{2F_t}{\frac{|u_r|}{\Delta r}+\frac{|u_\theta|}{r\Delta\theta}+4\nu\left(\frac{1}{(\Delta r)^2}+\frac{1}{(r\Delta\theta)^2}\right)+\sqrt{\left(\frac{u_r}{\Delta r}+\frac{u_\theta}{r\Delta\theta}\right)^2+4\delta\left(\frac{1}{(\Delta r)^2}+\frac{1}{(r\Delta\theta)^2}\right)}}\right) \tag{B.16}$$

where F_t is the time factor. It is found from experience that the time factor up to 0.9 can be used. In case of an unstable solution, the time factor is reduced by 0.1.

Time steps are calculated based on the initial conditions and are not updated during the calculations. Thus, the time step varies as a function of grid spacing only. The iteration process is repeated until it converges. The computation is considered to be converged when the root mean square of the residual in the velocity component u_r drops below 10^{-5}.

$$RMS = \sqrt{\frac{\sum_{i=1}^{N-1}\sum_{j=1}^{M}\left|u_{r_{i,j}}^{n+1} - u_{r_{i,j}}^{n}\right|^2}{(N-1)(M)}} \tag{B.17}$$

where:

N is the total number of grid lines from upstream to downstream extensions in the radial direction.

M is the number of grid lines from blade-to-blade in the θ direction.

A.3.7 Upstream and Downstream Boundary Conditions

The prescription of the inflow and the outflow boundary conditions is one of the most important tasks. These surfaces should be located far upstream and far downstream, where the influence of the blade rows under consideration is negligible. Hence, most investigators locate them usually at about one to one half-chord upstream and downstream. The far upstream and downstream boundaries are located about half-chord, and one-chord respectively as shown in figure 4.9. On the upstream boundary the relative velocity components u_r and u_θ are specified. On the downstream boundary only the static pressure p is required. The other variables at both upstream and downstream boundaries are to be obtained by interpolation from the interior points.

A.3.8 Walls and their Extension Boundary Conditions

The wall points are considered as if they were interior points for the calculation of all the variables. This required adding a grid line before the pressure side and a grid line after the suction side. The parameters at these lines were obtained by quadratic interpolation from the values at the wall and two interior points. This is numerically exactly the same as applying the conservation equations to a point on the half spacing

near the wall and then extrapolating from this point to the boundary (Abdalla, 1981). To simulate blade row conditions, it is essential to impose zero radial and tangential velocities in case of N-S equations. At the periodic boundary the variables were calculated as the interior points. In this case the periodicity condition could be used to obtain the variables which were located beyond the boundary. The results at corresponding points were then averaged after each time step.

A.3.9 Initial Conditions

The prescription of the flow for all grid points by relatively real values is one of the most important tasks. This initial guess can be completely arbitrary, where this procedure has no effect on the final solution, but it affects the number of iterations needed to converge the solution. Three zones of initial conditions are imaginary upstream grid points, imaginary downstream grid points, and inside blade passage grid points. The data required for the solution are the major impeller dimensions, the flow rate, the fluid viscosity and density, and the impeller speed. The radial velocity is obtained from continuity. The tangential velocities in upstream and downstream are obtained from Euler's equation. Inside the blade passage, as the flow is considered radial, the tangential velocity is neglected. Assuming reasonable starting value for the downstream static pressure, Bernoulli constant at downstream is calculated. The initial values of the static pressure at all the grid points can be readily calculated.

A.4 REFERENCES

Abdalla H., 1981, "A Theoretical and Experimental Investigation of the Regenerative Pump with Aerofoil Blades", Ph.D. Thesis, The Royal Military College of Science, UK.

Anderson D.A., Tannehill J.C. and Pletcher R.H., 1984, *Computational Fluid Mechanics and Heat Transfer*, McGraw Hill Book Company, Second Edition.

Chorin A.J., 1967, "A Numerical Method for Solving Incompressible Viscous Flow Problems", J. Comp. Phys., Vol. 2, pp. 12-26.

Hirsch C. and Warzee G., 1976, "A Finite-Element Method for Through Flow Calculations in Turbomachines", Transactions of the ASME, Journal of Fluids Engineering, pp. 403-410.

Lakshminarayana B., 1991, "An Assessment of Computational Fluid Dynamic Techniques in the Analysis and Design of Turbomachinery - The 1990

Freeman Scholar Lecture", Transactions of the ASME, Journal of Fluids Engineering, Vol. 113, pp. 315-352.

Wahba W.A., 1997, "Flow Pattern in A Centrifugal Pump used in Turbo-Pump Feed System", M.Sc. Thesis, Military Technical College, Cairo, Egypt.

Wahba W.A., Abdallh H. and Allam M., 1998a, "Flow Pattern Solution for a Radial Blade Centrifugal Pump", Seventh International Conference on Aerospace Sciences & Aviation Technology, 13 - 15 May, MTC, Cairo, Egypt.

Wahba W.A., Abdallh H. and Allam M., 1998b, "Flow Pattern Solution for a Backward Blade Centrifugal Pump", Seventh International Conference on Aerospace Sciences & Aviation Technology, 13 - 15 May, MTC, Cairo, Egypt.

Appendix

B

Some Codes Used

B.1 EVA_POP(GAPOPULATION & P) FUNCTION

This function was developed in the present work to over load the default population evaluation one in the GAlib code, in order to use parallel processing in evaluating the population. The parallelisation method used is micro-grain (master/slave) method. Which is the simplest parallel GA model.

This method uses single population (just as serial GA) which is maintained principally by a master, but the evaluation of fitness is distributed among several processors acting as slaves.

Function listing:

```
#ifndef _opt_cfd_mpi_cpp_
#define _opt_cfd_mpi_cpp_

#include <stdio.h>
#include <iostream.h>
#include <fstream.h>
#include <math.h>
#include <ga.h>
#include <ga.c>
#include <mpi++.h>
#include <Mac_LNS_mpi.cpp>              // objective function file (CFD calculation).

void Eva_POP(GAPopulation &) ;          // overload DefaultEvaluator(GAPopulation &) function
void get_max_min(float &, float &) ;
void set_max_min(float, float) ;
void Recv_Update() ;
int count = 0 ;
float max_fitness, min_fitness ;
```

```
int no_p, GADone=0 ;
int rank, size ;
MPI_Request  *req ;
MPI_Status status;

const int MG_Sleep          =0 ;
const int MG_Work           =1 ;
const int MG_Init           =2 ;
const int MG_Eval           =3 ;
const int MG_Fres           =4 ;
const int MG_End            =5 ;

int msg = 0 ;
#endif

void
Eva_POP(GAPopulation & p){

        void Send_Work(int, int, int, float*, int) ;
        float get_score(float*, int, int, float);
        void Send_Update(int) ;
        int gen ;
        float *y;
        y = new float[no_p] ;
        int to = 1, PSize=0 ;
        float score = 0.0 ;
        float *max_s, *min_s ;                          // for the reason of multi-objective.
        max_s = new float [size] ;
        min_s = new float [size] ;

        float qStart, qEnd, qStep ;
        if(msg!=MG_Fres)  {
                for(int i=0; i<p.size(); i++)
                        if(((GABin2DecGenome&)(p.individual(i))).evaluated() == gaFalse) PSize++ ;
        }else {
                PSize = p.size() ;
        }

        cout << "\n\n-------- count: " << count  << " ------- PSize: " << PSize << " ----------------------\n";
        cout.flush();
        for(int i=0; i<PSize; i++) {
                gen = ((GABin2DecGenome&)(p.individual(i))).geneticAlgorithm()->generation() ;

                for(int j=0; j<no_p; j++)       y[j] = ((GABin2DecGenome&)(p.individual(i))).phenotype(j) ;

                if(msg==MG_Fres && !i)
                        { qStart = 0.4 ;        qEnd = 1.6 ;      qStep = 0.1 ;      count = 0 ;        }
                else    {       qStart = 0.0 ;  qEnd = 0.05 ;     qStep = 0.1 ;                         }

                for(float q=qStart; q<=qEnd; q += qStep) {
                        if(count==0 && msg!=MG_Fres) {
                                score = get_score(y, rank, gen, q) ;
                                get_max_min(max_fitness, min_fitness);
                                cout << "\n In main Score: " << score << "  for pop: " << i  ;
                                cout << "\t\t max: " << max_fitness << " & min: " << min_fitness  <<"\n";
                                cout.flush();
                                for(int j=0; j<size; j++) {
                                        max_s[j] = max_fitness ;
```

```
                        min_s[j] = min_fitness ;
                }
                p.individual(i).score(score);
                p.individual(i).nevalpp() ;
                count++ ;

        }else if(to == size || i==PSize-1) {

                score = get_score(y, rank, gen, q) ;
                if(msg!=MG_Fres) {
                        get_max_min(max_s[0], min_s[0]);
                        max_fitness = (max_fitness>max_s[0])? max_fitness : max_s[0] ;
                        min_fitness = (min_fitness<min_s[0])? min_fitness : min_s[0] ;
                        set_max_min(max_fitness, min_fitness) ;
                        cout << "\n In main Score: " << score << "  for pop: " << i  ;
                        cout << "\t\t max: " << max_s[0] << " & min: " << min_s[0]  <<"\n";
                        cout.flush();

                        p.individual(i).score(score);
                        p.individual(i).nevalpp() ;
                }
                int k=0 ;
                for(int j=1; j<size; j++) {
                        if(j<to) {
                         MPI::COMM_WORLD.Recv(&k, 1, MPI::INT, j, 55);

                         cout << "\n I have to recv now from Pro: " << j << "\n" ;
                         cout.flush();
                         MPI_Wait(&(req[j]), &status) ;

                         MPI::COMM_WORLD.Recv(&max_s[j], 1, MPI::FLOAT, j, 77);
                         MPI::COMM_WORLD.Recv(&min_s[j], 1, MPI::FLOAT, j, 88);
                         if(msg!=MG_Fres) {
                                 max_fitness = (max_fitness>max_s[j])? max_fitness : max_s[j] ;
                                 min_fitness = (min_fitness<min_s[j])? min_fitness : min_s[j] ;
                                 cout << " I recved Score: " << score << "  for pop: " << k ;
                                 cout << " from Pro: " << j ;
                                 cout << "\t max: " << max_s[j] << " & min: " << min_s[j] ;
                                 cout <<"\n"; cout.flush();
                                 p.individual(k).score(score);
                                 p.individual(k).nevalpp() ;
                         }
                         }else {
//                       cout << "\n I will send to P: " << j << " ..Sleeeeeep..\n" ;
                         cout.flush();
                         msg = MG_Sleep ;
                         MPI::COMM_WORLD.Send(&msg, 1, MPI::INT, j, 10);
                         Send_Update(j) ;
                         }
                }
                to = 1 ;
                count++ ;
        }else {
//              cout << " I'll send to: " << to  << "\n"; cout.flush();
                Send_Work(to, i, gen, y, count) ;
                if(msg!=MG_Fres)        Send_Update(to) ;
                else    MPI::COMM_WORLD.Send(&q, 1, MPI::FLOAT, to, 11);
```

```
                    req[to] = MPI::COMM_WORLD.Irecv(&score, 1, MPI::FLOAT, to, 4);
                    count++ ;                    to++ ;
                }
            }
        }
}
```

B.2 DE CASTELJAU'S ALGORITHM

This function is used in the present work to develop Bézier curves.

The de Casteljau's algorithm can be explained geometrically: If we divide the edges of the control polygon in the ratio (1-*u*) to *u*, connect the resulting points by straight lines, divide the new edges again in the same ratios, and repeat this process a total of n times, then the dividing point obtained in the last step is the point on the curve corresponding to u, see subsection 4.4.2. So this is the idea of how this algorithm works.

Function listing:

```
float De_Casteljau(float *p, int n, float t, float a, float b) {

        float *temp ;
        temp = new float[n] ;
        for(int i=0; i<n; i++)
                temp[i] = p[i] ;        // save input
        float u ;
        for(int k=1; k<n; k++)
                for(int i=0; i<(n-k); i++) {
                        u = (t-a)/(b-a) ;
                        temp[i] = (1-u)*temp[i] + u*temp[i+1] ;
                }
        float res = temp[0] ;
        delete[] temp ;

        return res ;

}
```