CRANFIELD UNIVERSITY

J M ROGERO

A GENETIC ALGORITHMS BASED OPTIMISATION TOOL FOR THE PRELIMINARY DESIGN OF GAS TURBINE COMBUSTORS

SCHOOL OF ENGINEERING

PhD THESIS

CRANFIELD UNIVERSITY

SCHOOL OF MECHANICAL ENGINEERING

PhD THESIS

Academic Year 2002-2003

J M ROGERO

A Genetic Algorithms Based Optimisation Tool for the Preliminary Design of Gas Turbine Combustors

Supervisor: P A RUBINI

November 2002

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

The aim of this research is to develop an optimisation tool to support the preliminary design of gas turbine combustors by providing a partial automation of the design process. This tool is to enable better design to be obtained faster, providing a reduction in the development costs and time to market of new engines.

The first phase of this work involved the analysis of the combustor design process with the aim of identifying the critical tasks that are suitable for being automated and most importantly identifying the key parameters describing the performance of a combustor.

During the second phase of this work an adequate design methodology for this problem was defined. This led to the development of a design optimisation Toolbox based on genetic algorithms, containing the tools required for it's proper integration into the combustor preliminary design environment. For the development of this Toolbox, extensive work was performed on genetic algorithms and derived techniques in order to provide the most efficient and robust optimisation method possible.

The optimisation capability of the Toolbox was first validated and metered on analytical problems of known solution, where it demonstrated excellent optimisation performance especially for higher-dimensional problems. In a second step of the testing and validation process the combustor design capability of the Toolbox was demonstrated by applying it to diverse combustor

design test cases. There the Toolbox demonstrated its capacity to achieve the required performance targets and to successfully optimise some key combustor parameters such as liner wall cooling flow and NOx emissions. In addition, the Toolbox demonstrated its ability to be applied to different types of engineering problems such as wing profile optimisation.

Acknowledgements

I wish to specially thank my supervisor, Dr Philip Rubini for his helpful advice and direction throughout the entire work.

I am grateful to the Defense Evaluation and Research Agency (DERA now QinetiQ) and Rolls-Royce for the funding and the support provided. I would like to thank particularly Dave Lowe, Robert Hicks and Chris Wilson for their valuable technical input.

I wish to express my gratitude to my parents and family, which have been understanding and supportive, contributing through their encouragement to the completion of the work.

Finally I would like to thank all the friends that I met in Cranfield and in the Salsa club for providing a much needed distraction and support; I think, it wouldn't have been possible to stay in Cranfield that long without them.

Contents

List of Figures

List of Tables

Nomenclature

Letters

a	Constant
b	Constant
$\overrightarrow{b_l}$	Set of variable lower boundary
$\overrightarrow{b_u}$	Set of variable upper boundary
c	Constant
$[A]$	Molar concentration of spices A ($kmol/m^3$)
$\mathcal{F}$	Feasible search space
f	Fitness
g	Generation number
i	Index
k	Formation rate
l	Length of a bit string encoding a gene
LB	Lower bound of a variable

M	Mach number or Molecular weight ($kg/kmol$)
m	Number of bits encoding a chromosome
$\dot{m}$	Mass flow (kg/s)
n	Number
O	Optimisation factor
o	Order of a schemata
$\overrightarrow{o}$	Set of objectives
P	Pressure (kPa)
P_v	Problem input variable
P_o	Performance objective
P_p	Problem performance parameter
$\overrightarrow{p}$	Set of performance parameters
R	Specific Gas Constant ($kJ * kg^{-1} * K^{-1}$)
R_0	Universal Gas Constant $R_0 = 8.314$ ($kJ * kmol^{-1} * K^{-1}$))
R_e	Range error
r	Random number
$\mathcal{S}$	Search space
S	Set or vector of parameters
Sp	Selection Pressure
T	Temperature (K)

T_a	Target achievement factor
t	Time (s)
$\overrightarrow{ta}$	Set of performance targets
$\overrightarrow{t}$	Set of finite performance targets
U	Cooling mass-flow ratio
UB	Upper bound of a variable
V	Volume (m^3)
$\dot{V}$	Volume flow (m^3/s)
$\overrightarrow{Vc}$	Vector of a chromosome
$\overrightarrow{Vd}$	Displacement vector
$\overrightarrow{Vr}$	Resulting vector
v	Variable parameter
$\overrightarrow{v}$	Set of variables

Greek Symbols

Δ	Magnitude range
Λ	Loading Parameter
γ	Decay rate
δ	Magnitude range ratio
η	Efficiency

μ	Precision
ς	Schemata
χ	Relight Loading Parameter

Subscripts

$_3$	Compressor Delivery Conditions (combustor related)
$_4$	Turbine Entry Conditions (combustor related)
$_{an}$	Annulus
$_b$	Bit or binary
$_{bp}$	Base plate
$_c$	Combustion (combustor related) / Cumulative (optimisation related)
$_e$	Evaporation
$_{iw}$	Inner wall
$_{ow}$	Outer wall
$_{po}$	Port
$_{pz}$	Primary zone
$_R$	Relight Conditions
$_r$	Recirculating (combustor related) / Relative (optimisation related)
$_{st}$	Stoichiometric Conditions

Chapter 1

Introduction

Since the deregulation of the US airline market in 1978, which then propagated to the international airline markets [160], the pressure on the airframe and engine manufacturers to produce more efficient, low cost aircraft has increased dramatically. The increased competition forced the airline companies to reduce their commitment as launch customers for new airframe and engines, generating financial uncertainty for these new projects and in turn increasing competition among the manufacturers [118]. In addition, environmental concerns pushed for more stringent legislation on pollutant emissions and noise. The standard regulating NOx emissions of aero-engines [74] was first adopted in 1981, then was made more stringent in 1993 with a reduction of the permitted levels by 20 per cent. It was followed in 1999 by a further reduction of the standard by about 16 per cent on average for engines to be certified from the 31 December 2003.

The financial uncertainties pushed manufacturers to reduce their time to market from 5 years to 39 months for the Trent series [150]. In addition Rolls-Royce now plans to reduce its engine development time scale by a further 30% [2]. The increased competition in conjunction with the environmental concerns changed the market drivers which could be classified as follows [106]:

- Life cycle cost: acquisition, fuel burn, maintenance.
- Environmental impact: pollutants emissions, noise.
- Performance: thrust, weight, specific fuel consumption.

These drivers generated unprecedented pressures on engine design and in turn on combustor design to achieve higher targets in terms of design cycle time, performance, emissions and costs. It is therefore necessary to search means for making these targets easier to achieve.

1.1 Design Environment

During the last thirty years, the gradual shift of design methods from manual methods to methods based on analysis and simulation tools has allowed a dramatic reduction in development time and costs by giving more confidence in the quality of the design and therefore reducing testing requirements. The increasing complexity and performance requirements of engineering products necessitate more collaboration between the different departments, and more time for organisational and environmental tasks, which has a detrimental effect on the technical tasks [48]. This means that designers are faced with more technical decisions in the design phase than ever before, the number and complexity of these decision is growing rapidly while the time available to make them is decreasing. This increased complexity in conjunction with the ever reducing design cycle is a challenging task for the designer which would require means of automating part of the design process.

This is particularly true for gas turbine combustor design which involves a large number of inter-related complex phenomena, such as unsteady three dimensional flow, transport and evaporation of liquid droplets, gas phase mixing, complex chemistry, heat transfer and radiation. The complexity

of these inter-linked processes makes the derivation of any direct analytical design techniques extremely difficult.

Until the 1970s combustor design was considered more of an art rather than a science. It was mostly based on the "cut-and try" method where the approach was to repetitively test different variants of the combustor until a suitable arrangement could be found. It made use of a few empirically based design rules and relied on repeated testing to achieve a correct design. A study concerning gas turbine design costs in the 1970s highlighted that some 75% of all hardware costs were spent on those "cut-and-try" design feedback cycles [118].

Recent experimental and theoretical research on gas turbine combustors has resulted in better understanding of the physical processes taking place inside the combustor. This better understanding in conjunction with the powerful computational hardware made possible the development of numerical techniques capable of simulating, with a relatively high accuracy, most of the phenomena encountered inside a combustor. Based on these simulation tools, design techniques have been made more systematic and efficient.

The use of simulation tools, even though these are not perfect and still have serious deficiencies [2], resulted in a giant leap in terms of design capacity, costs and lead time. Martin Jones et all [83] reported more than 60% reduction of the direct operating costs, 70% increase in thrust to weight ratio, 90% reduction in soot, smoke, UHC, and CO, and 40% reduction of the NOx emissions due to the engineering advances in the last fifty years. In addition there was an amazing increase in reliability from more than 1 in 1,000 in-flight shut-down to 1 in 1,000,000 [83].

The aim is now to perform the redesign feedback loop on a numerical model of the combustor with testing only required to verify the design during the final steps of the design process. This was experimented with the Adour 915 design (Hawk retrofit program) where the combustor design was performed in a short time scale using numerical methods for the design and only 5 tests

were required compared to 40-50 for previous programs [2].

The simulation tools only give information on the particular design being simulated, but don't give any indication about the best design. It is up to the designer to perform the redesign cycle by manually modifying the design to find the best design point, using his past experience and trial and error. This type of performance tuning is still more of an art than a science and the optimisation levels highly depend on the designer skills and are inversely proportional to the complexity and the number of variable parameters of the design.

The difficulty of the optimisation task delays the progress in combustor design and hinders the achievement of the maximum performance point for a given technology and increases the time and cost to achieve a desired set performance target. It would be useful to automate the tuning of the design variables using optimisation techniques in order to support the designer in his task. However, the application of the traditional optimisation techniques is made practically impossible for this type of problems due to the large number of strongly correlated design variables with complex (non-derivable) relationships to a large number of conflicting performance targets.

However some unconventional optimisation techniques seems to be able to overcome the drawbacks of the traditional techniques. These techniques such as genetic algorithms or more generally evolutionary based techniques can accommodate the complex relationships between the design variables and are suitable for scaling up the number of variables.

1.2 The Emergence of Optimisation Techniques

During the early seventies optimisation techniques based on the theory of evolution were proposed. Those methods have the characteristic of being extremely robust and the ability to find global optima for a wide range of

problems without requiring any special knowledge of the actual problem. For this type of optimisation technique the problem is considered to be a black box, that is, these techniques do not require any information about the problem [69] unlike the analytical techniques which would usually require a known function that links the input parameters to performance parameters [86], in addition this function is usually required to be continuous and smooth [131]. The global optimisation capability of the evolutionary optimisation algorithms allows them to perform even in the presence of local optima where non global optimisation techniques would be trapped. Those two factors give a strong potential for those techniques to be applied to the optimisation of combustor design [36]. The fact that the problem is seen as a black box gives the possibility to use any simulation tool in conjunction with these optimisation techniques.

However the main drawback of these techniques lies in their relative computational inefficiency compared to analytical techniques. The lack of available computing power during the 70's meant that these methods had a limited applicability range. Recently, the advances in computing hardware combined with the development of more efficient genetic algorithms operators and techniques has allowed these techniques to reach maturity and to provide a wider range of applicability. Since then a number of successful applications of these techniques were demonstrated on hard engineering design problems such as aircraft wing profile optimisation [127], rotor systems [24], and many other problems [122, 167, 17].

The advances in these techniques allowed a good optimisation capacity to be demonstrated for high dimensional engineering optimisation problems [153, 152].

These advances combined with constraint handling techniques capable of tackling a large number of parameters allows the application of genetic algorithms based optimisation techniques to complex problems such as combustor design.

1.3 Aim of the Present Work

The use of optimisation techniques for combustor preliminary design could ease the pressure on the combustor designer by automating the optimisation of some performance parameters of the combustor, giving more time to the designer to concentrate on the technical tasks rather than tuning of the design. This will result in shorter design times and more refined optimisation at a reduced cost. The proof of concept for the application of optimisation algorithms to combustor design was discussed by Despierre, Stuttaford & Rubini [36]. This work provided the motivation and confidence to develop a complete approach to automated preliminary design tuning through evolutionary optimisation techniques.

The aim of this project is to propose and develop a design optimisation tool which will allow a novel approach to design and to apply it to the preliminary design of gas turbine combustors. This technique represents the first step towards autonomous engineering design with user-specified characteristics and objectives through the use of performance targets and constraints.

The proposed method consists of creating a modular 'toolbox', which regroup the necessary features to perform gas turbine preliminary design optimisation. These features are as follow:

- Ease of interfacing different analysis codes.
- Ability to deal with performance targets and constraints.
- Robust, non problem specific optimisation algorithms.
- Adaptation of these algorithms to engineering design.
- Tackle the computational cost of problem simulation.

The implementation of the tool box consisted in the development of a group of elements providing the functionalities previously cited, where the main

task was to create an efficient optimisation library capable of tackling a large number of target design parameters and constraints. This was achieved by implementing the state of the art techniques in the domain of genetic algorithms and developing some new ones such as a specific constraint handling method based on targets and the dynamic vectored mutation. Additionally, a simple NOx emission model was developed in order to complement the emissions simulation capability.

1.4 Outline of the Thesis

Firstly a detailed description of the gas turbine combustor preliminary design phase is presented in order to highlight its specific steps and and challenges. This is followed by a description of the available design methods and the design of the optimisation toolbox. Since the toolbox is based on genetic algorithms, basic theory behind the genetic algorithm optimisation technique will be presented in its traditional form. This will be followed by a description of the improvements made to the genetic algorithm library in order to increase its efficiency and adapt it as an engineering design tool. The results will be presented in two parts, the demonstration of the performance of the optimisation toolbox, and its application to combustor design optimisation. Finally proposals for future work will be presented and conclusions will be drawn from the present work.

Chapter 2

Combustor Preliminary Design

2.1 Introduction

It is important to have a clear idea of the combustor design process and of the details of preliminary design in order to be able to devise the best strategy to improve this design phase. In addition, since optimisation techniques are to be applied to the combustor design method, it is crucial to define a set of performance parameters that allows the performance of a combustor design to be measured. Therefore this chapter aims to describe the combustor preliminary design process and its integration within the complete combustor design process. As well, during this chapter, the different parameters controlling the design of a combustor will be identified and an attempt will be made to describe the relative importance of each parameter towards the quality of the combustor design.

During the previous chapter it has been highlighted that the technological drivers have changed in the recent years [106] and environmental issues have grown in importance up to a point where reducing pollutant emissions has become one of the main reason for technological change in combustion systems due to stricter legislation [74, 82]. In addition, the drive to reduce cost

has evolved from achieving low acquisition cost to the new goal of achieving low life-cycle cost [83]. These drivers in conjunction with the strong pressure to reduce development time have affected the design process [113], thus encouraging the development of accurate and more sophisticated simulation tools and radically changing the way combustors are designed.

In order to identify the key factors that can allow the improvement of the design process to achieve new levels of requirement, it is imperative to analyze different design phases of a gas turbine development project.

A combustor process is usually composed of four phases where the level of confidence in the design as well as its refinements increase as the design phases are completed. The costs will also increase steeply as the phases becomes more and more advanced being maximal for the testing and certification phase. These phases which follows a similar principle as the gas turbine design cycle [182] can be classified as follows: the conceptual study, the preliminary design, the detail design and finally the testing and certification, as in the sequence shown in Figure 2.1.

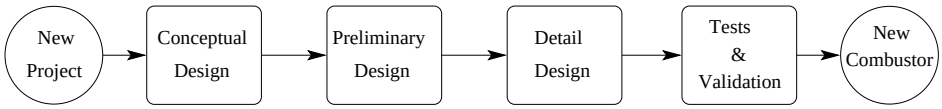

Figure 2.1: Design Cycle

- The conceptual study: This is the phase where the environment of the combustor is defined as well as the desired performance characteristics. Those are defined subjected to the turbine design, the customer requirements and legislative constraints. During this phase the design targets are identified, a combustor architecture is selected and the necessary R&D activities to achieve the targets are identified.

- The preliminary design: This Phase mainly consists of two tasks, the first one is to define the basic features of the combustor, and the second consists of tuning these features to achieve the desired combustor performance. After this phase variations in combustor requirements will not be permitted, the number of options selected during the conceptual study is reduced and critically analyzed in order to perform a first comparison between achievements and targets.

- The detailed design: This phase takes place once the combustor has been given its final shape and has been tuned during the preliminary design process. Its purpose is to use tools such as detailed CFD simulations to refine the points that can not be precisely designed due to the limitations of the preliminary design tools, to fine-tune some of the combustor features, and finally to highlight some potential problems. In addition to these thermo-fluid tasks it is also the place where the manufacturing objectives, tolerances, and local stress relieve features are fully defined. This phase gives an opportunity to solve some of the problems prior to the testing phase, thus reducing the duration of testing and the associated costs.

- The testing and certification: This is the validation phase of the combustor design, by verifying its performance. Since combustor design involves many phenomena which are not fully understood and that simulation tools are unable to predict correctly, some performances discrepancies exists between testing and simulation results. Therefore the testing phase often highlights some unexpected problems of the combustor design which were not apparent during simulation, thus requiring some modifications to the combustor. This is a costly phase and therefore every effort is made to limit the number of tests required by improving the simulation capabilities of the previous phases [2].

2.2 Background on Gas Turbine Combustor

The combustor is the component in the gas turbine where fuel is added to the airflow and subsequently burnt. In order to define the necessary features of combustors one has to look at a combustion chamber in its simplest form and then gradually increase its complexity.

The simplest combustor could be seen as a straight duct linking the compressor exit to the turbine entry, where fuel is injected as in figure 2.2a. This configuration is impractical because two problems arise due to the fact that compressor outlet velocities are relatively high (around 150 m/s). Firstly, the flow velocity is considerably above the flame speed of air-hydrocarbon flames. In addition the combustion induced pressure drop, which is proportional to the square of the air velocity in this configuration, would be unacceptable even if a flame could be sustained at these speed [95].

In order to reduce these problems it is possible to fit a diffuser between the compressor exit and the start of the combustion section to reduce the airstream velocity as in figure 2.2b. This allows the air velocity to be reduced by a factor of about 2 to 3, thus reducing the combustion induced pressure drop to acceptable limits. However it would still be difficult to maintain an attached flame in these conditions.

As a simple modification a baffle can be added creating a flow reversal which will generate a low velocity region where the flame can stay attached as in figure 2.2c. This configuration would allow the flame to stay attached and prevent it from being blown away. Configuration 2.2c is acceptable as a viable system and is used for re-heat systems in military engines. But as a gas turbine combustor, further improvements are required before it can be put into practice. Since in this configuration all the air arriving at the combustor takes part in the combustion process, a large amount of fuel would be required to maintain the mixture within its flamability limit, this would then result in high combustor exit temperature much higher than what is

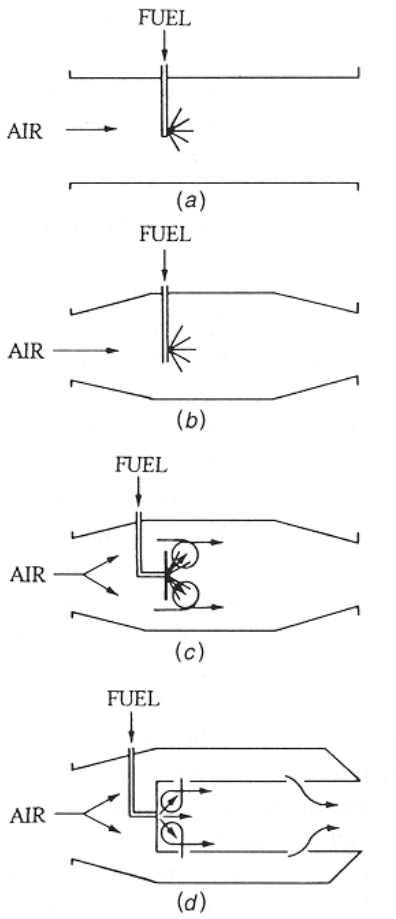

Figure 2.2: Derivation of the conventional combustor configuration (Lefebvre [95])

acceptable for the turbine.

This problem can be solved by using a perforated liner which allows the air to be added gradually as in figure 2.2d. This configuration allows the air / fuel ratio (AFR) to be controlled precisely in the flame region, called the primary zone. The remaining air can then be added after this zone to dilute the combustion gases and reduce the temperature to acceptable levels for the turbine. Furthermore adding air after the primary zone strengthen the recirculation which helps to anchor the flame. This delayed air addition can be further beneficial because it can be used to control the pollutant emission by choosing the air fuel ratio in different zones of the combustor. In practice smaller perforations are added to the liner to provide cooling of the liner walls. This configuration can be regarded as a basic combustor configuration.

In addition to this basic configuration some devices can be added to improve the performance of the combustor; an air swirler is often added to create a strong vortex motion to have a fixed recirculation zone detached from the baffle. Fuel is usually injected not as a simple spray but through an air blast atomizer which integrate the swirler in the atomizer thus forming finer droplets for better combustion efficiency as in figure 2.3. This results in the typical combustor configuration as observed in figure 2.4.

For a better control of pollutant formation, some alternative configurations are thought to have better performance over the whole operating range of the combustor. One such technique is to separate the flame region for the low power conditions and the high power conditions. This can be achieved either by selecting the double annular configuration shown in figure 2.5 or the staged combustion configuration represented in figure 2.6. These configurations permits stable operation of a rich pilot zone which maintains the combustion during idle and low power conditions and an efficient lean main zone reducing NOx emissions during high power conditions.

Aero gas turbines are usually fitted with straight flow combustors with an annular configuration as in figure 2.7. Other variants have also been used in

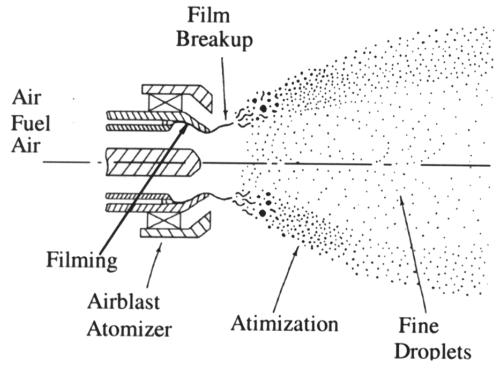

Figure 2.3: Air Blast Atomizer (Malecki [102])

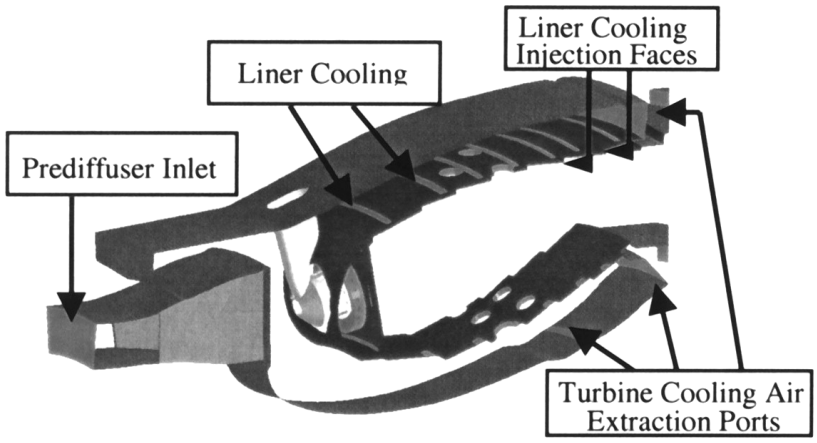

Figure 2.4: A typical Combustor Configuration (PW4098 Malecki [102])

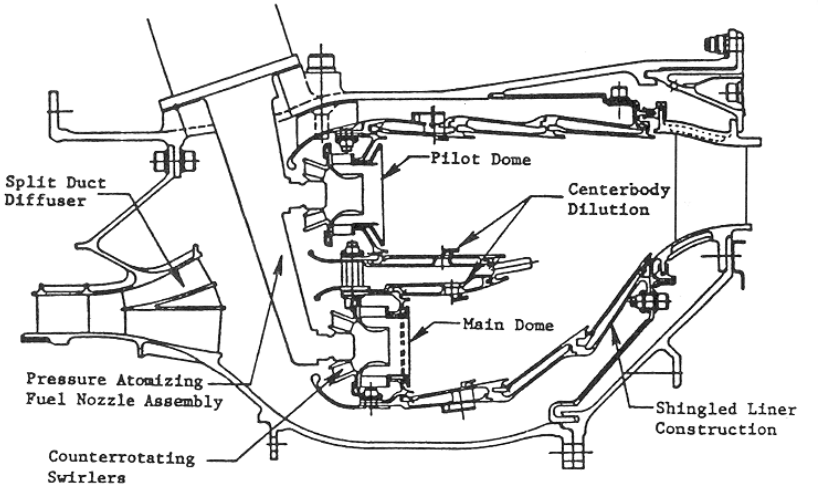

Figure 2.5: Dual Annular Combustor (Lefebvre [95])

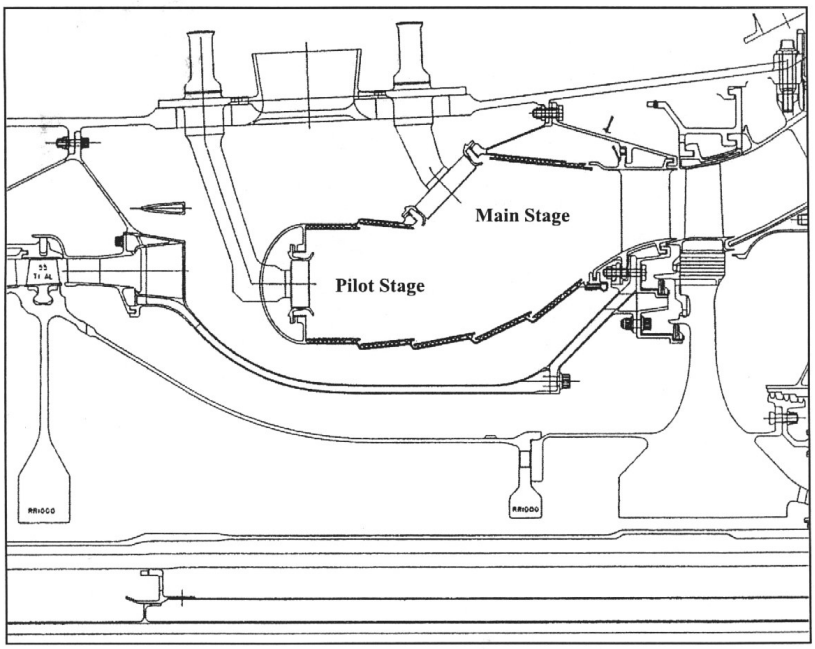

Figure 2.6: Staged Combustor ([73])

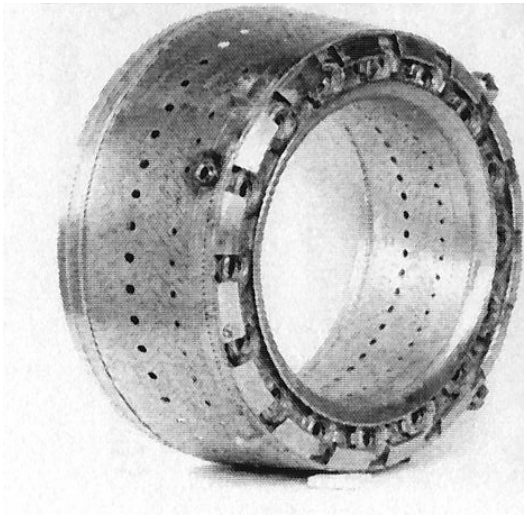

Figure 2.7: Annular GE Combustor (Mongia [113])

the past namely tubular or tuboannular but have mostly been abandoned due to their weight and pressure drop penalty. In some cases, when the distance between the compressor exit and the turbine entry is small it is necessary to use reverse flow combustor configuration as shown in figure 2.8.

2.2.1 Performance Requirements

A combustor must satisfy a wide range of requirements whose relative importance may vary depending on the engine type and specific application. The basic performances requirements are stated below:

- Pollutant Emissions: The pollutant emissions of a combustor should be minimal, where pollutant can be either particulates, NOx, CO, UHC, Soot, Smoke or SOx, generated by the combustion process. Emission of those products are legally limited for civil aero-engines by the ICAO regulations [74] which are becoming stricter. In addition some countries such as Switzerland and Sweden now impose emissions related landing

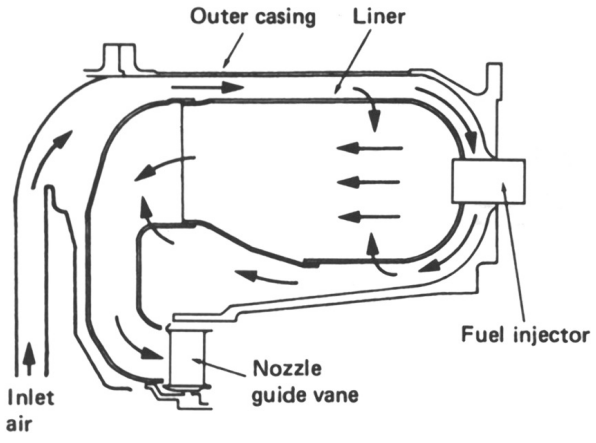

Figure 2.8: Reverse Flow Combustor (Lefebvre [95])

charges [187, 82]. These regulations and penalties provide additional incentive for gas turbine manufacturers to achieve significantly lower emissions than the legal limit to increase the competitiveness of their engines [147, 83, 63, 14, 176]. The problem is similar for industrial gas turbines where pollutant emissions regulations in some part of the world are even stricter. For military engines the problem is centered on particulate emissions such as soot or smoke which are detrimental to the low observability requirements of modern military aircraft. There NOx emissions will be of lesser importance, however these might become an issue if oxidised to NO_2 which is visible.

- Pressure Drop: The pressure drop within the combustor should be minimal to allow maximum performance of the gas turbine. However some pressure drop is required to obtain proper mixing of the burning gases within the combustor and to drive the cooling flow of the nozzle guide vanes. A compromise has to be made between an acceptable pressure drop level and an acceptable combustion performance and NGV cooling.

- Wall Cooling Flow: The wall cooling flow is taken from the liner flow. Reducing it would be beneficial because it leaves extra dilution air that can be used either to operate leaner or to generate better exit temperature profile. Reducing the wall cooling flow may also reduce the pollutant formation zones due to quenching in the regions where cooling flow is added [106]. However the reduction of the wall cooling flow may have an adverse effect by increasing the likelihood of combustion instabilities. The numerous holes used to distribute the cooling flow act as a very efficient damper of pressure pulsations. As the cooling requirements of combustors reduces due to improved materials and cooling techniques the damping effect of the cooling orifices is reduced [173]. Thus making the combustor more susceptible to combustion instability problems.

- Re-light: The combustor should be able to re-light reliably at a given altitude. The definition of the relight envelope is negotiated between the airframe manufacturer and the gas-turbine manufacturer and legislation demands the verification of this relight envelope [25]. Engine manufacturers generally aim to achieve a relatively large altitude-relight envelope to have a competitive advantage.

- Fluctuation: The combustor should be free of pressure pulsations and combustion instabilities. Existence of these will generate cyclic loading on the critical combustor and turbine components reducing their life through fatigue. In addition the noise generated can also be a source of nuisance to pilots and passengers.

- Efficiency: The combustion efficiency should be maximal so that all the chemical energy is converted into heat. This factor is important in order to minimize the specific fuel consumption of the engine.

- Stability Limits: The combustor should be able to operate within a wide range of pressure, AFR, velocities, which covers the whole operating

range of the gas turbine including windmilling and relight.

- Cost: The costs for the design, manufacturing, and maintenance of the combustor should be minimal. Additionally manufacturers are under pressure from customers who are expecting a low-cost of ownership of the engine. The cost of ownership is linked to the manufacturing costs, efficiency, maintenance, durability and pollutant emissions in the case where emission based landing fees are applied.

- Durability: The life of the combustor should be compatible with the mission of the engine and the intervals between necessary repairs on the combustor should be maximal, combustor should not be the life limiting component requiring overhaul. Civil applications usually require a very high durability with minimal maintenance while military applications favor a tradeoff more biased towards performance rather than durability.

2.2.2 Constraints from the Gas Turbine

The gas turbine, due to its size, shape, technology and operating conditions, does impose constraints on the combustor.

The constraints are in terms of:

- Mass Flow: The operating conditions of a gas turbine covers a wide range of mass-flow rates. The combustor should be designed to work efficiently throughout the whole mass-flow range.

- Length: The maximum length available between the compressor exit and the turbine entry is imposed by the shaft design to limit the effect of shaft torsion and vibrations. This length constraint will have an important effect on the shaping of the combustor. If the available length is large there is space to fit a good aerodynamic diffuser. Unfortunately

in most of the cases the space is limited forcing the designer to use the shorter dump diffuser and very short combustion zones. On the very small engines where the available space is even smaller it becomes necessary to use reverse flow combustors.

- Turbine Entry Temperature T_4: The Turbine Entry Temperature (TET) is fixed by the cycle requirement and is limited by the turbine design (blade cooling technique, material used). The gases exiting the combustor are directly fed to the turbine therefore the temperature of those gases must meet the required turbine entry temperature.

- Exit Temperature Traverse: To guarantee its expected life, the turbine needs to be given a defined temperature traverse. The temperature traverse or temperature profile is generated within the combustor by the non-perfect mixing of the cold cooling port flow with the hot core flow. The combustor temperature traverse should be as close as possible to the optimal temperature traverse (defined by the turbine cooling capability and stress profile) to achieve maximum turbine life.

- Operating Pressure P_3: As well as for the mass-flow the engine experience a very wide range of operating pressure. The combustor should be designed to work efficiently within this whole range of pressure, specifically for re-light conditions where the pressure is at its lowest and ignition becomes difficult.

- Inlet Temperature T_3: The combustor inlet temperature is directly defined by the engine pressure ratio of the compressor and inlet air temperature. It will vary with the operating pressure, at the highest pressure ratio the inlet temperature will be at its maximum. It will have an effect on the maximum fuel burn and since it is the inlet air (compressor delivery air) that is used as cooling flow, an increase of inlet temperature will have a significant effect on the cooling performance and pollutant emissions.

2.3 Combustor Preliminary Design Process

This section aims to briefly describe the preliminary design phase of a combustor. The preliminary design consists in finalizing the shape and tuning the basic features of the combustor to suit the gas turbine constraints and the performance requirements. It is where the combustor takes its final shape, and its main features, which are then tuned to achieve the required performance. The combustor preliminary design is currently performed using low dimensional empirical or semi-empirical tools. This implies that the preliminary design tools should be capable of modeling the global parameters of the combustor. However these tools do not have the capacity to pick-up the effects of local details.

The combustor preliminary design is originated from the combustor requirements which were defined during the conceptual phase; these requirements dictate the technology used for the design of the combustor and the selection of its main features. The preliminary design could be sub-divided in to four phases as follows:

- Phase 1: Freezing of the combustor architecture.
- Phase 2: Design of the diffuser.
- Phase 3: Selection of the Combustor Features.
- Phase 4: Tuning of the combustor Feature.

Those four phases can are algorithmically presented in figure 2.9, which shows a typical preliminary design cycle. Those phases will be described in more detail in the following sections. It is a design feed back process, with a strong feed back loop over the fourth phase, the design processes might in some cases loop back to the third phase and more rarely to the first phase.

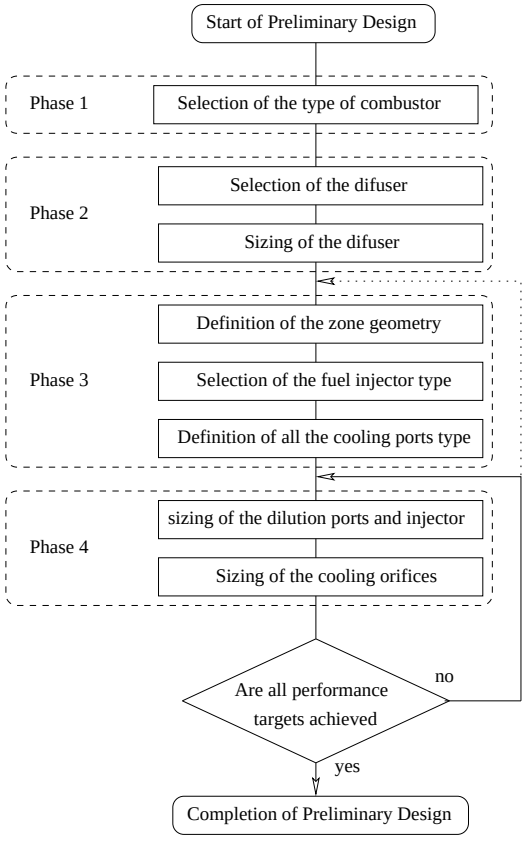

Figure 2.9: Preliminary Design Flow Chart

2.3.1 Combustor Architecture

The architecture of the combustor is selected according to the requirements described in section 2.2.2and 2.2.1. Usually the combustor preliminary design does not start from a blank sheet of paper, but from an existing combustor design that needs to be sized and shaped to suit its new application. Therefore the selection of the combustor architecture often means adapting an existing design for a new gas turbine design, with some modifications to improve its performance.

- The engine envelope constraints, the choice of the combustor and the diffuser type, (for the combustor: straight or reverse flow and for the diffuser: aerodynamic, dump or vortex controlled diffuser).

- The relight capability requirements will constrain the minimum volume of the primary zone.

- The cooling capacity of the air used for cooling is proportional to the compressor delivery temperature (T3). Therefore it affects the selection of the combustor wall material and cooling techniques together with the balance of life/cost/weight.

- The emission constraints can lead to different combustor geometries such as double annular combustor with a pilot and a main combustion zone and different types of combustion (premixed or diffusion).

Once these considerations have been taken into account, the basic geometry is frozen and the combustion zones are defined, fixing the location of the dilution ports. After the definition of the combustion zones, the cooling orifices can be selected and positioned.

2.3.2 Selection of the Combustor Features

This section describes the selection of the main features of the combustor that have to be defined during the preliminary design. Since this work concentrates on the tuning of the combustor parameters, the features will be selected prior to the use of the optimisation tool, however it is important to have an overview of the different technologies available.

2.3.2.1 Fuel Injection system selection

The choice of the fuel injection system has major implications for the combustor design and especially the emissions. These could be classified into three groups:

- Pressure Atomizer: These injectors rely on the injection of fuel at high pressure to generate a fine droplet spray. The simplest type corresponds to a simple orifice. Fuel is atomized by passing through a small circular hole, but the most common is the dual orifice atomizer. It is constructed by stacking two swirl atomizers; a smaller primary atomizer is fitted inside the main atomizer allowing good atomization over a wide range of fuel flows. However, these type of injectors require high fuel pressure and tend to create a high quantity of soot during high pressure operation.

- Twin Fluid Atomizer: These injectors rely on the use of high pressure air to improve the atomization process. High pressure ratio engines tend to use prefilming airblast injectors where the fuel is first spread onto a continuous sheet and then subjected to the atomizing action of high velocity high swirl air as in figure 2.10. These injectors exhibit a very high atomization quality, a low soot formation and do not require the high fuel pressures of the pressure atomizers. However they can suffer from a narrow burning range and a poor atomization quality at

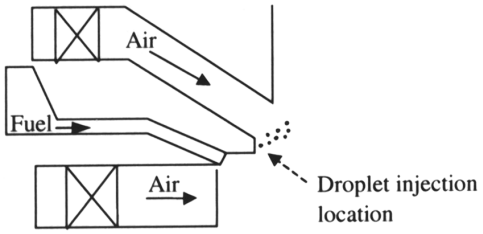

Figure 2.10: Schema of an Airblast Atomizer (Malecki [102])

low air velocities, for example at startup. In addition, since the fuel is fed at a low pressure this allows the potential for acoustic feedback through fuel flowrate variations, making the combustor more subject to combustion instabilities.

- Vaporizer: The aim of the vaporizer is to heat the fuel above the boiling point of its heaviest hydrocarbon ingredient. The simplest method consists of injecting the fuel with some air in a tube immersed in the flame. These are low cost injection systems, allowing low soot formation. However, they require special starting systems and suffer from the risk of thermal damage making them mechanically suspect. In addition these dose not guaranty a fully vaporized flow. The system that seems the most promising for low pollutant emission is the lean premix prevaporize system (LPP) which provides the combustion zone with a homogeneous fuel air mixture. They exhibit very low NOx, Soot or smoke and generate a constant pattern factor. However these are susceptible to flashback or auto-ignition. In addition since they operate close to the lean blowout limit they have a narrow stability region and hence are especially prone to combustion instabilities.

2.3.2.2 Liner Cooling Technologies

The combustor liner is subjected to high thermal loads and cooling is usually necessary to guarantee the required operating life. There are several options to maintain the temperature of the liner wall, which could be classified in three types as follows:

- Film Cooling: These cooling devices create a thin film of cold air on the flame side of the combustor wall, preventing the hot air from the combustion to be in direct contact with the wall. There are a wide variety of these cooling devices as show in figure 2.11 and 2.12. Although it is a relatively simple and cheap cooling method, it requires a large amount of cooling air and it is not capable of maintaining an uniform wall temperature.

- Augmented Convection Film Cooling: The cooling requirements can be reduced by augmenting the convection cooling of the walls. Figure 2.13 shows different methods for improving the cooling of the liner wall using convective effects. The most interesting of these techniques for practical purpose might be effusion cooling (figure2.13d) which exhibits high cooling efficiency especially when using angled holes. Laser drilling has allowed relatively cheap manufacturing of the effusion holes. However this arrangement requires thicker walls and there are some issues with regard to repairability.

- Thermal Protection: An effective solution to reduce wall temperature is to reduce the heat transmission to the wall, either through an interposition of refractory tiles, or through the use of a thin layer of low emissivity, low thermal conductivity refractory material. The first method provide an excellent protection of the liner wall, however it contributes to a substantial increase in weight. The second method can reflect a large part of the incident gas the radiation and allows a

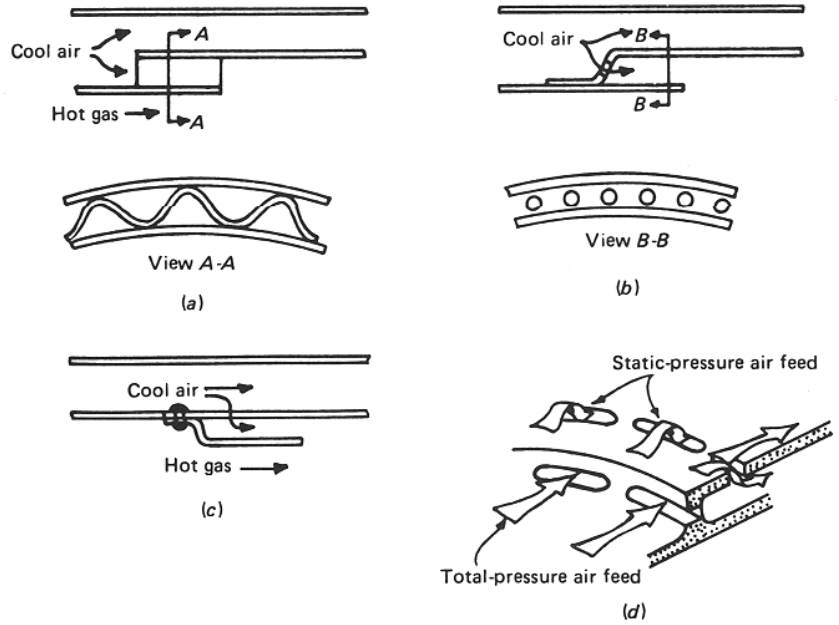

Figure 2.11: Film Cooling Devices 1: (a) wiggle-strip, (b) stacked ring, (c) splash-Cooling, (d) machined ring. (Lefebvre [95])

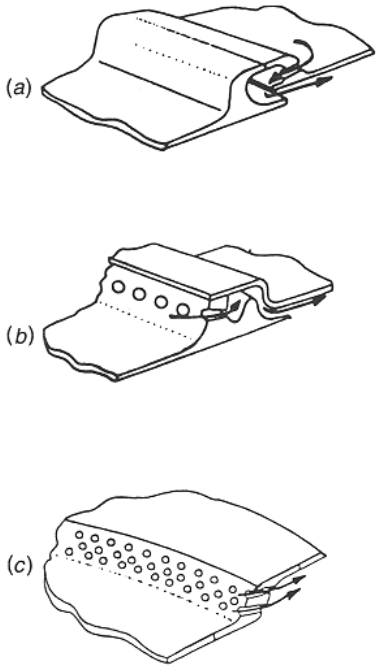

Figure 2.12: Film Cooling Devices 2: (a) rolled ring, (b) double-pass ring, (c) Z-ring. (Lefebvre [95])

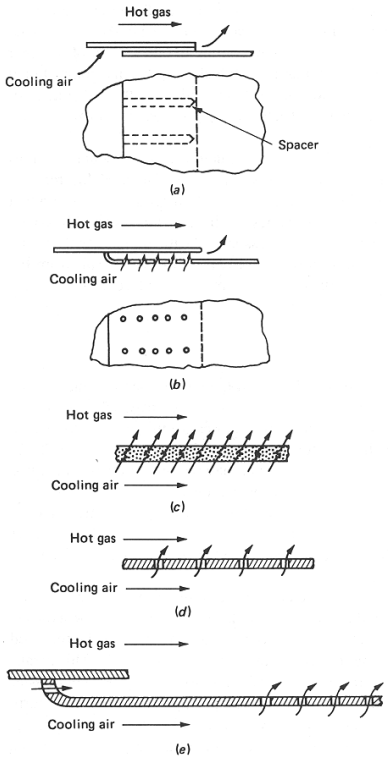

Figure 2.13: Augmented Convection Film Cooling Devices : (a) combine convection and film cooling, (b) combined impingement and film cooling, (c) transpiration cooling, (d) effusion cooling, (e) combined film and effusion cooling. (Lefebvre [95])

reduction of wall temperature of 40 to 70K. Thermal barrier coatings allow to dramatically reduce the cooling requirement, however it does not suppress it. Usually it is used in conjunction with effusion cooling patches or external convection cooling patches.

2.3.3 Tuning of the Combustor Features

Now that a rough picture of the combustor has been sketched, the features of the combustor are then optimised to achieve the performance requirements.

- The swirler strength and the primary port size are varied to achieve the desired recirculation flow in the primary zone.
- The dilution ports are varied to achieve the desired zone AFR to limit the emissions level, proper mixing of the the port flows with the combustion zone, and the desired outlet temperature profile.
- The cooling orifices are positioned and sized to minimize the amount of cooling flow while keeping the combustor liner below its maximum allowable temperature and avoiding hot spots.

It is important to note that theses parameters are closely linked and interact among each other. These choices are made using empirical rules and past experience. A set of fast simulation tools and correlations are used to assess the performance of the design. These tools are usually based on 1D semi-empirical codes.

This process of varying the features of the combustor and simulating its performance is repeated many times until the desired performance is achieved. During the preliminary design phase, only a limited use of CFD is made due to its computational cost compared to empirical codes.

During this phase of the design it is sometimes necessary to go back and modify some basic features of the combustor and then restart the process.

Once all the performance targets are met and all the constraints satisfied the preliminary design phase is completed. It is followed by the detailed design phase were the design is finalized using refined CFD simulations and testing to validate the design.

This preliminary design process was greatly improved along the years due to a better understanding of combustion processes and more accurate simulation tools. However it is still a long and costly process. It takes between 3 to 6 month for a team of combustion engineers to complete the preliminary design of a combustor. During this time other components of the engine may also be evolving and the combustor requirements may change as a result.

2.4 Combustor Performance Parameters

The design of the combustor is performed in order to satisfy the constraints imposed by the gas turbine 2.2.2 and the performance requirements of the combustor 2.2.1. Those constraints and requirement can be expressed in terms of performance parameters 2.4.1 that will define the quality of the combustor design.

2.4.1 The Performance Parameters of a Combustor

This section describe the various performance parameters used to define the quality of a combustor design. Some of the parameters describing the performance requirements, as described in section 2.2.1, may be difficult to quantify or may not be available at all during the preliminary design. Therefore use is made of a number of alternate performance parameters to constrain or approximate the ones that can not be calculated directly.

2.4.1.1 Average Zone Air Fuel Ratio

The average zone AFR corresponds to the ratio of total air Mass flow to fuel Mass flow in the given zone (equation 2.1). It is one of the crucial parameters controlling the combustion. The combustion temperature and hence pollutant emissions is strongly dependent on the AFR, therefore it should be controlled precisely throughout the whole combustion zone. However this zone averaged AFR does not constrain local AFR values. The local AFR distribution depends largely on the mixing. Therefore constraining the mixing will put a constraint on local AFR distribution.

$$AFR_{avg} = \frac{\dot{m}_{air}}{\dot{m}_{fuel}} \tag{2.1}$$

2.4.1.2 Mixing Quality

The mixing quality defines how air, fuel and combustion products are intermixed, and influences the local flame temperature. It is an important parameter governing the pollutant emissions as well as the combustor exit temperature traverse. The mixing should be constrained, however it is not possible to quantify the quality of the mixing with the preliminary design tools. Therefore alternative parameters that control the mixing have to be used. For this purpose, the following three parameters could be used; the pressure drop, the cross flow Mach number ratio which constrains the penetration of the port flow, and the amount of recirculating flow which constrains the primary zone mixing.

2.4.1.3 Pressure Drop

During the design of a combustor four different ratios of pressure drop are taken into consideration:

The overall pressure drop, expressed in 2.2, which defines the total pressure drop between the compressor delivery and the turbine entry. its value is usually fixed during the definition of the engine cycle.

$$\Delta P_{overall} = \frac{P_3 - P_4}{P_3} \tag{2.2}$$

The diffuser Pressure Drop expressed in 2.3 measures the pressure loss during pressure recovery process, it is a pure loss and therefore should be minimized.

$$\Delta P_{diffuser} = \frac{P_3 - P_{diffuser\,outlet}}{P_3} \tag{2.3}$$

The flame tube pressure drops defined in 2.4 and in 2.5 for the outer and inner walls. They represent the pressure difference between the liner flow and the flame-tube flow, and affects the energy in the port and cooling flows. The value of flame-tube pressure drop will affect the mixing in the flame-tube.

$$\Delta P_{ow} = \frac{P_{ow} - P_4}{P_{ow}} \tag{2.4}$$

$$\Delta P_{iw} = \frac{P_{iw} - P_4}{P_{iw}} \tag{2.5}$$

2.4.1.4 Cross-Flow Mach Number Ratio

The cross-flow Mach number ratio is defined as the ratio between the Mach number of the port flow at the location of the hole and the Mach number of the annulus flow, and is given in equation 2.6. The Cross-flow Mach ratio value is defined for the inner and outer primary and secondary ports. These parameters control the mixing and the temperature traverse.

$$Mach\ Ratio = \frac{M_{po}}{M_{an}} \tag{2.6}$$

2.4.1.5 Recirculating Flow

The amount of recirculating flow in the primary zone defines the quality of the mixing in the primary zone and influences the relight loading. The amount of recirculating flow can be estimated using the following rule of thumb:

$$\dot{m}_r = \sum \dot{m}_{bp} + \frac{1}{2}\sum \dot{m}_{po} + \frac{1}{3}\sum \dot{m}_{cool} \tag{2.7}$$

Even-though this representation is over simplistic and not representative for modern combustor with a strong swirl it is often used in the preliminary stage.

2.4.1.6 Overall Loading

The overall loading parameter is a measure of the efficiency of the combustor. Lefebvre in [90] expressed the combustion efficiency:

$$\eta_c = f\,(Air\ flow)^{-1}\left(\frac{1}{Evaporation\,rate} + \frac{1}{Mixing\,rate} + \frac{1}{Reaction\,rate}\right)^{-1} \tag{2.8}$$

In practical situations, equation 2.8 will be governed by either the evaporation, mixing, or the reaction rate. this led Greenhough and Lefebvre [57] to the definition of the θ parameter which can be expressed in the form 2.9.

$$\eta_\theta = f(\theta) = f\left[\frac{(P_3^{1.75}V_c)\,e^{(T_3/300)}}{\dot{m}_A}\right] \tag{2.9}$$

Note: Of the three terms controlling 2.8 only the reaction rate is directly dependent of the combustion volume leading to 2.9 being used to define the required combustion volume for a high enough efficiency for the engine to pull away after relight.

The overall loading 2.10 can be defined as function of the inverse of 2.9, and is usually expressed in the form given in 2.11 [98]. Where a,b, and c are

proprietary constants.

$$\Lambda_{SI} = f\left(\frac{1}{\eta_\theta}\right) \tag{2.10}$$

$$\Lambda_{SI} = \frac{\dot{m}_3 * a}{\left(P_3^b V_c\right) e^{T_3/c}} \tag{2.11}$$

2.4.1.7 Relight Loading

The relight loading parameter controls to the capability of the combustor to relight at a given altitude and flight condition. It is a function of the amount of recirculating flow and primary zone volume. The relight loading is expressed in equation 2.12 [99]. Here a,b, and c are proprietary constants, and $P_{3R}$and T_{3R} are $P_3$and T_3 at relight conditions.

$$\chi_{SI} = \frac{\dot{m}_r * a}{\left(P_{3R}^b V_{pz}\right) e^{T_{3R}/c}} \tag{2.12}$$

It is interesting to note that the relight and the si loading are of the same form except that for the relight loading only the recirculating flow and the primary zone volume are considered.

2.4.1.8 Cooling Mass Flow Ratio

The ratio of cooling mass flow defines the quantity of the combustor flow used for cooling the wall of the combustor over the total air mass flow of the combustor. It is of the form given in equation 2.13. The cooling flow should be minimized. leaving more flow to improve the AFR and exit temperature profile. In addition, the cold zones created by the cooling flow are strong pollutant formation zones through quenching and therefore reducing the coolant mass flow allows to decrease the pollutants formation zone [106].

$$U = \frac{\sum \dot{m}_{cooling}}{\dot{m}_4 - \dot{m}_{fuel}} \tag{2.13}$$

2.4.1.9 NOx Emissions

It is relatively difficult to describe the formation of pollutant emissions with out a detailed analysis, the process is complex and strongly dependent on local conditions. However within a family of designs these local condition will all be correlated together hence allowing simple correlations to be used, the reliability of which vary depending on the difference between the current design and combustors used to create the correlations. These will give a good indication of the level of emissions but more advanced emissions calculation techniques should be used as a predictive tool.

There is a wide range of correlations for combustor NOx emissions predictions [95, 92, 149, 10]. The correlations usually share some common basis, but depend upon different physical parameters. The Lefebvre [92] NOx correlation is not one of the most recent but has been well tested on a wide range of combustor applications.

$$NOx_{EI} = \frac{A_{NOx} P_3^{1.25} V_c e^{(0.01 T_{st})}}{\dot{m}_A T_{pz}} \quad (g/kg) \tag{2.14}$$

It is interesting to note as well the correlation proposed by Rizk and Mongia [149] which has the particularity to take into account the quality of the mixing through the presence of the ΔP term and the evaporation time t_e.

$$NOx_{EI} = 15 * 10^{14} \left(t - 0.5 t_e\right) e^{(71{,}100/T_{st})} P_3^{-0.05} \left(\frac{\Delta P}{P_3}\right)^{-0.5} \quad (g/kg) \tag{2.15}$$

2.4.1.10 CO emissions

The same comments can be made for the CO correlation as for the NOx emissions. However CO formation follows a different process. In a first step high quantities of CO are formed extremely rapidly in the burning zone. In

a second step Most of this CO will be later destroyed if enough oxygen and time are available.

The destruction of CO happens at a relatively longer time scale, therefore the relevant temperature is not the peak temperature as for NOx formation, but the average zone temperature. Rizk and Mongia [149] developed the following correlation.

$$CO_{EI} = 0.18 * 10^9 \frac{e^{(7800/T_{pz})}}{P^2 \left(t - 0.4t_e\right) \left(\frac{\Delta P}{P}\right)^{0.5}} \quad (g/kg) \tag{2.16}$$

2.4.1.11 Soot Emission

Some correlations do exist for the prediction of soot but do not seem to provide reliable results as those for NOx or CO. This is because soot is produced in large quantities, 99.9% of which is oxidised by the exit in a well designed combustor, so the resulting emissions are a balance between two big numbers (formation - oxidation) .

2.4.1.12 Combustion Instabilities

Combustion instabilities originate due to cyclic variation of heat release resulting from the excitation of the combustor natural frequencies by the combustion noise. This is a difficult problem for combustor design, since there are no simple models to predict the occurrence of combustion instabilities. It can be said that combustors equipped with low pressure injection systems tends to suffer more from instabilities. In addition, the reduction of the sources of damping, such as cooling orifices tend to increase the likelihood of instabilities [173].

2.4.1.13 Pattern Factor

In order to control the temperature traverse, the pattern factor can be used. The pattern factor is defined as the ratio of the maximum recorded temperature over the turbine entry temperature T_4. It is defined in equation 2.17.

$$\frac{T_{max} - T_4}{T_4 - T_3} \tag{2.17}$$

Lefebvre in [95] proposed a correlation for the pattern factor of annular combustors.

$$\frac{T_{max} - T_4}{T_4 - T_3} = 1 - exp\left(-0.050 \frac{L_L}{D_L} \frac{\Delta P_L}{q_{ref}}\right)^{-1} \tag{2.18}$$

Due to it's simplicity this expression is not universally applicable and is limited to the estimation of T_{max}. However it shows that for a fixed geometry 2.18 is only dependent of ΔP_L. It can therefore be deducted that controlling ΔP_L will contribute to control the pattern factor which may not be known during the preliminary design phase.

Since during the tuning phase of the combustor the liner length L_L and the liner height D_L are be fixed as well as q_{ref}, the pattern factor solely depend of the liner pressure drop ΔP_L. Therefore the Liner pressure drop can be used to control the pattern factor.

2.4.1.14 Wall Temperature

The temperature of the combustor liner walls will affect their installed life therefore it needs to be controlled. In terms of maximum average temperature for problems of cycling fatigue. In terms of maximum local temperature and maximum thermal gradients for problems of thermal stress.

2.5 Conclusion

During this chapter the combustor preliminary design process was presented. First some background is given on gas-turbine combustor describing the basic design principle of a combustor and its requirements. Then a brief discussion is provided on the selection of the architecture and the features. Finally the emphasis is put on the identification of the key parameters which define the performance of a combustor. These parameters will be critical to drive the optimisation process described in the following chapters.

Chapter 3

Design Optimisation

3.1 Introduction

This Chapter is oriented towards design optimisation techniques and their implementation. The first section, aims to introduce the reader to the concepts of design optimisation, the means of determining the quality of the design and the means of modifying a design. This will be followed by a presentation of the application of design optimisation in industry and the description of the specific aspects regarding application of design optimisation to gas turbine preliminary design, which highlights the needs for an efficient optimisation algorithm and for a complete set of tools that allows integration of the design optimisation in the design process. Therefore possible optimisation techniques will be presented and their relative strength and weaknesses will be discussed. Finally the conception of the optimisation Toolbox will be described. The optimisation Toolbox was designed to allow the integration of the optimisation algorithms into the combustor design process and it is still flexible enough to allow its use on other design processes.

3.2 Introduction to Design Optimisation

As it has been highlighted in the previous chapter, during the preliminary design phase of a combustor the goal of a designer can be defined as the achievement of a set of performance targets and constraints. In order to successfully achieve this set of targets the designer has to define the shape of the combustor, select its features, and tune all the parameters controlling the performance of the combustor. The tuning phase is relatively time consuming and mostly involves trial and error methods to define the combustor configuration that offers the best performance parameters, and satisfies the design constraints.

In order to relieve some of the work load from the designer, three options can be suggested to improve the design process:

- Simplify the design process by developing reliable design rules that allow the definition of the optimum parameters of the combustor. This route has been attempted since the birth of combustor design, leading to a number of empirical design rules. One example could be the work done by Murthy [115] who attempted to regroup a set of empirical design methods with some analytical techniques. However these design rules are derived from experimental data and are only valid for designs similar to the one used to derive them. In addition, the inherent complexity of combustor design makes the creation of a general set of design rules extremely difficult.

- Regroup all the correlations and empirical design rules and their known limitations in a expert system which would be a support tool for the designer. This tool will then be capable of suggesting the applicable design rules depending on the specific case. Although this tool might help to simplify the designer's work, it does not provide any help for novel design, and even for traditional designs, the set of existing design

rules supports the design process but does not give the the complete set of optimum design parameters required for a combustor design. Therefore the designer would still have to resort to trial and error to tune the design.

- Use an optimisation technique to automate the trial and error process by automatically searching for the optimum design parameters of the combustor. This process is usually referred as the design optimisation technique. This method allows the tuning of the numerous design variables until the the required performance parameters are obtained, thus releasing the designer from this painstaking task. The main drawback of this method is the selection of a suitable optimisation technique capable of dealing with multi dimensional, nonlinear, complex problems. In addition the selection of the criteria defining the quality of a proposed design (and hence it's fitness) is very delicate and crucial for the optimisation process.

Of these three options, the most promising technique seems to be the automation of the parameter tuning by making use of design optimisation techniques. The selection of a suitable optimisation algorithm, which is robust and efficient, is a difficult task. This route, however, provides many advantages, and has the potential to be used as a completely automated design method with user specified constraints and targets. In addition, the fact that the optimiser is free of any empirical design rules might allow the new design to be free of the "combustor family effect" that can be observed in current designs. Freeing the designs from the family effect will give the opportunity to search for the overall best design and not for the family's best.

The following sections present the concept of design optimisation (DO), a review of the potential optimisation techniques, and finally a description of the Optimisation Toolbox which represent the framework of the proposed combustor preliminary design method, and its design methodology.

3.3 Concept of Design Optimisation

From the early days of engineering, the goal has been to improve designs in order to generate the best possible solution with the available means. Engineering design could be viewed as a process which aims the optimisation of an existing design. A close inspection of the combustor design cycle (figure 2.1), shows that each phase of the design can be sub-dived to an iterative design cycle as presented by Roozenburg and Eekels [154] in figure 3.1. Every time new design proposals are generated, their properties are compared with the desired design criteria. If the properties meet the criteria then the design is accepted. Otherwise, the design is modified and reevaluated. This iterative process will be continued until the generation of a design that meets all the specified criteria not only in terms of performance but as well in terms of cost, manufacturability, and life. In practice several candidate designs will be generated and evaluated in parallel. This provides competition between the different designs.

Design optimisation allows to reformulate the design problem into an optimisation problem [166]. This process (figure 3.1) closely resembles to an optimisation process, as shown in figure 3.2, and reformulation of the design problem into a design optimisation process (figure 3.3) is therefore reasonably straight forward. However two important design aspects need to be formally defined:

- The quality of the design which is a measure of how good the design is and how well it satisfies the design criteria (cf. section 3.3.1).
- The design variables which are the parameters that allow the modification of the design (cf. section 3.3.2).

Once these two aspects have been carefully defined, it is possible to use an optimisation method that modify the design variables in the aim of creating a

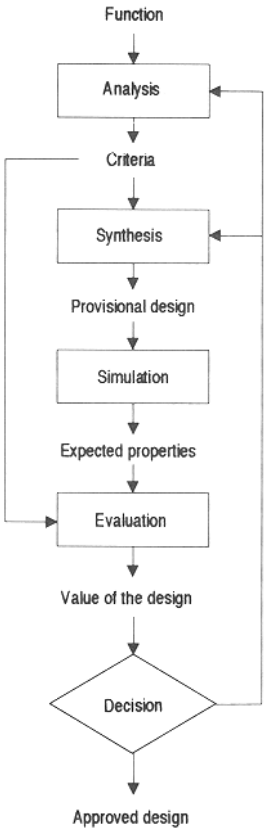

Figure 3.1: The basic design cycle, from Roozenburg and Eekels [154]

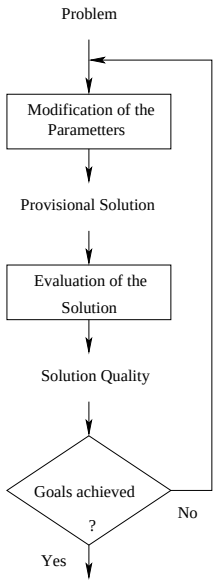

Figure 3.2: The basic optimisation cycle

design of optimum quality. The optimisation method can be manual as it has been the case since the early days of engineering or fully automated thanks to computational optimisation techniques. Different optimisation techniques will be discussed later in section 3.5.

3.3.1 The Quality of a Design

From the point of view of a company the ultimate aim of a design is to generate maximum profit, therefore this is the single objective defining the quality of the design. However in most cases it is impossible to predict reliably the profit that a particular design will generate. Different objectives, which are perceived to be able to maximize the potential profit, have to be used in order to measure the quality of a given design. These objectives can be of two types, performance objectives, and manufacturing objectives.

- The performance objectives describe targets and constraints for the

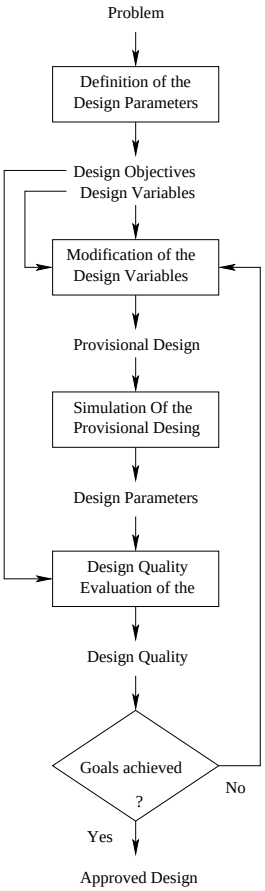

Figure 3.3: The Design Optimisation Cycle

performance parameters of the design, its capacity to achieve certain functions including reliability and durability [4]. The performance parameters of a specific design are generally defined through analysis or simulation of that design.

- The manufacturing objectives describe the aspects of the manufacturing of the design, mainly cost, time, investment, manufacturability, and factory workload. The relative lack of formal and reliable tools to define how a specific design achieve its manufacturing objectives means that these objectives are often overlooked. However, the development of costing methods and software [143, 75] capable of predicting the manufacturing cost of a design along with techniques capable of dealing with qualitative issues such as manufacturability [153] would allow a better account of these objectives. Some of these techniques are emerging, however at the present time they are not very well developed.

The choice of the method to determine the quality of the design is crucial for the design as the selection of different objectives may lead the project towards different horizons. In addition optimisation techniques in general can deal easily with one objective, but experience difficulties in the presence of multiple objectives. Up to the point where optimisation becomes highly difficult when a large number of objectives are required.

Defining the quality of a particular combustor preliminary design is not a simple task, since the performance of a combustor is described by nine key parameters which have been introduced in section 2.2.2. Preliminary design tools do not provide sufficient information to define all those performance parameters. Therefore it becomes necessary to use between 15 to 40 additional performance parameters to constrain the key ones that can not be calculated. As a first approach, the manufacturing objectives will be ignored, however to benefit from the maximum potential of design optimisation those objectives should also be included in future applications.

3.3.2 The design Variables

The design variables are the parameters that the designer may adjust with the aim to improve the quality of the optimised design. These variables represent the different components of the design that can be modified such as, geometry, size, materials, features, volume, and configuration. Those variables can be grouped into the following three categories:

- Binary: A binary variable usually refers to a component which can be either present or absent in the design.
- Integer: An integer variable refers to a feature that can have a number of discrete configurations, for example the number of holes on an effusion patch.
- Real Number: A real numbered variable refers to a feature that has a continuous range of possible value, for example the length or the diameter of dilution ports.

3.4 Application of Design Optimisation

3.4.1 Design Optimisation as applied In Industry

Currently, the most common optimisation technique used in industry is to manually optimise the design. Manual design and optimisation typically consists, in taking an existing design, and adapting it to new requirements by appropriate scaling and slight modifications. Optimisation is achieved using mostly past experience (rule of thumb) and trial and error, optionally employing user driven computational analysis programs recursively until a suitable solution is found. This is particularly the case for gas turbine combustor preliminary design where new designs are mostly based upon previous designs, the experience of the engineer and trial and error.

The application of manual design techniques on complicated design problems results either, in a lengthy and costly design phase, or in poor optimisation if time and resources are limited. The use of optimisation algorithms has the potential to tackle these issues by reducing the amount of manual work required to obtain an optimised design [39].

The Microprocessor industry was among the first to start the transition towards automation of the design process [81]. Design optimisation techniques are beginning be used in the aerospace design process [167, 127], but at present time their use has mainly been limited to the areas of airfoil design and structure.

However in general industry continue to trail behind in this domain, even though design optimisation techniques seems to have reached their maturity [122, 17]. This result in a loss of opportunity for obtaining better designs with reduced costs and design cycle times. A survey of British industries [157] highlighted this fact. In addition several factors inhibiting their use were identified:

- Lack of integration of existing optimisation tools.
- Limited optimisation skills among design engineers.
- The computational cost of simulations.
- The control needs of designers over the design process.
- The complexity of real life optimisation problems.

These inhibitors need to be overcome by the chosen design optimisation method for it to be accepted and effectively used in an industrial environment.

3.4.2 The Requirements for a Gas Turbine Preliminary Design Optimisation Method

The design optimisation method developed for this project, is aimed at providing support for a new preliminary design methodology for gas turbine combustors. its requirements will therefore be based on overcoming the factors inhibiting the use of design optimisation techniques described in the previous section. The requirements for the design optimisation method can be reformulated as follows:

- The chosen optimisation technique should be robust enough to support the complexity of combustor design.
- The optimisation technique should be as efficient as possible in order to achieve its goal within a reasonable amount of time.
- In order to be useful and flexible, the design optimisation method should be easily integrated with a wide range of simulation software.
- It should be simple to use and have a generic set of input parameters capable of dealing with different types of combustors designs.
- A tradeoff should be found between the designer's need of control and the simplicity of use.
- The designer should be able to use the outcome of the optimisation to better understand the design space.

These requirements lead to the design of a Toolbox implementing this framework of design optimisation method and promoted the search of a suitable optimisation technique.

3.4.3 Definition of the Gas Turbine Design Optimisation Problem Features

In order to better define the optimisation problem it is important to define the classification of optimisation problems. This one has been proposed by Roy and Tiwary [156] it allows to define the type and features of the problem posed by the design optimisation of gas turbine combustor preliminary design. It results in the following classification :

- Number Of Parameters: Combustor preliminary design is clearly classified as a multi dimensional problem since it involves 20 to 50 parameters.
- Existence of Constraints: The range of some performance parameters are constrained, therefore it can be classified as a constrained problem.
- Number of Objectives: The large number of performance parameters, 20-30, clearly shows that it belongs to the class of multi objective problems.
- Nature of the Search Space: The search space is unknown and is likely to be multi-modal.
- Nature of the Objectives: The objectives are quantitative since these concerns values of performance parameters.
- Nature of the Equations: The equations defining the relation between the input variable to the performance parameters are not available due to the use of a large quantity of tabulated data. however the problem is presumed to be non-linear, non-smooth and non-differentiable.
- Separability of the functions: The functions relating the input variables to the performance parameters are not separable.

- Nature of the Design Variables: The design variables are static (ie. it is not a dynamic problem).

- Nature Of the Variable Values: The search space is continuous, most of the design variables are real valued, however some integer variables may be present such as the number of holes of a cooling patch.

3.5 Review of the Available Optimisation Algorithms

This section briefly reviews the main types of optimisation techniques that could be used for the optimisation Toolbox.

3.5.1 Analytically based algorithms

The most common algorithm used for minimization is be the Newton-Raphson algorithm [131]. It is a Newton root finding method which uses the first few terms of the Taylor series of a function in the vicinity of a suspected root in order to zero in on the root.

It consists of guessing a starting point close to a suspected root, then geometrically extending the tangent of this point until it crosses zero and then setting the new point at the abscissa of this intersection with zero. This technique can be applied for multi dimensional problems, and provides quadratic convergence. However it requires the evaluation of the derivate of the function. In addition, it requires a relatively good guess for the position of the root and it is not stable

and diverges in some situations, failing to converge or converging to a local minimum.

Conjugate gradient methods such as the Fletcher-Reeves algorithm, and the Quasi Newton techniques, requires the computation of the function gradients or first partial derivate at arbitrary points to provide an approximation of the Hessian matrix.

These methods require a good knowledge of the function and of the search space to be able to guess the position of the likely optimum. It is not possible to apply these techniques to nonlinear constrained problems. In addition difficulties may arise when optimising non smooth functions or high dimensional problems.

3.5.2 Heuristic Search

Hill Climbing (HC) [158] is a robust optimisation technique capable of finding a local minimum for any black box problem, no prior knowledge about the problem's function is required. The idea behind hill climbing can be summarized as follow:

1. Pick a point in the search space.
2. Consider all the neighbors of the current state.
3. Choose the neighbor with the best quality and move to that state.
4. Repeat 2 thru 4 until all the neighboring states are of lower quality.

This technique is extremely simple and does not involve any assumptions about the shape of the fitness function or require the computation of any derivate, and is able to cope with constraints. However this technique requires more function evaluations than the analytically based algorithms and converges towards the first local optimum encountered.

The capacity to find a global optimum can be improved by using Simulated Annealing (SA) [84] techniques which behaves like a hill climbing method

but with the possibility in some conditions to "go downhill" to avoid being trapped at local optima. It was originally inspired by the formation process of crystals in solids during slow cooling, where the system moves randomly, but its probability to stay in a particular configuration depends directly on the energy of the system and on its temperature which is reduced slowly.

3.5.3 Design Of Experiment

Design of Experiments (DOE) is a technique based on the systematic variation of all the independent variables values. By applying statistics to the experimental process, the interactions among the different variables can be studied, and the optimal set of variables can be chosen. These techniques are very efficient for optimisation of a small number of design variables. For high dimensional problems the number of points (evaluations) increase exponentially. A full factorial experimental design requires 3^n points n being the number of variables. Therefore, it becomes very difficult to use these techniques for more than 6 variables, even with a reduced set such as D-optimal experimental design which requires $2^n + 2n + 1$ design points it is difficult to design for more than 10 variables.

The DOE is often coupled to regression techniques, and analysis of variance (ANOVA) to form a more efficient design methodology referred as response surface. The response surface methodology allows to improve the understanding of the search space and it is an extremely efficient design method for problems with limited number of variables. However it does not reduce the difficulties posed by high dimensional problems.

The Taguchi method, can be viewed as a standardized and simplified method of experimental design. Here orthogonal arrays are used to define the design points that needs to be evaluated and statistical analysis is used to analyze the results and to define the optimal parameters. This technique permits the use of a high number of variables. However the interpretation of the data

becomes difficult for high dimensional problems, especially in the cases where the different parameters are highly coupled or skewed. In addition it is more suited to discrete variables problems.

3.5.4 Evolutionary algorithms

These techniques are inspired by the evolution that can be observed in natural environments. The evolutionary algorithms are characterized by the fact that they use a population of design points which compete to survive. These techniques are global and extremely robust, and can be applied on a large number of problems. For small number of variables (<5), the cost in terms of number of evaluations is relatively large, higher than that for analytical methods, or for DOE. However, for problems with a large number of variables the tendency is inverted and for problems with more than 10-15 variables evolutionary algorithms becomes one of the most efficient technique. Three major variants of these techniques can be found in literature:

- Genetic Algorithms, are based on the evolution of problem parameters to maximize a goal [69].
- Evolution Strategies, are similar to genetic algorithms but uses different operators to generate new solutions [164].
- Genetic Programming, is based on the evolution of program trees to maximize a goal [88].

The extreme robustness of these techniques makes them well suited for the optimisation of problems where the functions relating the inputs to the outputs is not known and may have unexpected behavior. In addition, presence of non-smooth functions do not cause specific difficulties. The handling of constraints and multiple objective is also possible [112]. More details concerning these techniques are given in chapter 4.

3.5.4.1 Selection of the optimisation technique

Of the four type of optimisation methods reviewed here, the evolutionary algorithms seems to be the most suitable candidate for combustor preliminary design optimisation. This is due to their robustness and the possibility to handle constraints and multiple objectives.

Out of the three main evolutionary algorithms presented genetic programming seems to be the most suited to optimise the configuration of a combustor. However for the optimisation of parameters, genetic algorithms and evolution strategies are best suited and should provide similar performances. Genetic algorithms were chosen over evolution strategies due to their large number of previous successful applications. It is still necessary to modify the genetic algorithms to adapt them to engineering design problems in order to maximize the performance of the optimiser.

3.6 An Optimisation Toolbox for Gas-Turbine Preliminary Design

In order to integrate design optimisation methods within the preliminary design phase of a gas turbine, it is necessary to design a toolbox which provides the designer with a set of methods and tools that allows the partial-automation of the design process. This leaves the possibility for the designer to optimise numerous parameters of a complex problem using his traditional simulation software for the evaluation of the solutions.

3.6.1 Aim of the Optimisation Toolbox

The aim of this toolbox is to provide a design optimisation method that supports the designer during the preliminary design phase of a combustor, or

for similar design tasks. It needs to be capable of overcoming the inhibitors highlighted in section 3.4.1 and to satisfy the requirements given in 3.4.2. To be useful it has to be as efficient as possible and generate optimised design in a reasonable amount of time. It also needs to be flexible and versatile to permit improvements of existing methods and the addition of new tools. Finally to be used in real life it needs to be user friendly and should allow interaction between the designer and the optimisation process without requiring extensive training in the optimisation techniques.

3.6.2 Architecture of the Toolbox

In order to achieve objectives outlined in the previous sections it is useful to employ a modular, object oriented architecture that permits tools to be added, when available, thereby enhancing the versatility and the performance.

A high degree of modularity is required in order to achieve the desired aims of versatility and expendability of the tool box. This was obtained by using Java as the main programming language, being object oriented it readily allows development of this type of modular architecture. Also its platform independence avoids the burden of porting the code on different architectures and eases the problems associated with working on heterogenous network of computers. Another useful feature of Java is its advanced support for networking and graphics. However this language suffers from being relatively slow compared to C/C++ as well as not being memory efficient. The tool box is to be used for engineering design where simulation is performed by external analysis codes typically written in C or Fortran. Since the simulation process takes a large proportion of the computational time, the relative slowness of Java is not perceived as a handicap. In addition, the capacity of modern computer systems do not put heavy constraints on memory usage.

The object oriented design allows modules to be loaded at runtime depending

on the toolbox requirements, it even permits new modules to be added without recompilation of the code. These modular tools are organized around the implementation of three key functionalities :

- Interfacing: The interfacing module, allows the communication with any third-party software that use text files for input/output. This gives the designer the possibility to use his traditional analysis software to evaluate the quality of a design.
- Evaluation: The different evaluation modules reduce the computational overhead of simulation by giving the possibility to distribute the evaluation of the different designs over a network of heterogenous computers.
- Optimisation: The different optimisation modules are themselves composed of interchangeable sub-modules. This provides flexibility on the choice of operators and specific methods.

Figure 3.4 shows a representation of the Toolbox modular organization.

3.6.3 Description of the toolbox Modules

Once the Architecture has been defined, the details of the different modules of the Toolbox can be described. This section only present a brief description of the modules. A more detailed description at the user and code level is provided in the documentation of the code, which can be found in appendix A of this thesis.

3.6.3.1 The Optimisation Toolbox Main Module

This is the main module of the toolbox where all the different modules are connected. It is the module that controls all the non optimisation related tasks.

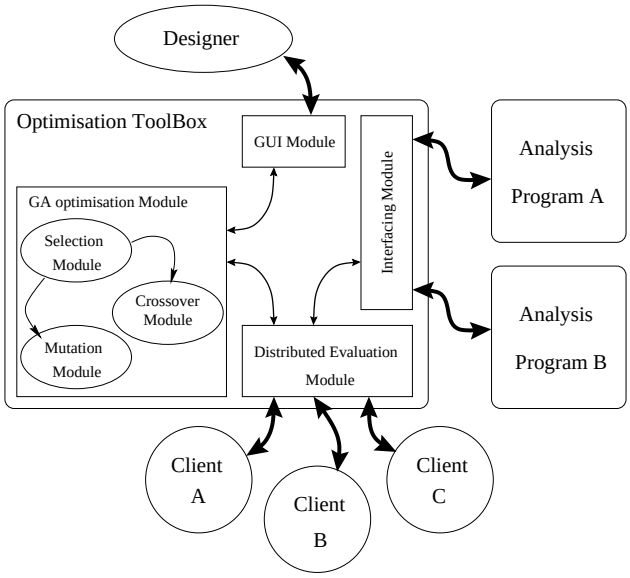

Figure 3.4: Modular Organization

3.6.3.2 The Optimisation Modules

The optimisation modules implements the optimisation techniques. Three different optimisation modules have been coded to implement random optimisation, genetic algorithms based optimisation, or hybrid optimisation. The Genetic algorithms optimisation module will be described in detail in chapters 4 and 5. The hybrid algorithm is in fact a genetic algorithm optimisation technique coupled with a hill climbing type technique.

3.6.3.3 The Interfacing Modules

One of the main requirements of the Toolbox concerns its capacity to be interfaced with a variety of existing simulation software, in order to allow the application of the toolbox to a range of different problems. Another requirement is, the ease of use, thus not requiring the user to program a specific interface to its analysis code.

A common feature of a majority of simulation codes, comes from their ability to communicate through text files. It was decided to take advantage of this feature and perform communication with these codes through text files. In order to generically access any type of text file a method had to be devised to locate and write or extract the relevant informations. The most robust way to arbitrarily locate data within an I/O text file was found to be the use of regular expressions [86] combined with the tokenisation of the file. This technique is based on searching data from a keyword within the file or from its position (line /column) or any combination of the two. This makes it a very robust way to access data for read or write.

The text file to read or modify is first entirely loaded into the memory it is then "Tokenised". A Token is defined as a string of characters between spaces or end of line characters. These Tokens are ordered in a matrix form, the spaces being used as columns separators, and the end of line characters being used as line separators.

This matrix resembles closely resembles closely to a spread sheet which can be assessed by line and columns number. This representation already allows to assess any part of the file if the line and columns numbers are known. However it is not a very robust approach since any change in the file format will affect the position of the accessed data and will require to setup new lines and columns numbers.

In order to improve the robustness of this file interface, a regular expression search technique has been implemented. This technique allows to search and locate keywords in the matrix.

The use of this search method allows to isolate the couple keyword / data from the rest of the file, improving dramatically the robustness of the interface. Ie: the file format can be modified as long as the relative position of the couple data keyword is not altered.

The user is only required to provide the key words and the relative position of the data in the configuration file as described in the code documentation

added in Appendix A.

3.6.3.4 The Evaluation Modules

A very efficient way to reduce the computational overhead of recurrent use of CPU intensive simulation software is to perform the execution of these application using Heterogeneous Distributed Computing [171]. It consists of distributing the execution of processes over a network of computers which might not all be of the same specifications or architecture.

In the case of the distribution of multiple evaluation of the fitness function through the execution of a sequential analysis code, communications only take place between the Toolbox and the analysis software. This communication is performed through text files thus alleviating the need for inter-process communication software (MPI or PVM libraries).

The organization of the distribution of the evaluation tasks is shown in figure 3.5. It consists of placing the objects to be evaluated in a stack. The content of this stack is then distributed to remote clients as these request new objects for evaluation. A client picks an object to be evaluated, evaluates it by running the required analysis code(s), and finally returns it to the Toolbox server and request another object. In order to increase the robustness of the process, the objects that were not yet returned when the stack gets empty are resent to free clients. This avoids waiting indefinitely for an object if it has been sent to a particularly slow client or if it has been damaged through a network error.

The server and the clients are Java modules which use sockets to exchange data. These are therefore unaffected by the heterogeneity of computer architecture across a network. However difficulties may arise from the simulation codes which are usually not platform independent. Obviously these needs to be available on all the architectures present on the network and a method is needed to call the appropriate executable. This problem was tackled through

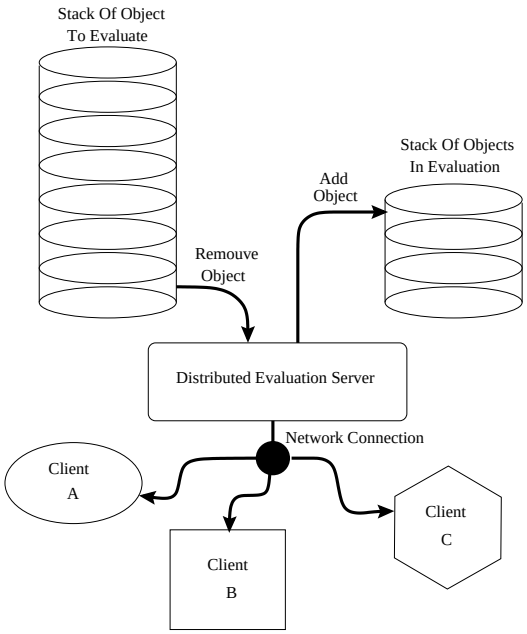

Figure 3.5: Distributed evaluation of objects.

the use of scripting.

This distribution technique of work load allows a linear scaling of the performances with the number of client used for the evaluation of problems with a computational requirement greater than one minute. The object transfer and distribution overhead starts to affect the scaling for problems requiring less than ten seconds in evaluation time. This phenomenon is demonstrated in figure 3.6 for a problem requiring an average of 0.5 seconds for the evaluation. This type of distribution technique was used on a network composed of a mix of architecture and operating systems.

3.6.3.5 The GUI Module

The last part of the tool box concerns the program / user interaction. The addition of a user friendly graphical interface where the designer can follow

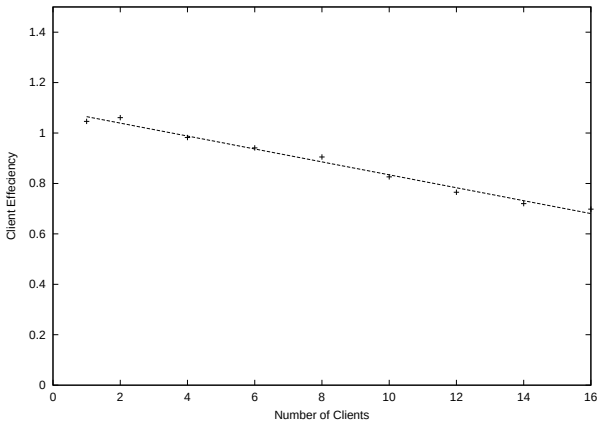

Figure 3.6: Distributed evaluation performance on a problem with short evaluation time (0.5s).

the evolution of all the problem parameters during the optimisation allows the user to get a 'feeling' of the optimisation process without requiring an in-depth knowledge of optimisation procedure. Figure 3.7 shows a typical output of the graphical user interface.

3.7 Conclusions

This chapter has presented design optimisation methods that can be applied to combustor preliminary design. Of the different optimisation techniques presented, a potential candidate for combustor design optimisation has been identified. Genetic algorithms were found to be the most suitable method. The design optimisation method was implemented in the form of a Toolbox that regroup a set of tools in an object oriented architecture. This optimisation toolbox of which the code size is 26500 lines has been coded from scratch and is operational.

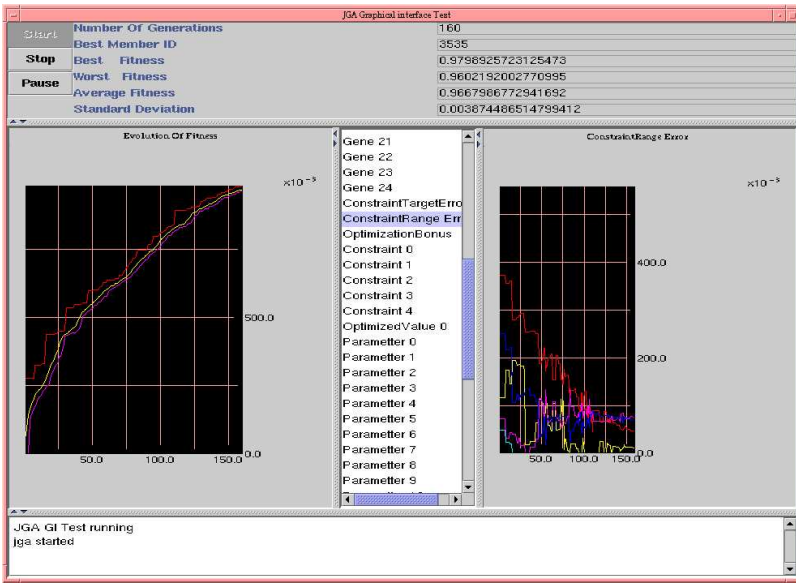

Figure 3.7: Screen-shot of the graphical interface output.

The Toolbox in it's present form the capabilities of the toolbox are the following :

- It is capable to be interfaced with any simulation tools communicating through text files such as the ones described in chapter 4.
- It's optimisation module has been developed around geneticassessed algorithms based techniques genetic(Chapter 5) which have been upgraded to suit the requirement for engineering design (Chapter 6)
- The quality of the proposed designs is assessed using a novel method proposed in Chapter 6.
- It has the capability to optimise problems with a large number of variables (genes) as it has been tested in Chapter 7.

- It has been designed for the optimisation of combustor design but can be applied to other engineering design problems as described in Chapter 8.

The following chapter will present the tools normally used for the preliminary design phase of a combustor and describe the selection of the tools interfaced with the Toolbox.

Chapter 4

Simulation Tools

4.1 Introduction

The aim of this chapter is to describe the simulation aspect of the design optimisation process. The first aspect covered in this chapter concerns the selection of an adequate simulation tool for a joint use with the optimisation Toolbox. For this purpose the requirements for the simulation tool will be identified and the different categories of simulation tool will be described. This will be followed by the description of the semi-empirical simulation package that will be used evaluate the candidates designs. This package is named "Flownet", and it is a proprietary code from Rolls Royce plc.

The following step consisted in increasing the simulation capabilities of the simulation tool by implementing a simple NOx emission prediction model, different modeling techniques have been tested to perform this task.

4.2 Selection of the simulation tool

The Optimisation relies entirely on the simulation capabilities to evaluate the quality of the proposed designs. The simulation tools are therefore crucial for

the performance of the optimisation process. The quality of the optimised design directly depends on the simulation tools.

Three main properties of the simulation tools will affect the optimisation process:

- The speed of the simulation: This affects the time to perform the optimisation.
- The fidelity level or the quality of the simulation: A good prediction of trends is crucial and is more important than prediction of exact values. The optimisation relies more on qualitative rather than quantitative simulation. The variation of a performance parameter is more important than its actual value. Since it allows to find the design with minimal emissions, more accurate calculations of emissions will be made in more advanced phases of the design.
- The level of multi-disciplinarity of the simulation tools: This is crucial to have a view of the design as complete as possible in order to integrate all the performance parameters. Performance parameters that can not be deduced by the output of the simulation tools will have to be fixed by imposing some specific constraints on the design reducing the optimisation potential.

These three factors were used to asses the different simulation techniques in order to define which is the most suitable. The three main approaches are as follow:

- Empirical models (correlations)
- Semi empirical models (1D)
- Numerical Model (CFD Simulation)

All these simulation approaches have the potential to be used in conjunction with the optimisation Toolbox and will be discussed in the following sections.

4.2.1 Empirical Correlations

The processes taking place in a combustor being difficult to model, combustor design has been relying heavily on empirical correlations. These correlations are usually accurate within a given design family. Since their physical representation is poor, the accuracy of the correlations decreases rapidly when the combustor diverges from the original design family. Furthermore, although their quantitative accuracy is relatively good for the type of combustor these have been designed for, their qualitative accuracy highly depends on the variables used for the correlation.

These are the simplest and fastest method to calculate the performance of a combustor. Empirical correlations are usually used for the fast evaluation of different combustor performance parameters like:

- Pollutant emissions (Lefebvre, Rizk, Odgers[92, 148, 180])
- Combustor Sizing (Lefebvre, Murthy [95, 115])
- Combustion Efficiency (Lefebvre, Odgers[93, 179])
- Relight performance (Lefebvre [93])
- Ports discharge coefficients (Lefebvre, Adkins [95, 1])
- Cooling performance (Lefebvre, Stollery, Ballal [95, 168, 7])
- Drop size relationships (Lefebvre, Radcliffe, Jasuja[94, 133, 78])

4.2.2 Semi Empirical Code

These codes are the result of the use of physical simulation in conjunction with empirical relationships and correlations. These methods allow a more refined simulation of the combustor to be achieved in conjunction with a good

physical representation. However they are not able to capture the effect of small geometrical details. They are less anchored to the design family and provide an adequate compromise between simulation speed and accuracy, which makes them a widely used method for the preliminary design.

- Pollutant emissions (Fletcher, Tonouchi[43, 177])
- Simulation of flow within the combustor (Murthy, Stuttaford[115, 170, 169])

4.2.3 CFD Simulations

Computational Fluid Dynamics (CFD) consists in the simulation of the physical processes within the flow field. It can give a very detailed picture of the flow, allowing to appreciate visually the effect of small changes in the design. its 2D or 3D visualisation allow to get a better understanding of the different processes taking place within the combustor. Its flow field definition is very high, however its accuracy in the prediction of quantitative values, such as, the pressure drop and the pollutant emissions is much lower [146], unless it has been tuned to a particular geometry. This problem can be partially alleviated by using Anchored CFD methodologies [113] making in a very efficient tool for combustor design. However its use for preliminary design is still limited due to the high computational cost. The constant increase in computer performance should allow in the future an increased usage of simple CFD model during the preliminary design phase.

4.2.4 Conclusion on the Simulation Technique

Out of the three simulation techniques that have been presented, Numerical simulation seems to be capable of providing the most informations about the design with the capability of obtaining a good accuracy if anchored design

methods are used. However, the computational cost of full CFD simulations is so prohibitive that this solution could not be considered.

The selection of the simulation technique was oriented towards semi-empirical models rather than fully empirical model due to their better physical representation and good accuracy. Semi empirical tools are usually proprietary and are not freely available. The tool used here is Flownet a simulation tool developed in Cranfield for Rolls-Royce [170].

4.3 Background on Flownet

Flownet is a combustor preliminary design simulation tool, it can be described as a geometry-independent, semi-empirical, network model.

The aim of Flownet is to provide information for any operating conditions on:

- Massflow splits
- Pressure drop
- Heat release
- Liner wall temperatures

Flownet is based on a pipe network solver [62] which has been adapted to combustor analysis though the addition of flow and heat transfer analysis capability.

The combustor is modeled by overlaying a network on the system geometry. This network is composed by a set of elements and nodes. The elements represent the domain physical features, such as liner holes, ducts, while the nodes are used to join the elements to one another.

The flow is governed by the satisfaction of the continuity equations and of the pressure drop/flow-rate relationships, which are specific for each type of element. The heat release is calculated through a constrained equilibrium combustion model. Mixing Calculations are performed through a basic mixing model in order to describe the rate of combination of flows. Liner wall temperature are estimated through the use of a conjugate heat transfer model and a discrete radiation model.

While lacking the resolution of three-dimensional models, it is able to provide rapidly reasonably accurate results about the key parameters of a combustor design. The usefulness of Flownet as a design tool has been demonstrated through its industrial user base.

4.4 NOx Model

In order to provide more precise informations in order to assess the combustor design it is important to provide informations on the pollutants emissions generated by the evaluated design and in particular informations about NOx emissions. The model quantitative accuracy in the prediction of pollutant emissions is not vital for the optimisation process. However the model is required to provide reliable information about trends.

As a first approach empirical correlations have been used however their performance was found not to be satisfying. Therefore a simple NOx model was developed to provide a model with a more physical representation.

4.4.1 Empirical NOx Correlations

Two Empirical NOx Correlations were tested for optimisation purpose, The Lefebvre 1983 correlation equation 4.1[91] and the Lefebvre 1984 correlation

equation 4.2 [92].

$$(NOx) = \frac{A_{NOx} P_3^{1.2} e^{(0.009 T_{pz})}}{\dot{m}_A T_{pz} \left(\frac{\Delta P}{P}\right)^{0.5}} \quad (ppmv) \tag{4.1}$$

$$(NOx_{EI}) = \frac{A_{NOx} P_3^{1.25} V_c e^{(0.01 T_{st})}}{\dot{m}_A T_{pz}} \quad (g/kg) \tag{4.2}$$

The use of these correlation provided surprising results. The Lefebvre 1983 correlation gave increasing NOx emissions values as T_{pz} increased and the 1984 correlation exhibited the opposite effect. This can easily be explained by observing the two equations, it can be seen that in equation 4.1 T_{pz} is present both in the numerator and the denominator and that in equation 4.2 in the exponential section T_{pz} has been replaced by T_{st} which is a constant therefore explaining the change in behavior.

Equation 4.1 seems to behave in a way that is more physical (it increase with T_{pz} and it takes account of $\frac{\Delta P}{P}$) than equation 4.2. However correlations are generated using a large amount of historical data that will have inbeadedinclude the effects of many variables which are not represented in these equations, such as: droplet evaporation, geometry effects, swirl, and in general technology improvements.

4.4.2 Introduction to the Simple NOx Model

The aim of this program is to predict the trends in NOx emissions generated by a combustor. Since this program is to be used for optimisation, emphasis has been put in designing a simple code to predict correctly the trends in emissions and in keeping calculation time short, giving a lower importance to the accurate quantitative prediction of the emissions value.

4.4.2.1 Model Approach

The model represents the flametube element of the Flownet network as perfectly stirred reactors, linked together to form a reactor network. Flownet is used to calculate the mass flow of mixed air and fuel, the average temperature, the mixture fraction of the gases, and the average residence time for each element of the emission network. This provides all the necessary information for the initialization of the reactors.

4.4.2.2 Perfectly Stirred Reactor Calculations

The combustion calculations are based on equilibrium calculations and the NO emission calculations are based on the simplified Zeldovich mechanism. The equilibrium calculations are performed by extracting the different component molar fractions and the product temperature from three dimensional tables storing the properties of the products for different condition of stoichiometry, temperature and pressure of reactants.

The three dimensional tables are externally generated. For the tables supplied with this code the program GasEq V0.71 from Chris Morley[49] was used to generate the equilibrium tables.

4.4.2.3 Oxides of Nitrogen Mechanism

Gas turbine combustor generated oxides of nitrogen (NO_x) involve the combined concentration of nitric oxide (NO), nitrogen dioxide (NO_2) and nitrous oxide (N_2O). Of these emissions, NO generally represents more than 90% of the NO_x. We will therefore concentrate on the production of NO.

There are three main ways of NO formations in combustion:

- The *thermal NO* also called Zeldovich Mechanism [192] which consists in the oxidation of molecular Nitrogen in the hot post flame region

- The *prompt NO* [41] which comes the formation of NO in the flame zone.
- The *fuel-bound NO [15]* In this case NO is produced from fuel bound N; this is a problem of fuel quality.

The *fuel-bound NO* is not of a high interest for our model since it is mainly dependent of the fuel quality. The amount of NO produced via this route will be very small since aero-engines fuels are refined to minimize fuel bound N and will be nearly unaffected by the combustor design and therefore will not affect the trends in NO emission.

The *prompt NO* consist in [135] (a) the rapid formation of NO in the flame region due to the reaction of hydrocarbon radicals in the reaction zone $N_2 + CH \rightleftharpoons HCN + N$. The HCN then evolves following the path $HCN \Rightarrow CN \Rightarrow NCO \Rightarrow NO$ as described by Nicol [117], (b) the accelerated rate of *thermal NO* due to super-equilibrium concentration of N and OH, (c) the production of NO by reaction of O atoms with N_2. *Prompt NO* is delicate to model since its mechanisms happens in, or near, the flame region. Therefore a detailed chemistry and flow analysis is required to define the flame region and its local conditions. An analysis of this type is computationally intensive and would not be appropriate for optimisation calculations. The contribution from *prompt NO* can usually be neglected for traditional combustors. It is only for low emission cases like lean premixed combustors that the *prompt NO* route becomes of importance.

The NO is calculated using the *thermal NO* (extended Zeldovich) Mechanism[12] described in 4.3.

$$N_2 + O \rightleftharpoons NO + N$$

$$N + O_2 \rightleftharpoons NO + O \tag{4.3}$$

$$N + OH \rightleftharpoons NO + H$$

This mechanism can be simplified to equation 4.4where [] denotes the molar concentration and $k_1(T)$ is given by equation 4.5This simplification is described in more detail in [67].

$$\frac{d\left[NO\right]}{dt} \cong 2k_1\left[N_2\right]\left[O\right] \tag{4.4}$$

$$k_1 = 1.4 * 10^{11} * \exp\left(\frac{-37947}{T}\right) \quad \left(m^3 kmol^{-1} s^{-1}\right) \tag{4.5}$$

If one assumes that $[N_2]$ and $[O]$ are in equilibrium and that the mixture fraction is constant over time (which is not) equation 4.4 can be simply integrated to equation 4.6

$$\left[NO\right] = 2k_1\left(T\right)\left[N2\right]_{eq}\left[O\right]_{eq}\int_0^{tres} dt \tag{4.6}$$

Equation 4.6 can be expressed in terms of molar fractions see equation 4.7.

$$\frac{n_{NO}}{n_t} = 2k_1\left(T\right)\left(\frac{n_{N_2}}{n_t}\right)\left(\frac{n_O}{n_t}\right)\left(\frac{n_t}{V}\right)t_{res} \tag{4.7}$$

$$PV = n_t R_0 T \;\; therefore \;\; \frac{n_t}{V} = \frac{p}{R_0 T} \tag{4.8}$$

From the equation of state the expression 4.8 can be deduced and replaced in equation 4.7 to express it as a function of T and P instead of V. Finally equation 4.9 and 4.5 will be used for the NO calculations.

$$\frac{n_{NO}}{n_t} = 2k_1\left(T\right)\left(\frac{n_{N_2}}{n_t}\right)\left(\frac{n_O}{n_t}\right)\left(\frac{p}{R_0 T}\right)t_{res} \tag{4.9}$$

Since for a given combustor operating condition n_t, n_{N_2} and p will stay fairly constant we can see that the NO production will be dependent on T and the amount of O atoms, the number of O atoms also being highly dependent of T . Equation 4.9 will be extremely dependent on temperature.

4.4.2.4 Limitations and Assumptions

Since this is a very simple emission model the reader needs to be aware of the limitations and the assumptions made for this code. The reader should as well keep in mind that this code is aimed at predicting trends in pollutant emissions and it is not aimed at predicting accurately the exact amount of emission. For accurate predictions of the pollutant emission a much more complex and detailed model would be required which would be outside the scope of gas turbine design optimisation.

- Perfect mixing is assumed for combustion. This is simplifying a lot the combustion process, giving only an average AFR and therefore an average temperature for a whole element given the exponential dependency of NO emission to temperature, this assumption clearly affect the quantitative accuracy of the results. A better but more complex approach would be to use PDF (probability density functions) for the mixing, which would allow to have temperature distributions and a more accurate prediction of the emissions.

- The residence time calculation is performed using the average residence time, however in reality there will be a distribution of residence time and like for mixing using a PDF approach and tacking account of the variations of mixture fraction over time, would be closer to reality.

- The combustion calculations assume that equilibrium is reached at the exit of each combustion element. A chemical kinetics scheme would be more accurate and allow to provide informations on CO emissions.

- Jet-A is used as a fuel and Jet-A formulation has been simplified to $C12H23$.
- No account is taken of fuel droplets evaporation time for non premix combustion.
- No account is taken of ignition delay.
- Only thermal NO emission created via the Zeldovich Mechanism are considered, neglecting the contribution from *prompt* NO and *fuel-bound* NO. This is a correct assumption for traditional combustors using fuel atomization. In very low NO conditions, such as for lean premixed prevaporised combustors, *prompt* NO becomes a significant proportion of the total NO produced.

4.4.2.5 Validation of the emission model

Since the aim of this model is to provide trends rather than accurate quantitative prediction of emissions, the validation work concerned more the qualitative aspect of the NO prediction. Some verifications were performed on the model to confirm that the predicted trend followed a logical behavior, dependence on the temperature and therefore on the AFR. In [73] a much more complete validation work is presented for an autonomous emission code using the same modeling technique [134].

4.5 Conclusion

During this chapter the modeling aspect of the work has been reviewed, through the presentation of the existing simulation techniques and the description of the selected simulation technique. In addition the capacity of the simulation tools has been increased by the addition of a model to predict

NOx emission trends. The combination of Flownet and the NOx model, provides enough informations to give a fairly accurate definition of the quality of the design. The definition of the design quality could further improved through the addition of other models such as CO prediction or pattern factor estimations.

Chapter 5

Genetic Algorithms

5.1 Introduction

This chapter aims to describe the genetic algorithm based techniques designed for the optimisation Toolbox. These techniques are based on the simple genetic algorithms that have been modified in order to adapt them to engineering design problems.

Firstly some background on genetic algorithms will be given for the reader who is not familiar with this techniques. The idea behind genetic algorithms will be described and some application examples will be presented for diverse engineering domains with a closer look at some applications in the aerospace and gas turbine domains.

In the second section of this chapter the principle behind the conventional genetic algorithms sometimes called simple genetic algorithms (SGA) will be explained, and some theoretical explanation will be given on the effectiveness of those algorithms. In addition, some of the advantages and shortcomings of the SGA will be highlighted.

The third section of this chapter will be concerned with modifications made to the genetic algorithms in order to improve their performance, adapt them

to the design of engineering systems and overcome the inconveniences of conventional genetic algorithms.

Finally, the fourth section of this chapter will deal with the important issue in engineering design optimisation of handling constraints and optimisation for different objectives.

5.2 Evolutionary Optimisation Method Background

This section aims to describe the general idea and basic principles common to the different evolutionary optimisation methods.

5.2.1 The Idea Behind Evolutionary Optimisation

Genetic algorithms belong to the more general classification of evolutionary optimisation techniques or evolutionary programs as Michalewicz [109] classifies them. Those techniques are derived from observations of the natural evolution process.

The different life-forms present on the earth demonstrate an amazing capacity to adapt to the environment as diverse as it can be. Life is even able to adapt to the different changes in this environment. A sudden change will wipe out a few animal species which will soon be replaced by some others newly adapted to the resulting environment as narrated by Pratchet [130].

This extraordinary capacity of evolution possessed by all life-forms has inspired the development of computer-based evolution techniques, which adapt a population of solutions to a particular problem.

All the evolutionary optimisation techniques share the common concept of a population of individual solutions that have been evolved using various

techniques to adapt themselves such as they become the optimum solution of a given problem. The most famous of these techniques are, Evolutionary Strategies [164], Genetic Algorithms [69], and Genetic Programming [88].

5.2.2 Basic Principle

The first question that comes to mind when thinking about the natural evolution process, is how do all the species manage to adapt themselves to all the different types of environment. In order to answer this question it is best to quote a paragraph from Pratchet [130] which explains the natural evolution in a clear and simple way.

> Imagine a lot of creatures of the same species. they are in competition for resources such as food - competing with each other, and with animals of other species. Now suppose that by random chance, one or more of these animals has offspring that are *better* at winning the competition. then those animals are more likely to survive long enough to produce the next generation, and the next generation is *also* better at wining. In contrast, if one or more of the of these animals has offspring that are *worse* at winning the competition, then those animals are less likely to produce a succeeding generation - and even if they somehow do, that next generation is still worse at wining. Clearly even a tiny advantage will, over many generations, lead to a population composed almost entirely of the new high-powered winners. In fact, the effect of any advantage grows like compound interest, so it doesn't take all that long.

This is the mechanism as Michalewicz [109] puts it that will give an advantage towards the faster running rabbits over the slower running rabbits, especially

when they are chased by a fox. This competition among rabbits lead to rabbits being quite fast running animals.

This natural strife for survival is at the basis of the evolution forces that allows the species to adapt to their environment [172]. This natural and biological anchorage of the evolutionary optimisation techniques will be described in 5.4 and is widely described in literature [69, 103, 27].

This idea of species evolving to become more adapted to their environment to win the *competition* and survive, was used as the basis of the the evolutionary programs. Those programs encode the parameter of the problems into a data structure called a chromosome. A population of these chromosomes is generated, a bit like the population of the rabbits. The quality of the solution encoded in each of the chromosomes is evaluated. Some of the chromosomes that are the most adapted to the problem are selected for the recombination process which uses a number of parents usually two and combine their properties to create offsprings. Offsprings from parents which are well adapted to their environment will have a high probability to have a similar or even a better adaptation to their environment. This process is repeated in cycles until a suitable solution is found.

5.3 Example Of Uses

Before going into the details of Genetic Algorithms and their derivations, let's have a look at some applications in order to get a better understanding about the possibilities offered by those optimisation techniques. Although evolutionary programs have a very wide range of applications, the application presented here will be centered towards the engineering domain and more specifically the aerospace and gas turbines domain.

From the start genetic algorithms were viewed as a promising tool for engineering, however the computational power available in the late sixties limited

their use. It is only during the late eighties and nineties when the computing power of desktop computers became significant that these techniques started to be applied to a wide range of engineering problems. Thanks to the work of some researchers in the domain, engineers are becoming more and more aware of the evolutionary based optimisation techniques and apply these techniques to solve various problems. The evolutionary computation techniques are now facing a transition. They have been successfully used as research tools, they now need to be integrated in the engineer's design environment do become an everyday tool as it has been the case for CFD and FEM.

5.3.1 General Engineering Examples

One of the first Genetic algorithm applications consisted of the control optimisation of gas pipeline systems [52]. Evolutionary based algorithms have also been applied to diverse domains such as the traveling salesman problem [46], and the evolution of computer programs [88], a technique that would become known as genetic programming. Schwefel developed Evolutionary Strategies to optimise the parameters of computer models [164]. Deb has been working on the application of Genetic algorithms to engineering design problems [28, 34]. Parmee has been working on the use of evolutionary computation as an engineering design tool rather than just a function optimisation technique [123, 9]. A limiting factor on the application of evolution programs is the handling of constraints . It is not simple to implement methods for handling the constraints required by most of the engineering design problems. Some Basic Constraints handling techniques have been proposed [145, 27] however these techniques reduce the efficiency of the optimisation algorithms especially in the case of highly constrained design where the feasible design space is small compared to the infeasible design space. Michalewicz [110], and Deb [30], have developed better constraints handling techniques that ease the engineering applications.

5.3.2 Aerospace Applications

In the aerospace domain, until recently, there have been only a few applications of evolutionary computation techniques. However there is a strong interest for optimisation in general and genetic algorithms have been considered at [89]. In addition the number of publications mentioning their use is growing rapidly. Early applications consisted of conceptual phase design, such as optimisation of the parametric design of an aircraft for a given flight mission [16], or the stage optimisation of a rocket in order to maximise the payload ratio [178]. For Aerospace design work, similarly to a large number of engineering domains the limiting factor in the use of evolutionary optimisation techniques is the computational expense of the evaluation of the quality of the solution. This limits the use of optimisation to the conceptual phase or the preliminary design phase where the tools used for the evaluation of the solution are either analytical or empirical with low computational requirements[3, 24, 13]. Since the end of the nineties massively parallel computers have become more widely available as well as highly distributed computing over a network of desktop computers. It has now become feasible to use evolutionary optimisation for more computationally intensive tasks such as propulsive nozzle optimisation [151] or wing design using CFD [37, 132, 121].

5.3.3 Gas turbine and Combustor Applications

There has been a few applications to the gas turbine domain such as sensor diagnostics [191], compressor preliminary design [23], turbine nozzle design [51], axial compressor airfoil design [181], or the performance domain were various studies made use of genetic algorithm to optimise the performance of aero gas turbine [185, 186, 116, 71]. On the thermal side some work has been carried out on blade cooling [155].

Unfortunately there does not seems to be much work done on the application

of genetic algorithms to the design of combustors, although Genetic algorithms have been used for other combustion systems such as adaptive control of a boiler burner [183, 44], the evolution of chemical kinetic schemes, and a simple optimisation of NOx emissions of a burner [8].

From those applications it can be seen that evolutionary optimisation techniques have the potential to solve a lot of engineering optimisation problems. These techniques have been successfully applied to many aspects of engineering design. However from these examples two limiting factors emerges. One is the problem of the computational cost of the evaluation of engineering solution which makes the application of evolutionary optimisation techniques more suitable for conceptual / preliminary design than for detailed design where CPU intensive simulation tools are generally used such as CFD or FEM. The other limiting factor concerns the handling of the constraints. Engineering design usually requires a large number of constraints which need to be satisfied in order to generate a good feasible design. Some work will be required to develop efficient and flexible constraint handling techniques.

5.4 Principle of the Conventional Genetic Algorithms

This section will describe the operating principle of the conventional genetic algorithms by going through the algorithm process and the genetic operators. In addition some highlights will be given of the theoretical foundations of conventional genetic algorithms and some of their main advantages and disadvantages will be described.

Conventional genetic algorithms, often referred to as the simple genetic algorithm (SGA) first proposed by John Holland [69] and further developed by Goldberg [54]. The SGA follows the general principle explained in 5.2.2. The genetic algorithm process is very closely inspired by the natural evolution

process [172], it uses a population of individuals whose features are encoded in a way resembling the DNA coding. It uses genetic operators closely coupled to the natural evolution process such as, recombination of two parents to generate offspring, random mutation in the chromosome to generate new genetic material, and a selection mechanism that is biased towards the best members of the population.

5.4.1 Genetic Algorithms Process

The SGA can be seen as a four step process with a two steps initialisation which consists first in generating a population of chromosomes representing the potential solution of the problem to be optimised. The second step is to evaluate the quality of the potential solutions, this allows each member of the population to be assigned a value of fitness reflecting the quality of the solution encoded in its chromosome. The third step consists of selecting some of the best chromosomes issued from the population. During the fourth step recombination will be performed on those high performance chromosomes to generate offspring, in addition some of the chromosomes from the original population are mutated to create a new modified chromosome. A new population is formed from those newly generated chromosomes and the process starts again at the second step for the second generation. The steps two three, and four are repeated until the desired number of generations have been achieved. Figure 5.1 shows a flowchart representing this process.

5.4.2 Problem Encoding and Initial population

A problem can be described by a set or vector of n parameters 5.1.

$$S_v = (v_1, ..., v_n) \tag{5.1}$$

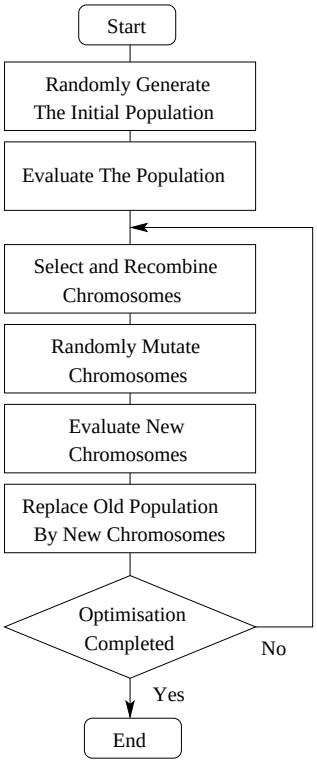

Figure 5.1: Genetic algorithm flowchart process.

each defining a variable of the problem $v_1$to v_n. For the SGA the encoding of the problem parameter is based on a simplified transposition of the coding of the DNA. The DNA uses nucleotides which can be of four types as the basic elements of the code. These nucleotides are assembled into triplets to form a codon. The 64 possibilities of different codons are used to code for the formation of 20 different amino acids. Finally the genes are formed by a set of codons with an alphabet size of 20 [159].

The SGA uses a simplified structure, the codon forming the gene has been given an alphabet size of 2 which is one bit. The value of each gene is represented as a string of bits, the number of bits used to encode the gene will depend on the precision required for the encoding of the parameter value. The precision μ of the binary encoding is defined in equation 6.19,

$$\mu_i = \frac{UB - LB}{2^l - 1} \tag{5.2}$$

where UB and LB are the variable upper and lower bounds and l is the length of the binary string encoding the variable i of the problem. The gene's bit strings are bound together to form a chromosome represented as 5.3

$$S_b = (b_1, ..., b_m) \tag{5.3}$$

where m is the number of bits used to encode the chromosome m is defined as 5.4.

$$m = \sum_{i=0}^{n} (l_i) \tag{5.4}$$

Figure 5.2 shows the encoding process from parameter values to an encoded chromosomes.

At the beginning of each run, the initial population of the SGA is generated randomly by generating a random string of bits for each chromosomes.

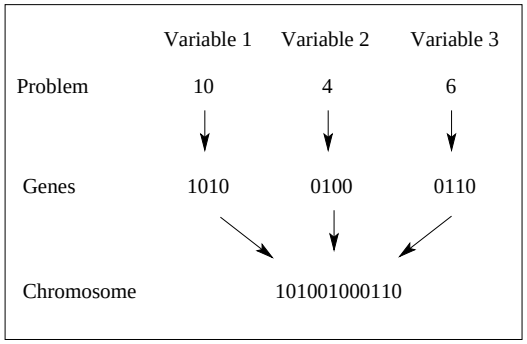

Figure 5.2: Encoding of the problem into chromosomes.

5.4.3 Recombination (Crossover)

The recombination or crossover consists of the combination of the genetic information of two parent chromosomes to form an offspring. Here as well the crossover operator of the SGA is inspired from the biological recombination of the DNA which consists in strand exchange between two DNA duplex [139].

Similarly the crossover operator of the SGA exchange bit strings between two parents chromosomes to form two offspring. The process is performed by randomly selecting one or more break-points on the chromosome. The pieces of string in between those break-points are then exchanged between the two parents to form two offspring.

In the case of the SGA the Crossover operator consists of selecting a crossover point at a position i in the two parent's (x and y) chromosome bit strings 5.55.6.

$$S(x)_b = (b(x)_1, ..., b(x)_i, ..., b(x)_m) \tag{5.5}$$

$$S(y)_b = (b(y)_1, ..., b(y)_i, ..., b(y)_m) \tag{5.6}$$

Where i is a randomly selected integer within the range $[1, m]$. The part of strings from the two parents delimited by the crossover point can be exchanged to create two new child chromosomes $z1$ and $z2$. 5.7 5.8, as shown

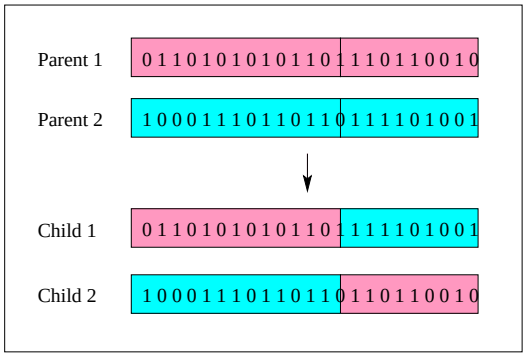

Figure 5.3: Recombination of two chromosomes to form two offsprings.

in figure.

$$S(z1)_b = (b(x)_1, ..., b(x)_{i-1}, b(y)_i, ..., b(y)_m) \tag{5.7}$$

$$S(z2)_b = (b(y)_1, ..., b(y)_{i-1}, b(x)_i, ..., b(x)_m) \tag{5.8}$$

5.4.4 Mutation

The process of mutation in biological systems consists of a blind change in the coding of the gene, this process happens relatively rarely in nature and is mostly detrimental to the cell affected [139]. However in some rare cases this mutation can bring an advantage to the cell and natural selection will favor the organisms which are the most adapted.

In the case of the SGA the mutation operator consist of flicking a bit at a position i in the chromosome bit string 5.9.

$$S_b = (b_1, ..., b_i, ..., b_m) \tag{5.9}$$

Where i is a randomly selected integer within the range $[1, m]$. a as shown in figure 5.4.

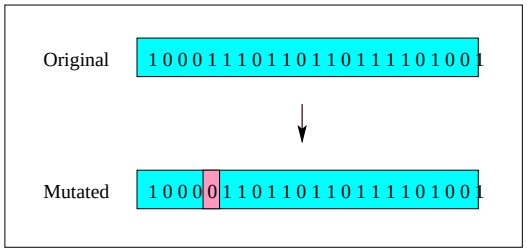

Figure 5.4: Bit mutation in a Chromosome.

5.4.5 Selection and Replacement Mechanism

The selection process gives a bias towards the members of the population that are the most adapted to their environment. For example a rabbit that runs faster than the other rabbits will have an advantage over them and therefore it will have more chances to survive and do what rabbits do best, reproduce and forward its good genes to the next generation of rabbits. On the other hand an unlucky rabbit that would run slower than the other rabbits, will have a higher probability of being eaten by the fox, thus reducing its chances of transferring its bad genes to the next generation.

For the SGA the selection process has a similar purpose, its aim is to favor the dissemination of the good genes in the following generations therefore the selection is achieved by a random process biased towards the members of the population that responds the best to the problem ie: that have the best fitness $f(x)$. The traditional selection operator for the SGA is called the roulette wheel. The idea behind this operator is that each chromosome is allocated an area proportional to their fitness on a virtual roulette wheel. The wheel is spun and the chromosome pointed by the ticker when the wheel stops is selected as shown in figure 5.5.

This roulette wheel selection is performed by first calculating the fitness f of all the member of the population x_i by evaluating them. Once the fitness of each member of the population is known the relative fitness f_r can be

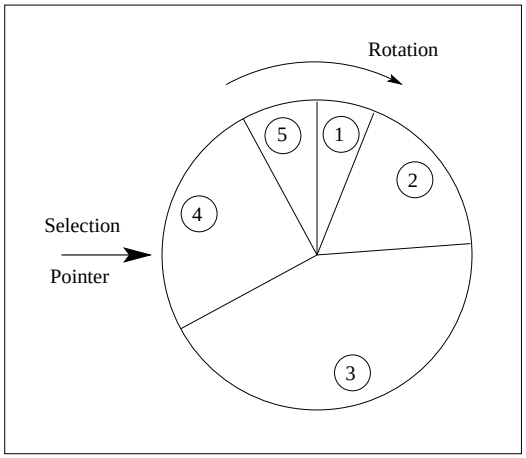

Figure 5.5: Roulette wheel selection process.

calculated, it is defined as follows.

$$f_r(x_i) = \frac{f(x_i)}{\sum_1^n f(x)} \tag{5.10}$$

The cumulative fitness f_c is then calculated.

$$f_c(x_i) = \sum_1^i f(x) \tag{5.11}$$

From 5.11 it can be seen that $f_c(x_n) = 1$ where x_n is the last element of the population. Generate a random number r in the range $[0:1]$ and return the first chromosome whose cumulative fitness is superior or equal to the random number r as follows.

$$f_c(x_i) \geq r \tag{5.12}$$

5.4.6 Theoretical Background of Simple Genetic Algorithms.

When John Holland [69] first developed the genetic algorithms he laid the basic of a theory attempting to explain why genetics algorithms work. This theory is called the schema theory [69] and could be seen as a straight forward version of Fisher's 'Genetical Theory' [42]. The schema theory was then further developed by Goldberg [54] and some others researchers of this field [11, 174, 60].

This theory relies on the binary representation of the solution, this representation allows the use of substrings templates called schema which exploits the similarities between chromosomes.

Let's consider a problem that can be encoded with three bits. This representation could be expressed as the corners of a three dimensional cube of origin 000 and with each corners labeled with a bit string that differs only by one from the adjacent corners as shown in figure 5.6. On this representation of the solution we can introduce the symbol '*' representing a wild card. A schemata will be a string made of binary values and '*' symbols. It will represents all chromosome strings that matches all its binary values except the '*' symbol. For example in the three dimensional cube the schemata '0**' will represent all chromosomes strings starting with '0' ie: all solution located on the front plane of the cube. Each schemata refers to an hyper-plane in the solution space.

It can be shown that a schemata matches exactly 2^n strings where n is the number of don't care symbols '*' in the string, and that each string is matched by 2^m schemata where m is the length of the chromosome string. Since genetic algorithms are population based, they provide information about numerous schemata. The intrinsic parallelism in genetic algorithms comes from the fact that many hyper-planes are sampled when a population of strings are evaluated. Therefore the schemata representation will vary depending on the

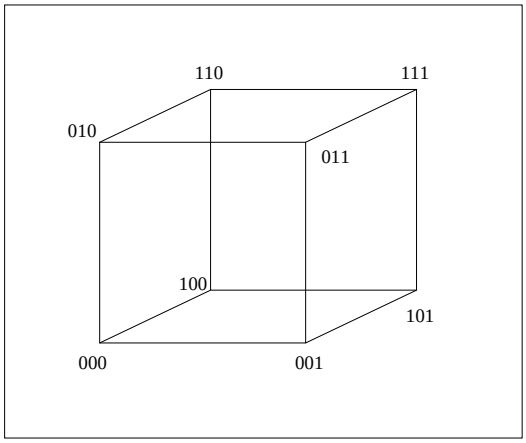

Figure 5.6: Planes on a three-dimensional cube.

fitness of the strings it represents.

There are two parameters that define schemas. The order $o(\varsigma)$ of the schemata ς represents the number of fixed positions (all non '*' symbols) in the schemata ς. A schemata with a low order will be very general and a schemata with a high order will be very specific. The length $l(\varsigma)$ of the schemata ς can be defined as the distance between the first and the last fixed position of the schemata.

The theory [69] suggests that selection will increase exponentially the representation of an above average schemata in the next generation. However the crossover and the mutation will disrupt this process favoring the short, low order schema. The short low-order schemas have been presented as building blocks by Zbigniew Michalewicz [109] to formulate his building blocks hypothesis. " Genetic algorithms seek optimal performance through the juxtaposition of short low-order, high performance schemata called the building blocks"

This schema theory has provided a greater understanding of the behavior of genetic algorithms, such as the population size requirement for a given problem [64] and a better understanding of convergence [141]. As well it

highlighted some problems of the genetic algorithms such as the deceptive effects that can be encountered on certain parameter optimisation problems [53, 60]. However the schemata theory does not completely model genetic algorithms.

Radcliffe has developed the forma theory [137] which can be seen as a generalization of the schemata theory which can be used to analyze the performance of GA coded with non-binary alphabets. In addition, some efforts have been made to try to theorize and model the genetic algorithms using 'Stochastic Modeling and 'Markov Chains' and others methods[161, 162, 104, 144, 184]. These new theories, although mathematically complex, allow the performance of genetic algorithms to be predicted for a given application.

5.4.7 Shortcomings of the Simple Genetic Algorithm

The simple genetic algorithms are very successful in parameter optimisation. However certain problems inherent to their design prohibit their use or reduce their efficiency when applying them to the more problems such as engineering design.

- The most obvious difficulty for parameter optimisation using the simple genetic algorithm is the precision problem. It is inherent to the binary encoding, the precision will depend on the domain size and of the number of bits used to encode the gene [77]. A high precision will require a large number of bits for encoding, increasing the optimisation time. A low precision encoding will mean losses in the optimisation efficiency by looking at discrete points of the solution surface as well as not being able to resolve the actual optimal point.

- The binary encoding of continuous variables creates artifacts known as Hamming Cliffs which adversely affect performance [138].

- The deceptive effects of the simple genetic algorithms will make them converge to sub optimal solutions missing the global optimum [53].
- Genetic algorithms seem to be good at the area of maximum performance for a given problem, however they seem to have difficulty to fine-tune the solution in this area [26].
- The convergence rate of genetic algorithms can be premature leading to sub optimum solutions [97].
- In the simple genetic algorithms there are no methods for constraint handling. This is a very important drawback for engineering problems which are usually highly constrained [124].

All these issues needs to be addressed in order to create an efficient genetic algorithms design tool for engineering. The improvements to the technique proposed during the implementation of the optimisation library, next chapter, have the aim to tackle theses issues, the evaluation of the resulting optimisation technique will be performed in chapter 7.

5.5 Conclusions

During this chapter the idea behind the genetic algorithms have been presented with some examples of use in the engineering domain. This was followed by a description of the simple genetic algorithm techniques, through a description of the problem encoding and of the different operators. Finally some theoretical background about these techniques was provided and the shortcomings of these techniques were identified.

Chapter 6

Implementation of the Optimisation Module

6.1 Introduction

During the previous chapter the principles and theory of the simple genetic algorithms (SGA) have been described. The SGA is at the basis of all others genetic algorithms developments. It has a large potential for engineering optimisation [52] . However it suffers from some shortcomings and inefficiencies [27, 109, 124]. These problems need to be addressed in order to create an efficient engineering optimisation technique.

This Chapter will relate the work done in order to build and improve an SGA Library to transform it in to a efficient engineering optimisation library. First an SGA library was coded in order to be used as the basis of the optimisation library. This was then modified and improved in order to be suitable for engineering design optimisation problems. Such problems typically require a system capable of handling complex definitions of multiples constraints and objectives in order to precisely define the quality of a given solution. In addition the performances of the SGA had to be improved in order to

tackle complex highly dimensional problems in a reasonable amount of time. This performance requirement is emphasized in engineering design where the evaluation of the solution through simulation is generally costly in terms of computational time.

This chapter is therefore structured in four parts:

1. The selection and implementation of the SGA based optimisation library.
2. Review of the available constraint handling techniques
3. The development of a novel constraints and objectives handling technique.
4. The implementation and development of improved techniques and operators for the optimisation library. Including a proposition for a novel mutation operator.

6.2 The SGA Library

The optimisation library is based on an implementation of the SGA for this purpose, different libraries were looked at, with the aim of using them as the basis of the optimisation library. In order to select the right library for optimization purposes a package requirement was established and the available GA libraries were tested against these requirements. The requirements were mostly concerned with the flexibility of the software, its capability to be expanded easily, and its features such as the support of different type of genes or the capacity to run in parallel. The following packages were tested:

- PGAPack [96]
- Genocoop [111]

- SGA Java [65]
- SGA C [52]
- SUgal [72]

These trials showed that none of the packages fulfilled all the requirements. The advanced packages were not very open and would not allow modifications easily, and the simple packages were lacking flexibility and functionalities. It was therefore decided create a new GA library inspired from the "SGA Java" package.

The main aim in the design of the SGA library was to achieve a high degree of modularity in order to be able to easily add new operators or optimization strategies. The core of the optimizer was largely inspired by the basic architecture of a simple (1500 lines of code) but well designed GA library in Java named "SGA Java V1.03" from Stephen J. Hartley [65]. However it was recoded in order to achieve the required modularity and flexibility. The optimizer library uses modules following its needs. The main module contains the optimisation algorithm which is in this case the SAG algorithm but this can be changed to offer a greater choice of algorithms. The algorithm module uses other modules related to GA, such as selection, crossover, mutation, or replacement modules, and some other are more process oriented such as the evaluation module to evaluate the quality of the solution. This modular approach enables new functionalities to be added into the optimisation library.

6.3 Objective and Constraints Handling

This section will describe the development of the objective and constraints handling techniques, these techniques are crucial for the optimiser since they will direct it towards the optimum feasible solution. The description will start

by a brief presentation of the problem posed to the optimisation library. Then the different techniques available concerning the handling of the constraints and the objectives will be briefly discussed. Finally a new method devised to handle constraints and objective will be presented to the reader.

6.3.1 Objective and Constraints for Engineering Design

Gas-turbine combustor design, like as many engineering design problems, is controlled by three factors, explicit constraints which are constraints on the problem input parameters, implicit constraints which are constraints on the performance parameters, and performance parameters that will need to be maximized or minimized to achieve optimal design. These three factors, and their combinations, need to be handled by the optimiser which should implement techniques to control a large number of implicit and explicit constraints and methods to deal with the different parameters to be optimised. A combustor design optimisation will have around twenty to thirty performance parameters which will be composed of implicit constraints and performance objectives.

- Explicit constraints, these are constraints on the problem input variables $f(Pv_i) > a$. They can be calculated directly from the input variables and do not require the evaluation of the chromosome. For example for the optimisation of an effusion patch. Let Pv_1 be the diameter of the hole of the effusion patch and Pv_2 the number of holes for those patch. If the area covered by this effusion patch is constant there will be a limit on the quantity and size of the holes, imposing a constraint on the problem of the type $f(Pv_1, Pv_2) < A_{max}$ where A_{max} represents the maximum allowed hole area and where $f(Pv_1, Pv_2) = \pi(1/2Pv_1)^2 * Pv_2$.

- Implicit constraints, these are constraints on the problem performance parameters $f(Pp_i) > a$. They can not be calculated directly from the

input variables and require the evaluation of the chromosome to determine the performance parameter they constrain. For example for the optimisation of a gas turbine combustor design one might want to constrain the zone air / fuel ratio (AFR) in order to achieve the desired combustion conditions. Let Pp_1 be the zone AFR then the optimiser will need to satisfy the following constraint $Pp_1 = AFR_{desired}$. However since it is impossible to achieve exactly $AFR_{desired}$ the equality constraint is transformed into two inequality constraints: $AFR_{desired\,Min} < Pp_1 < AFR_{desired\,Max}$.

- Performance objectives, these are objectives for the problem performance parameters $max\ f(Po_i)$. They can not be calculated directly from the input variables and require the evaluation of the chromosome to determine the optimised performance parameter. For example for the optimisation of a gas-turbine combustor design, let Po_1 be total NOx emissions produced by the combustor, the aim of the optimisation will be to $min\ Po_1$.

Different techniques need to be implemented in order to allow the optimisation library to deal efficiently with these three factors. Let's first review the different constraint handling techniques available as well as the different objective optimisation techniques. Then describe the method that was proposed and implemented to tackle these three control factors.

6.3.2 Review Of the Available Constrains Handling Techniques

Michalewicz in [109] presented formally the problem of constraint handling as a general non linear programming problem which can be expressed in the following form:

$$\text{Find a set of variable } \overrightarrow{v} \text{ which maximise } f\left(\overrightarrow{v}\right) \tag{6.1}$$

subjected to m inequalities and n equalities conditions.

$$g_i(\overrightarrow{v}) \leq 0 \; i = 1, ..., m \tag{6.2}$$

$$g_i(\overrightarrow{v}) = 0 \; i = m, ..., n \tag{6.3}$$

There are different techniques available to handle the different constraints and direct the optimisation towards a feasible design. This section will briefly describe the main available techniques and discuss their different merits. There are two types of constraints, explicit and implicit constraints as Parmee classifies them [124]. The explicit ones do not cause an important problem since in many case these can be alleviated, see section 6.3.3. However the implicit ones are more difficult to tackle and will affect the performance and the efficiency of the optimisation see sections 6.3.4 to 6.3.9.

6.3.3 The Special Case of Explicit Constraints

The problem caused by the Explicit constraints, since they are based on the problem input variables can be relatively easily solved by making sure the input variables never breach the constraint. This can be achieved either through a good representation of the problem or through some specific operators. In most cases a good representation of the problem can allow the elimination of these constraints. For example for the case of the holes in the effusion patch described in the previous section, the problem could be represented differently. Pv_1 would still represent the diameter of the hole of the effusion patch, but if Pv_2 is used to represent the area of the holes in the effusion patch A_{max} simply becomes the upper boundary of Pv_2. Removing completely the need for the constraints. In the more complex case were it is not possible to change the problem representation it is possible to design special mutation and crossover operators that take the explicit constraints into account and use it to produce only valid chromosomes [112].

6.3.4 The Death Penalty Approach

This might be the simplest technique, the individual that violates the constraints is simply eliminated, this can be done through setting its fitness $f_i' = 0$giving it a null probability of being selected.

This technique can be useful when there is no way to determine how bad the solution is. For example when the solver evaluating the solution crashes, failing to return any value there is no other way than to eliminate the member of the population that caused this crash. This is the traditional method for evolutionary strategies [164].

However since this technique just eliminates the infeasible solutions, it does not give any indication to the optimiser to direct it towards feasible regions, resulting in poor performance when the feasible region represents a small region of the search space. This problem in emphasized for complex optimisation where the feasible region represents an extremely small fraction of the search space and the initial population is unlikely to contain any feasible solutions.

6.3.5 The Fixed Penalty Approach

This method consists of imposing the same penalty to each member of the population failing to satisfy all the constraints $f_i' = f_i - p$. This approach ensures that the individuals which breach a constraint become sub optimal, but without eliminating them completely. Since the fitness value is still present the optimiser will be guided towards the region of maximum f_i and not towards the feasible region. This can be acceptable if the region of maximum f_i is enclosed in the feasibility region however it can be problematic when region of maximum f_i is not enclosed in the feasible region. This problem could be slightly improved by using s set of discrete values defining the amount of constraint violation [70].

6.3.6 The Penalty Function Approach

An improvement on this method is to use a penalty function that defines a penalty proportional either to the magnitude of the constraint violation or to the number of constraints violated $f_i' = f_i - P(x)$. There are different methods to define $P(x)$, which can be defined as:

- A function depending of the number of unsatisfied constraints.
- A function proportional to the distance of the constraints.
- A function proportional to the cost of repairing the individual.

These techniques when $P(x)$ is defined adequately, allows the optimisation to be directed towards the less infeasible portion of the search space whether the region of maximum f_i is enclosed in the feasibility region or not. They can therefore be used in the case of difficult constraint satisfaction problems where the initial population is unlikely to contain any feasible solutions. In order to minimize the disturbance of the fitness function the penalty should be as small as possible [26]. In addition Richardson [145] derived a few guidelines for the definition of the penalty function. However the definition of $P(x)$ is difficult and problem dependent, inappropriate setting of $P(x)$ can make the problem become GA-hard [26] and will not lead to the feasibility region. In addition the definition of $P(x)$ might not be suitable during the whole optimisation process.

6.3.7 The Variable Penalty Function Approach

The idea that invalid solutions can be tolerated during the initial part of the optimisation but have clearly to be avoided during the later stages of the optimisation have led researchers to develop variable penalty functions that dynamically adapt the penalty to the stage of the optimisation. Joines and

Houck [80] proposed to have the penalty function a function of the generation number, Increasing the selective pressure towards feasible individuals as the number of generations increase. Michalewicz and Attia [110] proposed a method inspired from simulated annealing techniques.

Those methods are relatively complex and selection of the tuning parameters they use is critical [21].

6.3.8 The Feasible / Infeasible Approach

For this approach the general idea comes from the assumption that feasible chromosomes should always be preferred over the infeasible ones as suggested by [145].

Some researchers have implemented techniques based on this assumption [129, 30] and derived three precepts.

1. Any feasible solution is preferred to any infeasible solution.
2. Among two feasible solutions, the one having the better objective function value is preferred.
3. Among two infeasible solutions, the one having smaller constraint violation is preferred.

These techniques have in common the separation of the optimisation function for valid and invalid parameters and the use of the worst feasible element fitness to devise the fitness of the maximum fitness of the infeasible element.

The fitness function proposed by Powel and Skolnick [129] is of the type:

$$F(\overrightarrow{v}) = \begin{cases} f(\overrightarrow{v}) & \text{if } \overrightarrow{v} \in \mathcal{F} \\ f(\overrightarrow{v}) + r\sum_{i=1}^{n} f_i(\overrightarrow{v}) + \rho(\overrightarrow{v}, t) & \text{if } \overrightarrow{v} \in \mathcal{S} - \mathcal{F} \end{cases} \tag{6.4}$$

Where $\mathcal{S}$ designate the entire search space, and $\mathcal{F}$ designate the feasible one, r is a constant, n represents the number of constraints and $\rho(\overrightarrow{v}, t)$ is defined as follows in equation 6.5 and is variable with time since it depends of the max and the min fitness present in the population.

$$\rho(\overrightarrow{v}, t) = \max\left\{0, \max_{\overrightarrow{v} \in \mathcal{F}} \{f(\overrightarrow{v})\} - \min_{\overrightarrow{v} \in \mathcal{S}-\mathcal{F}} \left\{f(\overrightarrow{v}) + r\sum_{i=1}^{n} f_i(\overrightarrow{v})\right\}\right\} \quad (6.5)$$

Deb [30] proposed a function of the type:

$$F(\overrightarrow{v}) = \begin{cases} f(\overrightarrow{v}) & \text{if } \overrightarrow{v} \in \mathcal{F} \\ \min_{\overrightarrow{v} \in \mathcal{F}} \{f(\overrightarrow{v})\} + \sum_{i=1}^{n} f_i(\overrightarrow{v}) & \text{if } \overrightarrow{v} \in \mathcal{S} - \mathcal{F} \end{cases} \quad (6.6)$$

Both methods achieved interesting results and seem to be promising. Especially Deb's Implementation. From this it can be seen that both fitness definitions depend on the minimum feasible fitness present in the population. The Powel and Skolnick method tends to have difficulties when the ratio of $|\mathcal{F}| / |\mathcal{S}|$ is small [108].

Deb's infeasible individuals fitness depends only on the distance towards the satisfaction of the constraints $\sum_{i=1}^{n} f_i(\overrightarrow{v})$ since the minimum fitness of the feasible individual will be a constant during each generation. This allows the optimisation to be directed towards the feasible region without any perturbation due to the fitness function, which is not the case in the Powel and Skolnick method. One inconvenience that could be found for this method is the fact that the fitness of the infeasible individuals is dependent of the fitness on other members of the population.

6.3.9 Exotic Constraint Handling Approaches

In addition some more exotic approaches can be found in literature and have been mentioned in [112] or in [21].

- Handling the constrains in a particular order [165].
- Co-evolution of solution and constraints
- Computational model of immune system approach

6.3.10 Single / Multi Objective Optimisation

Single objective optimisation consists of the optimisation of a problem in order to maximize / minimize the value of this objective. As it name says it can only be used to optimize one single objective. However this single objective can be a function of many different objectives. This particularity has been commonly used through different techniques such as the simple weighted sum of the objectives [76], goal attained or target vector optimisation [189]. All these techniques are able to give a set of optimum parameters but will not be able to give a tradeoff surface for the different objectives. This is the most widely used optimisation method amongst GA practitioners.

Multi objective optimisation consists in the simultaneous optimisation of competing objective functions by making the GA population evolve towards the optimum tradeoff surface of the different objective. The dominating approach for these optimisation problems seems to be the Pareto-based techniques.

A individual is said to be Pareto-optimal or non dominated when no other individual is better in all the objectives [19], it therefore lies on the optimal tradeoff surface.

The GA implementation consists in giving an equal probability of reproduction to all the non-dominated individuals [52] and then lower probability for the dominated one depending on their degree of domination either through non-dominated sorting [52] or the number of individuals by which it is dominated [47]. Many researchers reported good results of multi objective optimisation for different engineering domains [22, 101, 40, 105, 175].

When considering the multi-objective techniques as bi-objectives these techniques are extremely interesting and give meaningful result to study the solution space. For example it could be used to study the tradeoff between cooling flow and wall temperature, or NOx emissions and relight loading.

However these techniques start to become less interesting as the number of objective dimensions increase [79]. Multi-objective techniques can still be exploited reasonably easily for three objective problems, but, they find their limitations for higher dimensional problems. Visualization becomes highly complex and less meaningful, how can a tradeoff surface can be visualized in twenty dimensions? It still is possible but it becomes quite difficult to extract valuable informations, one can look at a 2D slice of the multi dimensional space, however the amount of meaningful information decrease rapidly as the number and the inter-relations of dimensions increase. The second problem is more critical, it concerns the the resolution of the tradeoff surface. The dimensional increase of the tradeoff surface will require a larger number of sample points in order to resolve its shape which means a larger population. For a resolution r and a number of objectives n the number of sample points is defined as $S = r^{n-1}$, it can clearly be seen that n will need to be small in order to have a reasonable number of sample points and $n = 20$ is clearly out of question.

To conclude for the Pareto based multi-objective techniques, although theses techniques are very useful and interesting for engineering design, they can only be directly used on problem with a low number of objectives. Therefore there is a need to to reduce the number of objectives for some problems.

6.4 Target Optimisation

The proposed idea to tackle constraint and parameter optimisation for engineering design comes from approaching the optimisation problem through a different angle. Resulting in the creation of a generic framework suitable

for the optimisation of combustor parameters and other engineering design problems.

6.4.1 Target Representation

Looking at an engineering design problem one might reduce it to the optimisation of a set of performance parameters satisfying a certain range constraints on others. Wienke's idea of target vector optimisation [189] and extending it, this could be expressed as the optimisation of a set of n problem variables $\overrightarrow{v}$ such that the set of m performance parameter $\overrightarrow{p}$ approach a set of targets $\overrightarrow{ta}$ while $\overrightarrow{p}$ satisfies the lower and upper boundaries $\overrightarrow{b_l}$ and $\overrightarrow{b_u}$ such that $\overrightarrow{b_l} \leq \overrightarrow{p} \leq \overrightarrow{b_u}$. Therefore the optimisation of each performance parameters correspond to the achievement of a target. For this technique all the targets do not need to be finite values. The targets which are not finite consist of maximization or minimization objectives $\overrightarrow{o}$ and the finite targets $\overrightarrow{t}$ form a subset of $\overrightarrow{ta}$ such as $\overrightarrow{ta} = \overrightarrow{o} + \overrightarrow{t}$.

6.4.2 Range Error, Target Achievement and Optimisation Factors

This representation allows three factors to be extracted defining the quality of a given design.

Assume that $\overrightarrow{p}$, $\overrightarrow{t}$, $\overrightarrow{b_l}$ and $\overrightarrow{b_u}$ are defined such that they are strictly positive.

1. The range error factor $R_e\left(\overrightarrow{v}\right)$is defined as a function of the normalized distance between $\overrightarrow{p}$ and the closest point inside $\overrightarrow{b_l}$ and $\overrightarrow{b_u}$. λ_i representing the normalized distance between p_i and point inside $b_{l\,i}$ and $b_{u\,i}$.

$$R_e\left(\overrightarrow{v}\right) = \left(\frac{\sum_{i=1}^{n} \lambda_i^2}{n}\right)^{\frac{1}{2}} \tag{6.7}$$

$$\lambda_i = \begin{cases} 1 & \text{if } b_{l\,i} \leq p_i \leq b_{u\,i} \\ 1 - \max\left\{\frac{p_i - b_{u\,i}}{p_i}, \frac{b_{l\,i} - p_i}{b_{l\,i}}\right\} & \text{otherwise} \end{cases} \tag{6.8}$$

It can be seen from 6.7 and 6.8 that the target achievement factor range error factor $R_e(\overrightarrow{v})$ will vary between 1 when all range constraints are satisfied $\overrightarrow{v} \in \mathcal{F}$, and tend towards 0 when the performance parameters are at the extreme of the range $\overrightarrow{v} \in \mathcal{S} - \mathcal{F}$.

2. The target achievement factor $T_a(\overrightarrow{v})$ is a function of the normalized distance between $\overrightarrow{p}$ and $\overrightarrow{t}$. $\delta_i$representing the normalized distance between p_i and t_i.

$$T_a(\overrightarrow{v}) = \left(\frac{\sum_{i=1}^{n} \delta_i^2}{n}\right)^{\frac{1}{2}} \tag{6.9}$$

$$\delta_i = \begin{cases} \frac{p_i - t_i}{p_i} & \text{if } p_i > t_i \\ \frac{t_i - p_i}{t_i} & \text{oderwise} \end{cases} \tag{6.10}$$

It can be seen from 6.9 and 6.10 that the target achievement factor will be 0 if all targets are achieved exactly and will tend towards 1 when the performance parameters are well out of range.

3. The optimisation factors $O_i(\overrightarrow{v})$ give an indication of the optimisation of the optimisation objective o_i which represents the direction of the optimisation, maximization or minimization . $O_i(\overrightarrow{v})$ needs to be selected such as poor performance towards the minimization or maximization of t_i, $O_i(\overrightarrow{v})$ will tends towards 1 and a good performance it will tend towards $+\infty$.

6.4.3 Handling of the Range Constraints.

In section 6.3.2 the different constraint handling techniques have been reviewed. The most interesting technique among all those presented seems to be the feasible / infeasible approach presented in section 6.3.8. The three

precepts defined by this approach will be used to design the constraint handling technique. However the approach used by [30] or [129] will not be used since they makes use of the fitness of the worst feasible individual present in the current population, which further complicates the algorithm and reduces generality. The constrained fitness will be set as

$$F\left(\overrightarrow{v}\right) = \begin{cases} f\left(\overrightarrow{v}\right) & \text{if } \overrightarrow{v} \in \mathcal{F} \\ R_e\left(\overrightarrow{v}\right) & \text{if } \overrightarrow{v} \in \mathcal{S} - \mathcal{F} \end{cases} \tag{6.11}$$

Since $R_e\left(\overrightarrow{v}\right) \in [0:1]$ then $f\left(\overrightarrow{v}\right)$should be chosen such as $f\left(\overrightarrow{v}\right) \in [1:+\infty[$ therefore satisfying all three principles of the feasible / infeasible approach but avoiding the use of the lowest feasible fitness.

6.4.4 Implementation

For combustor design this target representation can be very useful, the designer can be faced with three kinds of problem that require solving in a generic method:

1. Achieve a design based on a set of m target performance parameters $\overrightarrow{t}$ satisfying m constraints $\overrightarrow{b_l}$ and $\overrightarrow{b_u}$, there the constraints become relatively unnecessary since these will be aligned with $\overrightarrow{t}$.

2. Optimise a single optimisation objective then $\overrightarrow{o} = o_1$ and a set of $m-1$ target performance parameters $\overrightarrow{t}$, while satisfying m constraints $\overrightarrow{b_l}$ and $\overrightarrow{b_u}$.

3. Optimise a set of l optimisation objective $\overrightarrow{o}$ and a set of $m-l$ target performance parameters $\overrightarrow{t}$, while satisfying m constraints $\overrightarrow{b_l}$ and $\overrightarrow{b_u}$.

Problems of the type one and two can easily be implemented using single objective representation.

Problems of the type three should be optimised using multi-objective optimisation techniques. However the single objective optimisation of a problem of type three can still be performed using single objective by transforming the optimisation objectives into targets when the designer has some idea of the achievable objectives which is usually the case in combustor design where the designs of new combustors are based on existing designs. In addition, a problem of type three can also be optimised using a single objective by creating a global optimisation factor $O(\overrightarrow{v})$ as a function of all the $O_i(\overrightarrow{v})$ even when the user does not have a good knowledge of the achievable objectives.

For the problems two and three the targets are just there to give a direction to the optimiser, the optimisation objective should have the priority over the target. For example when optimising NOx emissions the combustor will tend to go lean, pushing all targets related to the AFR towards their upper range. The target optimisation is just used to push using a small pressure the unaffected performance parameters towards their target value. This is possible by using to our advantage a drawback of the penalty function, if $T_a(\overrightarrow{v})$ is treated as a weak penalty function, it will direct the optimisation towards the target in the absence of $\overrightarrow{o}$, but in the presence of $\overrightarrow{o}$ it will just create a drift towards the targets since $O(\overrightarrow{v}) \gg T_a(\overrightarrow{v})$.

Based on this the single objective fitness function can be defined as:

$$F(\overrightarrow{v}) = \begin{cases} O(\overrightarrow{v}) - T_a(\overrightarrow{v}) & \text{if } \overrightarrow{v} \in \mathcal{F} \\ R_e(\overrightarrow{v}) & \text{if } \overrightarrow{v} \in \mathcal{S} - \mathcal{F} \end{cases} \tag{6.12}$$

Or for a multi objective fitness function it becomes

$$F(\overrightarrow{v}) = \begin{cases} O_i(\overrightarrow{v}) - T_a(\overrightarrow{v}) & \text{if } \overrightarrow{v} \in \mathcal{F} \\ R_e(\overrightarrow{v}) & \text{if } \overrightarrow{v} \in \mathcal{S} - \mathcal{F} \end{cases} \tag{6.13}$$

6.5 Performance improvement

Within the domain of evolutionary optimisation techniques one could find six different ways of improving the performance of the optimisation: technique improvement, adaption to the domain, operators improvement, hybridation with other optimisation techniques, multi processing, reduction of the number of exact evaluations. Four of the above come from Ginnakoglou [50], however he omitted to mention technique improvements and adaption to the domain.

Some of the techniques described in the following sections such as dynamic vector mutation are novel methods specifically developed for this problem but most were found from the literature. All the techniques described here were implemented in the optimisation library except the reduction of the number of exact evaluations section 6.11. This was due to the fact that the application of this type of performance improvement was not necessary for the type of problems studied during this work. The following sections will describe the implementation of the performance improvements techniques regrouped in the six ways previously cited.

6.6 Technique Improvement

This section regroup the improvement of the genetic algorithm technique.

6.6.1 Elitism

One of the first improvements proposed for the SGA technique is called elitism and was developed by Dejong [35], it is now a classical operator for the GA. In the traditional SGA the population is not carried through the next generation and therefore the population at the next generation might have a best member lower than the previous generation since there is no guarantee

that the genetic material of the best member will be carried through to the next generation.

It is now a common technique to reintroduce the best member of the population of generation g into the population of $g + 1$. This technique guarantees that the genetic material of the best members is not lost in between generations.

6.6.2 Steady State Replacement

An observation derived from the idea of elitism highlighted the fact that good genetic material is destroyed when the population is totally replaced with a new one "generational replacement". The idea that was proposed for the steady state replacement is to only replace a fraction of the newly generated chromosomes in order to keep a large gene pool and avoid the destruction of potentially good genetic material [27]. This technique does not replace the usefulness of elitism due to the stochastic nature of the the replacement process that can occasionally replace the best solution by a lower quality one.

6.6.3 Fitness Scaling

It has been shown in [188] that the selection pressure $Sp = \frac{f_{max}}{f_{avg}}$ has a strong influence on the performance of the GA and that it should be controlled as directly as possible. A selection pressure that would be to high might bring the optimisation to a premature convergence and Conversely, a selection pressure that would be to low will not direct the optimisation strongly enough and genetic drift will appear in the population [54]. Whitley has even stated in [188]:

> It can be argued that there are only two primary factors in genetic search: population diversity and selection pressure. These

> two factors are inversely related. Increasing selective pressure results in a faster loss of population diversity. Maintaining population diversity offsets the effect of increasing selective pressure. In some sense this is just another variation on the idea of exploration versus exploitation that was discussed by Holland and others.

For un-scaled fitness the selection pressure depends largely on the shape of the fitness function which therefore needs to be carefully chosen to provide the right selection pressure. Another problem comes from the fact that even with a good fitness function the selection pressure will vary from the start to the end of the optimisation. At the start the selection pressure will be quite high with large improvements in the population, it will then weaken along the optimisation to finish very low towards the end when only small improvements are possible.

Fitness scaling consists of techniques to maintain the selection pressure relatively constant [52]. One of the most efficient and simple techniques is called fitness scaling or windowing. The fitness of all the chromosomes is scalded linearly using 6.14.

$$f' = af + b \tag{6.14}$$

which represents a simple linear scaling where the constants a and b are calculated at each generation in order to keep the selection pressure Sp constant along the whole optimisation process. Using constant Sp as a control constant one can define the maximum scaled fitness as 6.15.

$$f'_{max} = f'_{avg} * Sp \tag{6.15}$$

Taking the minimum scaled fitness constant $f_{avg} = 1$ the coefficients a and b can then be calculated using respectively 6.16 and 6.17.

$$a = \frac{f'_{max} - f'_{avg}}{f_{max} - f_{avg}} \tag{6.16}$$

$$b = f'_{avg} - a f_{avg} \tag{6.17}$$

With linear scaling the fitness average ratio, 6.18, will be kept constant during the scaling.

$$fr = \frac{f_{avg} - f_{min}}{f_{max} - f_{min}} \tag{6.18}$$

It can then be easily shown that for f_{min} to be positive it is required that $fr \leq \frac{1}{Sp}$, this is a problem since the GA requires positive fitness values. The traditional approach suggested by Goldberg [52] is to set the invalid fitness to 0. Setting the scaled fitness of a gene to 0 means to eliminate it from the reproduction selection by giving it a zero probability of being picked. However this should only happen for a small number of genes of very bad quality.

6.6.4 Random Number Generator

During the design of the GA module concerns were raised about the effect of the quality of the pseudo random number generator over the performance of the optimiser. At a time it was thought to design a high quality random generator. However a study by Meysenburg and Foster [107] showed that the quality of the random number generator has hardly any influence over the performance or the efficiency of the genetic algorithms. A recent experimental study from Cantu-Paz [18] seems to suggest that a random number generator of poor quality when used for the initialization can affect the performance. These results are for a random generator with a very small periodicity of $10^3$and therefore does not show the need to change from a reasonable quality pseudo-random number generator to a more complex high quality generator. Therefore a relatively simple random number generator was used based on the Java language random number generator. This employs a 48-bit seed modified using a linear congruential formula to create numbers formed by 32 pseudo-randomly generated bits [86, 131] modified in order to provide a

generator for random numbers of different types with a reliably settable seed. The user defined seeds gives a repeatability capacity for the optimisation experiments.

6.7 Adaptation to the domain

The SGA optimisation technique is entirely independent from the application domain and therefore can be applied to an extremely wide range of problems however this problem independence affects the efficiency of the optimisation [58]. For the optimisation library developed here the application domain is engineering design, therefore some modifications can be performed in order to adapt the optimisation technique to the domain. These adaptations will result in a loss of domain independence but will allow the performance of the optimisation to be maximized for the engineering domain.

6.7.1 Real Coding

Engineering design parameters are usually expressed as real numbers or integers, however in the SGA the problem is encoded as binary numbers. This means that the design space needs to be transposed into binary space to create a new problem suitable for the SGA [109] (pp7). Using a binary (discrete) coding when optimising on a continuous search space can cause a number of difficulties:

- In certain conditions the transition from a gene value to a neighboring value of the continuous search space will require the modification of a large number of bits. This problem is referred as the Hamming cliffs [68].
- The transposition increase the complexity of the problem. It requires

mapping between the binary solution and the equivalent solution in the problem domain representation

- It create some accuracy losses. The mapping between the real space and the binary space will reduce the precision of the representation by increasing μ depending on the length l of the bit string used to encode the gene i of the problem as stated in equation 6.19.

$$\mu_i = \frac{UB - LB}{2^l - 1} \tag{6.19}$$

where UB and LB are the variable range upper and lower bounds. From 6.19 it can be seen the number of bits will depend on the range of the variable to be encoded and of the required accuracy. However the complexity, the size and the duration of the optimisation depends on the number of bits to be optimised. Therefore there will have to be a compromise between the accuracy of the encoding and the time available for the optimisation.

It was found desirable to avoid this design space transposition which has the disadvantages previously cited, this would be an unnecessary burden for the design engineer. The use of real encoded genes allowed this transposition to be avoided.

The schemata theory [52] seems to suggest that a low cardinality alphabet will be more efficient than a higher cardinality one [53]. This would normally reduce our interest for this kind of encoding. However some studies report good performance of the real coded GA [77, 190, 26, 38] which seems to perform similarly or even better than binary coded GA. Until recently there was no theoretical explanation for such good performance of the real coded GA. Antonisse [5] showed that the implicit parallelism of GA does not depend on the use of binary encoding and recently Radcliffe [136] developed the forma theory which can be seen as a generalization of the schemata theory. This

can be used to explain the good performance of real coded GA, in addition Schmitt [161] developed a comprehensive theory of GA that uses a general-size alphabet. These attempts could be a start of some firm theoretical background for real coded GA.

Practitioners started to use real coded GA [16, 107] despite the lack of theoretical background, some specific operators adapted to the real coded [77, 190, 38] GA were subsequently designed. Providing further performance improvement for real coded GA, resulting in a wider use of this encoding and more development in the domain [32, 140, 85, 31, 66, 119]. This is truly evolution at work on the evolution algorithms.

The real encoding of the gene was implemented in the GA library in conjunction with a definition of the allowable range for the gene. This encoding technique subsequently became the main encoding technique .

6.7.2 History of all the Created Chromosomes

Usually the designer has an extremely vague idea of the design space therefore it can be useful to keep an history of all the created chromosomes. This history can have many interesting uses, the four most obvious uses will be described as follow.

- It is used to prevent the creation of duplicate chromosomes, see section 6.7.3.
- It can be used by the designer to understand better the design space using multi-dimensional visualization tools. From the visualization of this set of data the designer will be able to highlight the different performance ranges of the problem variables. For example if a variable has a narrow performance range or if it has a large performance range or even if it has two or more discrete performance ranges. In addition the

designer will be able to visualize the joint effect of the combined problem variables on the different performance parameter surfaces. However this visualization becomes complicated for high-dimensional problems.

- It can be used to perform screening of the solution, where the nearest neighbor of the potential solution is found in the database and its fitness used to make a decision about the interest of this potential solution. see section 6.11.

- Or it can as well be used as sample data to train a neural network that can be used to perform approximative simulation. Allowing the optimisation speed to be improved, see section 6.11.

6.7.3 Duplicate Prevention

At the end of an optimisation with the SGA there will be a large proportion of duplicate chromosomes in the population. This creates two problems, first it reduces the genetic diversity of the population [27] and increases the premature convergence tendency of the population. In addition Radcliffe's first design principle which was derived from the forma theory [136] states that:

> Ideally, each member of the space being searched should be represented by only one chromosome.

The second problem is that it requires a large number of redundant evaluations because with SGA each new chromosome will be evaluated even though a similar copy of it might exist already and has already been evaluated, creating a loss in performance.

In order to alleviate this performance penalty the duplicate prevention suggested by Davis [27] was implemented. This method will prevent the creation

of a chromosome that duplicates one already created. This checks in the history of all the created chromosomes to try to find a copy of this chromosome.

This can be carried out because engineering designs usually are deterministic, for a set of problem variables there is a single set of problem performance parameters. Duplicate prevention would need to be switched off for non deterministic problems.

6.7.4 Parameters Adaptation

In order to achieve maximum performance not only the most efficient techniques needs to be used but the parameters controlling these techniques needs to be set to their optimal values.

However for certain parameters the optimal values vary during optimisation. For example the mutation operator should operate on a wide range at the beginning of the optimisation to allow a good exploration of the search space. This range should reduce as the the optimisation approaches completion (the exploitation phase) to reduce its disruptivity. Therefore some of the mutation operators that have been implemented use some techniques to dynamically control their operating range.

In addition it is possible to implement operators that posses self adaptive behavior, this mean that without having to set any parameters for the dynamic control of their range these will automatically adapt themselves. This is the case for the SBX operator [33].

6.8 Operators Improvements

The third phase of the performance improvements consisted in the implementation of more advanced and efficient operators. Some SGA operators such as the one point crossover have some deficiencies as described in [140]

(pp296). For constrained optimisation they might be unable to create a valid child chromosome from two valid parent chromosomes. The SGA operators are adapted for the binary encoding and might not provide the optimum performances for real coded genes[77].

The work described in this section includes the implementation of the most efficient operators found in literature as well as in designing novel operators adapted to the type of problem encountered in the optimisation of complex engineering designs.

6.8.1 Mutation Operators

In addition to the traditional random mutation operator described in section 4.2.4 where a gene is randomly selected and its value is changed to a new randomly selected value. The following operators have been implemented.

6.8.1.1 Creep Mutation

The traditional SGA mutation operator directly applied to real coded gene is found to be highly disruptive. The creep mutation described in [27] is an attempt to reduce the disruptivity of the mutation.

A random gene is selected for mutation at a position x in the chromosome $S_v = (v_1, ..., v_x, ..., v_n)$. Where x is a randomly selected integer within the range $[1, n]$. The mutation of the selected gene is limited to a creep range centered around the original gene value in order to create a new value v'_x as stated in 6.20.

$$v'_x = v_x + (2r - 1) * \Delta_{max}) \tag{6.20}$$

$$\Delta_{max} = \delta * (vu_x - vl_x) \tag{6.21}$$

Where Δ_{max} represents the maximum possible size used for the creep mutation, δ represents the range ratio, vu_x & vl_x represents the upper and lower

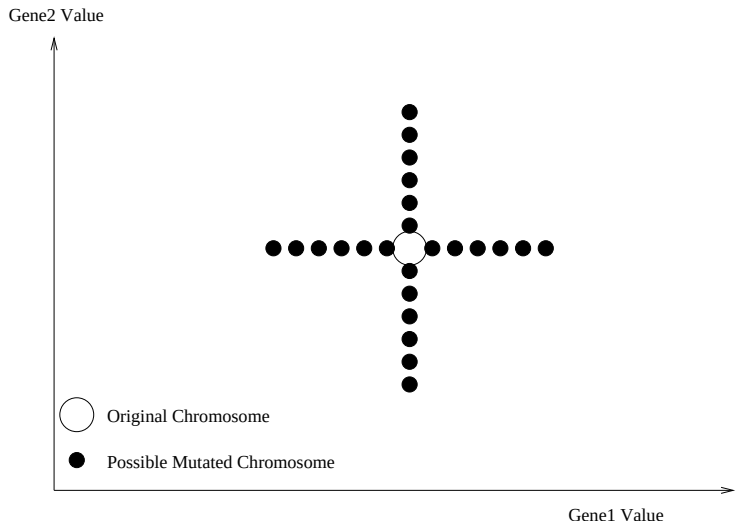

Figure 6.1: Effect of the Creep Mutation operator on a two dimensional chromosome

bounds of the value coded in the gene x and r is a uniformly distributed random real number within the range $[0, 1]$.

In this way the disruptivity of the mutation is controlled by the creep size δ however this creates a tradeoff between the necessity of exploration at the beginning of the optimisation process which would require large values of δ and the exploitation phase during the end of the optimisation which would require small values of δ. Figure 6.1 represents the creep mutation process.

6.8.1.2 Creep Mutation With decay

The creep mutation with decay is an improved version of the creep mutation where a decay rate inspired from the shrinking window mutation in [140] was added to allow adaptation capabilities for the mutation operator. It allows a large value of δ for the beginning of the optimisation during the exploration phase, the value of δ will be decreased during each generation giving a small value of δ at the end of the optimisation during the exploitation phase.

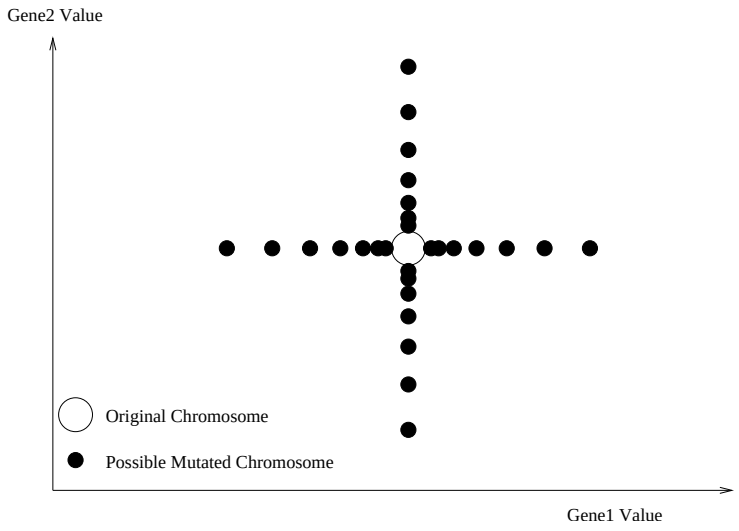

Figure 6.2: Effect of the Creep Mutation With Decay operator on a two dimensional chromosome

Creep mutation with decay reuses the exact same implementation as the creep mutation except that the δ value is altered as follow:

$$\delta_{g+1} = \delta_g * (1 - \gamma) \tag{6.22}$$

where g represents the generation number and γ the creep decay rate. Figure 6.2 represents the creep mutation process and Figure 6.3 represents the evolution of the δ value against the number of generations.

6.8.1.3 Dynamic Vectored Mutation (DVM)

All the operators that have been implemented up to now act on a single gene which means that they have an effect in a single dimension of the solution space. However in engineering design the solution space is usually multi-dimensional and coupled (dimensions are not independent). All the mutation operators that have been implemented up to now operate in a single dimension, which only go along the axis and can not go in a diagonal

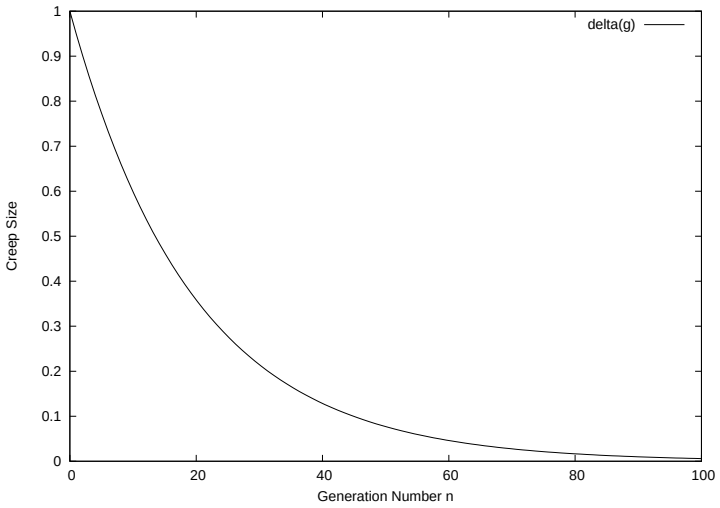

Figure 6.3: Plot of δ against the number of generations for $\gamma = 0.05$

direction, see Figure 6.4, creating unreachable zones, which reduce the GA capacity to explore the the solution space along the directions that are not aligned to the search space.

A new mutation operator the 'Dynamic Vectored Mutation' is proposed that allows mutation in all directions and not only along a dimension axis.

Considering a chromosome containing n genes as an n dimensional vector Vc composed of the n genes. One can create a displacement vector Vd of magnitude m and random direction. The resulting vector Vr of the addition of Vc and Vd will point on the surface of the hyper-sphere around the extremity of Vc as shown in Figure 6.5 for a two dimensional case.

In order to calculate the magnitude m of Vd a method had to be designed. The next step consisted in investigating potential methods to calculate m.

1. The first idea was to define the minimum distance towards the gene boundaries as the maximum allowable magnitude. Then the mutation can be performed with a shrinking magnitude range similar to the creep

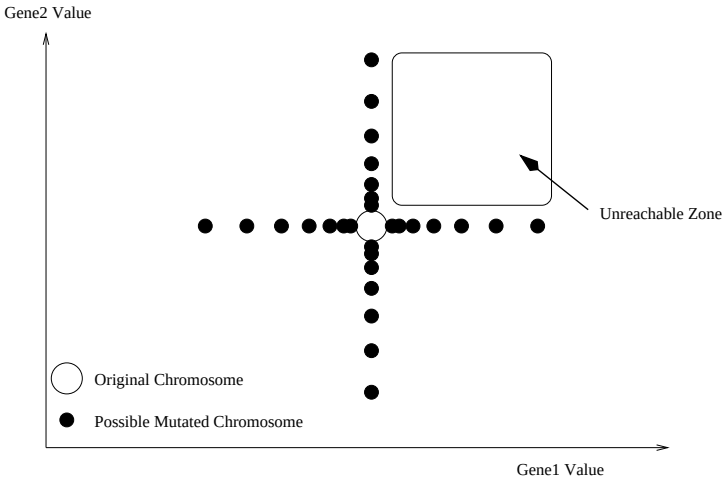

Figure 6.4: Unreachable zones with the traditional mutation operators

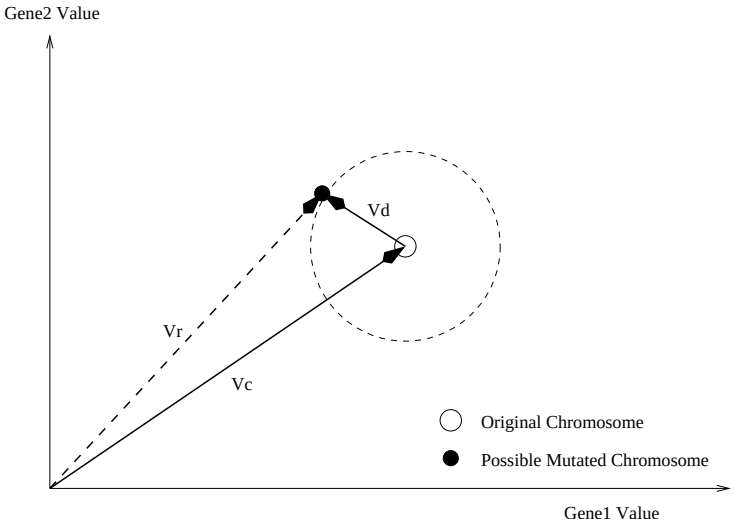

Figure 6.5: Effect of the Vector Mutation in two dimensions

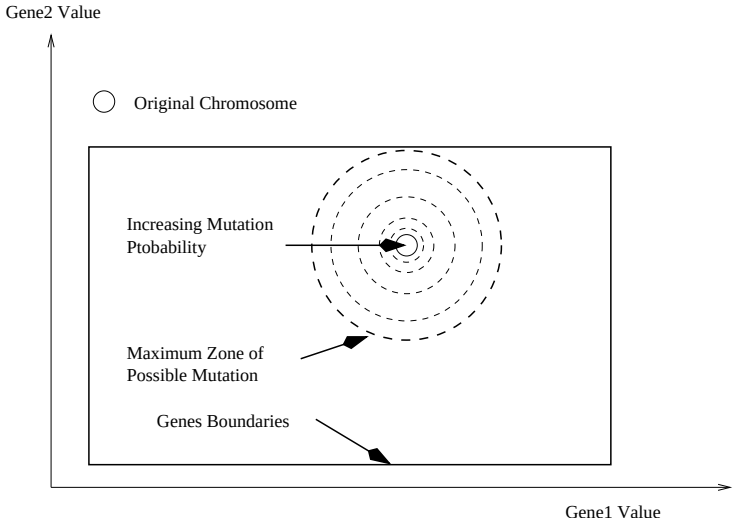

Figure 6.6: Vector Mutation in two dimensions using the maximum distance towards the boundary

mutation with decay and then to create a random vector of the chosen magnitude. This will give a distribution as shown in Figure 6.6. This method was found not to be adequate because it would make the access to the boundary of the domain very difficult. Corners, such as the area 1 in figure 6.7 will be nearly impossible to access.

2. The second idea consisted in first choosing a vector of random direction, then calculate the maximum magnitude Δ_{max} that would result in a solution within the boundaries, and then allocate the magnitude m of Vd with a probability inversely proportional to the fraction of the Δ_{max}, with the magnitude m calculated as follows $m = \delta(\eta) * \Delta_{max}$ were $\delta(\eta)$ is inspired from Janikow & Michalewicz dynamic mutation [77], as defined in 6.23.

$$\delta(\eta) = \left(1 - r^{(1-\eta)^{\gamma}}\right) \tag{6.23}$$

η refers to the completion ratio of the optimisation, the γ factor com-

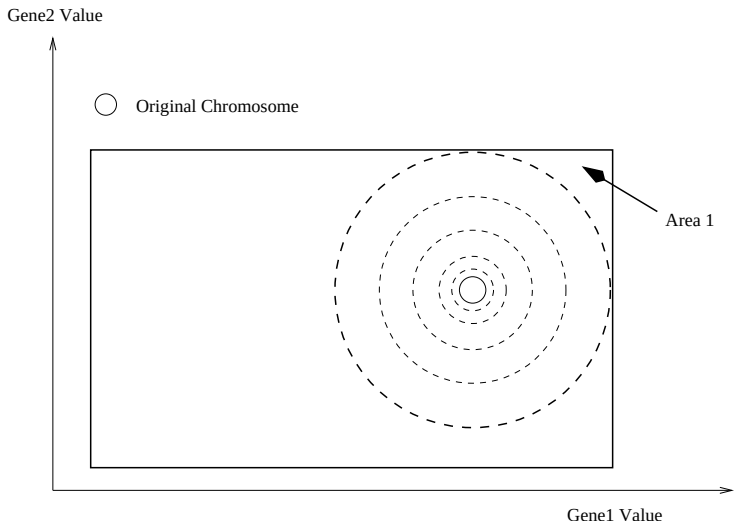

Figure 6.7: Hard to reach zones using the maximum distance towards the boundary

mands the dependency on the completion ratio η. r is a uniformly distributed random number in the range $[0:1]$ The profile of the $\delta(\eta)$ function can be viewed in Figure 6.8. During the start of the optimisation phase there will be an uniform of m . However as the optimisation progresses the probability of m being large will reduces. Resulting in a distribution as shown in figure 6.9. This method reduces the problem of access to the boundary and especially the boundary corners. In addition it has the advantage of having the capacity to explore the whole solution space. However as can be seen in Figure 6.9, this operator is biased towards the center of the domain and against the creation of solutions close to the boundaries, because the probability of creation is not dependent on the distance from the original point.

3. Finally a method was devised based on the idea number two, where calculation of the magnitude m of Vd does not result in a probability of creation inversely proportional of the Δ_{max} but with a probability

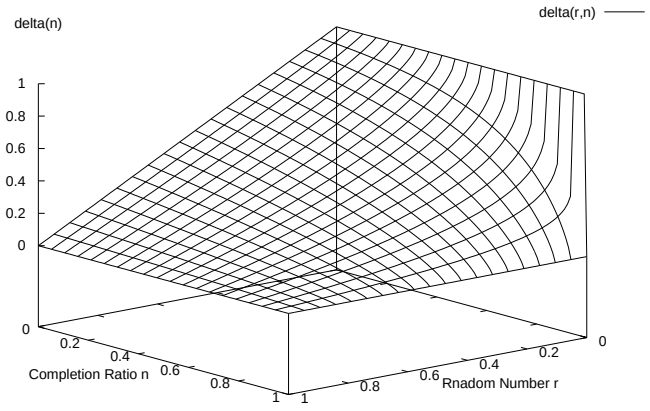

Figure 6.8: Plot of δ against r and η for $\gamma = 0.8$

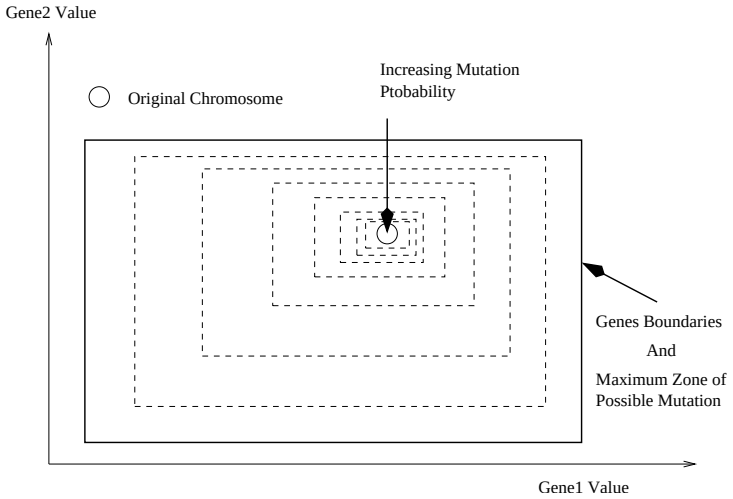

Figure 6.9: Vector Mutation in two dimensions using a probability based on boundaries distance

inversely proportional to its magnitude m and bounded by the maximum allowable magnitude. The magnitude is calculated using equation 6.24.

$$m = a * \overline{\beta}(\eta) \tag{6.24}$$

Were a is the multiplication factor constant. $\overline{\beta}$ was derived by modifying Deb & Agrawal's SBX polynomial probability distribution [32], as defined in equation 6.25.

$$\overline{\beta}(\eta) = \left(\frac{1}{1-\alpha r}\right)^{(1-\eta)^{\gamma}} \tag{6.25}$$

In $\overline{\beta}$ the constant γ is called the iteration dependency factor and will control the curvature of the function depending of η which is the completion ratio of the optimisation, α is defined in 6.26. The profile of $\overline{\beta}$ depending on the random number value r and the completion ratio γ can be seen in figure 6.10. In addition the profile of $\overline{\beta}$ depending on the random number value r and the maximum allowable magnitude Δ_{max} is shown in figure 6.11.

$$\alpha = 1 - (a.\Delta_{Max} + 1)^{-\left(\frac{1}{1-n}\right)\gamma} \tag{6.26}$$

4. This results in a distribution as shown in figure 6.12. The probability of creation will reduce with distance from the original point and the line of iso-probability will be approximately circular, in two dimensions, and approximate an hyper-sphere in n dimensions. It will not be an exact circle or hyper-sphere due to the bounding technique which slightly affects the probability distribution.

This new operator which is based on the third method, satisfies the following requirements:

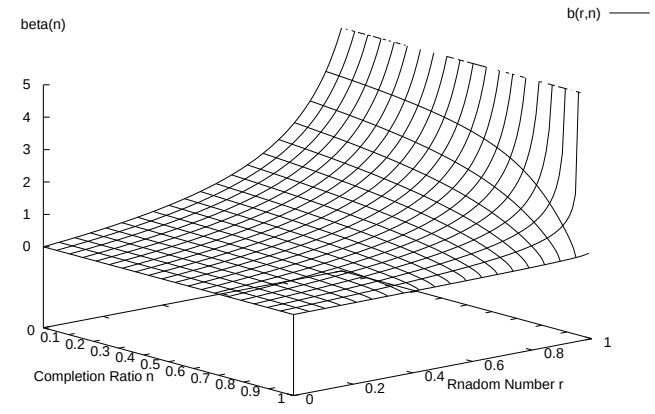

Figure 6.10: Plot of $\overline{\beta}$ against r and η for $x = 1$, $\gamma = 0.5$, $a = 1$.

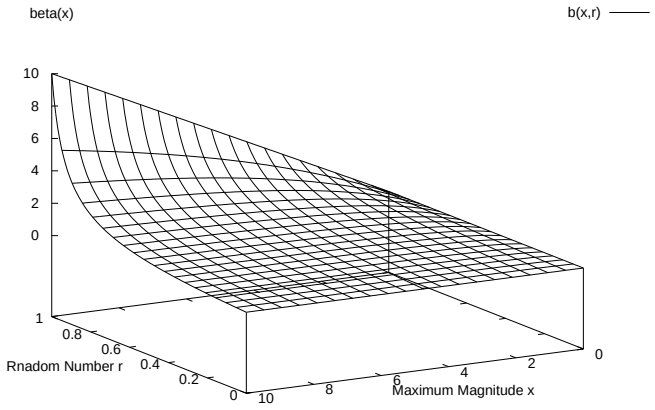

Figure 6.11: Plot of $\overline{\beta}$ against r and x for $\eta = 0.3$, $\gamma = 0.5$, $a = 1$.

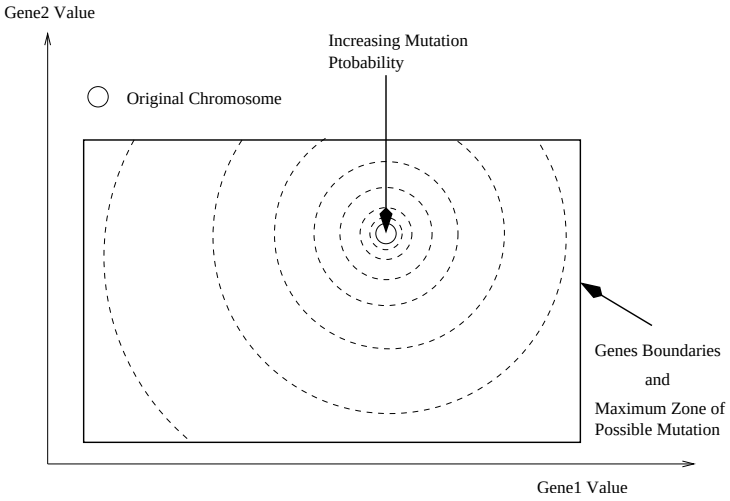

Figure 6.12: Effect Of the Dynamic Vector Mutation using a probability of creation depending of the magnitude m of the mutation vector, bounded by the gene boundaries

- It gives a probability of creation inversely proportional to the magnitude of the vector.
- The probability of creation of a point outside the gene boundary is null.
- It is able to reach the whole search space which satisfies Radcliffe [136] ergodicity criterion.
- It is not biased.

6.8.2 Crossover Operators

The one-point and the n-point crossover operators are not the most adapted for performing crossovers on real coded genes[140, 109]. In addition in [53], Goldberg argues that real coded GA works by first converging to above average points referred to as virtual alphabets, but the convergence to the global optimum gets blocked by the presence of two or more local optima. Since real

coded chromosome were selected as the default encoding for the optimisation, some alternative crossover techniques alleviating the problem of convergence to virtual alphabets and specialized for the use of real coded genes had to be implemented.

One of the simplest method to improve the search power of the crossover is to average or blend the two parents chromosomes to form offsprings, different variations of this blending / averaging idea have been proposed, [27, 38, 190, 140]. Two crossover operators were implemented from this family, in conjunction with a third more advanced crossover method still derived from this family. There are as well other types of crossover that could be implemented which possess different features, some of which use more than two parents to perform the crossover, For example the Unimodal Normal Distribution Crossover (UNDX) [85], the Simplex Crossover (SPX) [66], or the Parent Centric Recombination (PCX) [31].

6.8.2.1 Consanguinity Prevention

One feature that was implemented in all the crossover operators is consanguinity prevention. As it has been seen in section 6.7.3 it is necessary to prevent the creation of duplicate chromosomes. Therefore each crossover operator was modified to prevent the crossover between two copies of the same chromosome which would result in two copies of this chromosome being created. In addition the capacity is there to prevent the recombination between two chromosomes which share the same parents.

6.8.2.2 Weighted Averaging Crossover

For the Weighted Averaging crossover, which is an implementation similar to what Davis proposed in [27], the children are taken as a weighted average of the two parents points. The weighting is performed using random weights.

The production of two child solutions c_1 and c_2 from two parent chromosome $p_1 = (v_1^{p_1}, ..., v_i^{p_1}, ..., v_n^{p_1})$ and $p_2 = (v_1^{p_2}, ...v_i^{p_2}, ..., v_n^{p_2})$ is done as follows .

$$v_i^{c_1} = \frac{1}{2}\left((1+r)\, v_i^{p_1} + (1-r)\, v_i^{p_2}\right) \tag{6.27}$$

$$v_i^{c_2} = \frac{1}{2}\left((1-r)\, v_i^{p_1} + (1+r)\, v_i^{p_2}\right) \tag{6.28}$$

The implementation of these methods considerably improves the search power of the crossover. In addition since it adds diversity by creating children over a continuous space rather than selecting parent points, it reduces the premature convergence effect generated by the use of the n-point crossover applied to real numbers. However it suffers from contraction effects resulting from the fact that the children will always be enclosed between the two parents, which through succeeding generations means the region searched by the population will reduce. To counter this contraction phenomenon an improved operator was implemented, the BLX-α.

6.8.2.3 Blend Crossover BLX-α

The blend crossover BLX was developed by [38]. This operator can be seen as an improved version of the weighted averaging crossover. The improvement consisted in giving the operator some exploration capability through the addition of an exploration factor α which defines the capacity of the crossover to generate offspring outside the line generated between the two parents chromosomes. The crossover is performed as follow:

$$v_i^{c_1} = \frac{1}{2}\left(\gamma v_i^{p_1} + (1-\gamma)\, v_i^{p_2}\right) \tag{6.29}$$

$$v_i^{c_2} = \frac{1}{2}\left((1-\gamma)\, v_i^{p_1} + v_i^{p_2}\right) \tag{6.30}$$

with

$$\gamma = (1 + 2\alpha)r - \alpha \tag{6.31}$$

Where the exploration factor α is in the range $[0 : 1]$ and is chosen to generate the required amount of exploration. A careful selection of α compensate for the contraction effect observed with the weighted averaging crossover. However a new difficulty appears with this crossover technique, the exploration capacity means that this crossover has a probability to create chromosomes outside the upper and lower boundaries of the gene. Therefore it is necessary to test each gene generated for violation of the boundaries, repeating the crossover when violation of the boundary occurs.

6.8.2.4 Simulated Binary Crossover SBX

Deb and Agrawal [32] developed a crossover operator that creates two children from two parents with a probability distribution of the gene values similar to the probability distribution generated by the binary crossover method. It relies on the same basic principle of blending the characteristics of the two parent chromosomes in a manner similar to the BLX-α or the weighted linear averaging. However its uniqueness comes from the definition of a spread factor β and the use of a polynomial distribution function $\overline{\beta}$ to perform the blending between the two parent genes values.

$$\beta = \left| \frac{v_i^{c_1} + v_i^{c_2}}{v_i^{p_1} - v_i^{p_2}} \right| \tag{6.32}$$

$$\overline{\beta} = \begin{cases} (2r)^{\frac{1}{\eta+1}} & \text{if } r \leq \frac{1}{2} \\ \left(\frac{1}{2(1-r)}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \tag{6.33}$$

were η is the distribution index controlling the spread of the $\overline{\beta}$ function. The child solution will be calculated in the following manner:

$$v_i^{c_1} = \frac{1}{2}\left((1+\overline{\beta})v_i^{p_1} + \left(1-\overline{\beta}\right)v_i^{p_2}\right) \tag{6.34}$$

$$v_i^{c_2} = \frac{1}{2}\left(\left(1-\overline{\beta}\right)v_i^{p_1} + (1+\overline{\beta})v_i^{p_2}\right) \tag{6.35}$$

One interesting feature of this operator is its capacity to to create solutions within the whole search space. Thus the SBX always satisfies Radcliffe's ergodicity criterion [136]. Also in the form presented in 6.33 it is not bounded and the child gene values can occur in the range $]-\infty : +\infty[$. This wide range capability can be a problem for engineering design where the search space is usually bounded. Therefore the bounded version of $\overline{\beta}$ was used which has a zero probability of creating a solution outside of the fixed boundaries. $\overline{\beta}$ is then modified as follows:

$$\overline{\beta} = \begin{cases} (2\alpha)^{\frac{1}{\eta+1}} & \text{if } r \leq \frac{1}{\alpha} \\ \left(\frac{1}{2-\alpha r}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \tag{6.36}$$

$$\alpha = 2 - \beta^{-(\eta+1)} \tag{6.37}$$

$$\beta = 1 + \frac{2}{v_i^{p_2} - v_i^{p_1}} \min\left[\left(v_i^{p_1} - v_i^{l}\right)\left(v_i^{u} - v_i^{p_2}\right)\right] \tag{6.38}$$

were v_i^l represents the lower boundary of the gene range and v_i^u the upper boundary and it is assumed that $v_i^{p_2} > v_i^{p_1}$. This modified $\overline{\beta}$ ensures that all the genes of the child will be in the range $\left[v_i^l : v_i^u\right]$alleviating the need to check range violations and regenerate child solutions.

The SBX operator was demonstrated to be more efficient then the BLX-0.5 and the one point crossover on a range of test functions [32]. In addition it is interesting to note that this operator demonstrated self adaptive behavior [33].

6.8.3 Selection Operators

The selection operator plays a critical role during the search in a genetic algorithm Goldberg in [52] has described several selection operators using different techniques. In addition Baker [6]has defined a range of criterion to asses the quality of the different selection techniques.

6.8.3.1 Modified Roulette Wheel Selection

The Roulette-Wheel operator was described in section 5.4.5, it is one of the most widely used selection operator. However it suffers from one drawback, it has a strong tendency to select numerous copies of the best chromosome. This is emphasized in the presence of a super chromosome that can monopolize the selection and create a loss of diversity. In addition it was highlighted in section 6.7.3 that it is required to reduce or prevent the generation of duplicate chromosomes to maximize the efficiency of the algorithm. But if the selected chromosomes for crossover contain a majority of duplicate chromosomes it will be difficult to prevent the creation of duplicate child.

The roulette wheel was modified in order to limit the number of instances of a chromosome that can be selected by removing the chromosome from the wheel after their selection, therefore avoiding the creation of duplicates in the selection process.

6.8.3.2 Stochastic Universal Sampling SUS

The SUS developed by Baker [6] was designed to satisfy the minimum bias and spread criterion. It consists of a similar process to the roulette wheel where each member of the population is represented on the wheel by an area proportional to its relative fitness with the chromosomes randomly ordered around the wheel. Then instead of repeating the process of spinning the wheel and picking a chromosome, a number of pointers equal to the number

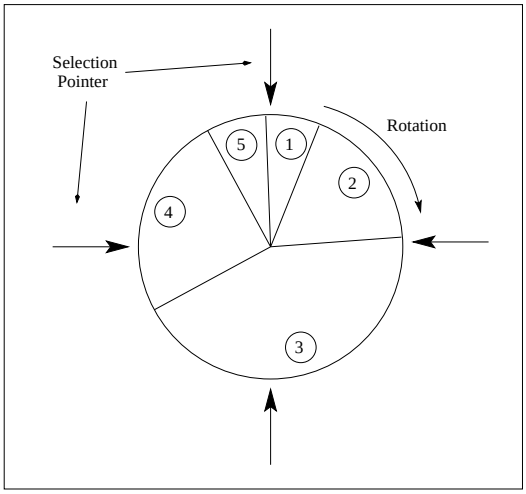

Figure 6.13: Stochastic Universal Sampling Wheel

of chromosomes to select are equally spaced around the wheel. The wheel is spun and the chromosomes in front of the pointers are selected. This minimizes the bias and drift connected with the repeated spinning of the wheel. Figure 6.13 shows the SUS wheel.

6.8.4 Replacement Operators

The constraint handling of the the two traditional replacement operators, the tournament replacement and the ranked replacement have been improved. They have been modified in order to support constraint based niching using the following principles:

- A chromosome satisfying all the constraints should be preferred over one not satisfying all of the constraints.
- Of two chromosomes not satisfying all the constraints the one that satisfies the most the set of constraints should be preferred.

These operators use a constraints satisfaction factor that defines how well the constraints are satisfied. This factor is used for controlling the ranking and the tournament. It takes the place of the fitness when the constraints are not satisfied.

6.9 Hybridation with other optimisation techniques

Although genetic algorithm is an extremely efficient optimisation technique it is not the most efficient in all the search phases. its performances may therefore be improved by hybridizing it with other optimisation techniques which performs better in a given search phase.

6.9.1 Random Search Phase

Goldberg stated that the quantity of good building blocks generated during the initial generation of the population is critical for the performance of GA [52]. Therefore as suggested by Davis [27] a random search phase was added at the start of the optimisation. Looking at a large number of individuals and only selecting the best to create the initial population will give a good quality initial population. The population is therefore free of the failing chromosomes which might arise during the early stages of the optimisation due to the incapacity of the simulation tools to cope with certain combination of inputs.

6.9.2 Hill Climbing

Researchers [27, 109, 52] have been showing the weakness of GA in refining the position of the optimum point once the global optimum region has been

found. Therefore it becomes interesting to add a hill climbing phase at the end of the optimisation in order to refine the quality of the optimum point.

6.10 Multi Processing

For engineering optimization such as combustor optimisation or wing profile design the evaluation of the fitness function is costly in terms of CPU time due to the high computational requirement of the simulation tools. Therefore the optimisation can be significantly shortened by using many processors in parallel in order to evaluate all the chromosomes. Two ways of parallelizing the evaluation have been implemented depending on the available hardware as follows.

6.10.1 Parallel Processing

A very simple parallelization method was developed to run on multi processor computers. This method consists of separating the evaluation sub processes and letting the operating system of the computer run these sub processes independently. This is a very simple and efficient method of parallelizing the evaluation, however the availability of these computers is relatively limited due to their high purchasing cost compared to the traditional desktop workstations.

6.10.2 Heterogenous Distributed Processing

A very efficient way to reduce the computational overhead of recurrent runs of CPU intensive simulation software is to perform the execution of these applications using Heterogeneous Distributed Computing [171]. This consists of distributing the execution of processes over a network of computers which might not all be of the same architecture. In the case of the distribution of the

multiple evaluation of the fitness function through the parallel execution of multiple sequential analysis code, communication only takes place between the GA and the analysis software. This was developed in a simpler form in [151] and then improved for this problem. The technical details of the implementation have been described section 3.8 of the toolbox modules.

Heterogenous Distributed evaluation of the chromosomes allows them to be evaluated in parallel over a distributed network of computers such as a local network, a computer Beowulf cluster, or even across the Internet. There is no requirement for the computers to be symmetric (all the same) this can even be performed on computers running different operating systems.

6.11 Reduction of the Number of Exact Evaluation

The last performance improvement technique that will be discussed here offers an interesting possibility of reduction of the duration of an optimisation however the techniques described in this section have not been implemented contrary to the previous sections. This is due to the fact that these techniques are suited for problems requiring highly computationally expensive simulations such as FEM or CFD which is not yet the case for preliminary design tools.

The idea common to these techniques is that a lot of the simulations performed during the optimisation are not very useful, therefore the two techniques that will be described next try to reduce the number of simulations performed.

6.11.1 Screening

Screening consists of using a library of design points to assess the interest of a newly created chromosome [140] by defining whether it lies in an area of low performance in which case the chromosome will be discarded. The library is composed of a large number of chromosomes representing the design points encountered up to now. When a new chromosome is created the closest neighbors from the library are found and if their fitness is lower than a defined value the chromosome will not be evaluated. The value could be the fitness of the worst chromosome of the population or a predefined value. This screening technique could save some time avoiding the evaluation of chromosome from a very bad area of the search space. However it generates a computational overhead by having to compare the new chromosome with all the chromosomes in the database, although this is rather small for normal conditions it becomes important for highly dimensional problems due to the large number of dimensions to compare and the large number of samples that the library must contain to represent the search space.

6.11.2 Neural Network Evaluation

Another technique which can dramatically reduce the number of exact evaluations, consists of using a neural network to perform an inexact evaluation of the chromosomes [126]. The population is pre-evaluated using the neural network. Then a small proportion of the population, the most promising solutions, are evaluated through the normal (exact) evaluation method [50]. These exact evaluations are then used to retrain the NN to improve its accuracy. This technique greatly reduces the number of exact evaluations. However the reduction in the number of evaluation can be offset by the cost of training the neural network when the computational cost of the direct evaluation is not very high.

In addition an interesting feature of the NN can be used to improve even

further the performance of the optimisation. Once trained the NN is a curve fitting representation of the solution space. This might allow analytical optimisation techniques, based on derivatives to find the optimum of the neural network [128].

6.12 Conclusions

Within this chapter the implementation of a robust and efficient optimisation library was related . This implementation was performed through the development of a Java based SGA library, the creation of novel techniques to handle efficiently a large number of objectives and constraints, and extensive transformation of the SGA library in order to implement the state of the art methods and operators and some novel ones specifically developed for this application.

During the next chapter the techniques presented here and implemented in the optimisation library will be tested against analytical problems of known optimum and difficulty in order define their efficiency and tune their parameters.

Chapter 7

Testing of the Optimisation Technique

7.1 Introduction

The aim of this Chapter is twofold, its primary purpose is to demonstrate the optimisation performance of the optimisation library for different analytical problems with known optimal values. The secondary aim of this chapter is to provide an insight into the behavior of these optimisation techniques for the reader who wishes to use these techniques. The following sections will describe the tests that have been performed in order to verify the performance of the optimiser and provide some understanding on the behavior of the different operators and of the optimisation technique applied to a range of different problems.

7.2 Mutation Operators analysis

The aim of this section is to examine the behavior of the new mutation operator proposed in section 6.8.1.3, the Dynamic Vector Mutation compared

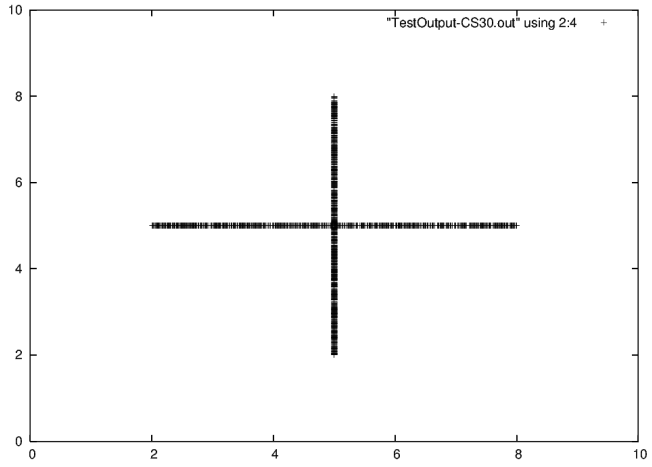

Figure 7.1: Operator: Creep Mutate With Decay, (Creep Size = 30)

to the more conventional Creep Mutate With Decay operator section 6.8.1.2.

The test consists of mutating a population of individuals through the different mutation operators and observing the distribution of the newly generated chromosomes. All the runs were performed using two dimensional genes, a population composed of 2000 individuals and the boundaries of all the genes were $[0 : 10]$. The position of the genes of the initial population is set to 5.

7.2.1 Creep Mutate With Decay

The first operator to be tested was the Creep Mutate With Decay. Figure 7.1 shows the new chromosomes generated by this operator for the start of an optimisation (Progress Ratio = 0.0), where the creep size is 30% of the genes range. Figure 7.2 shows the chromosomes generated with a Creep Size of 10% which is typical from the middle of an optimisation run (Progress Ratio = 0.5). Finally figure 7.3 shows the chromosomes generated with a Creep Size of 1% which is typical from the end of an optimisation run (Progress Ratio = 0.98).

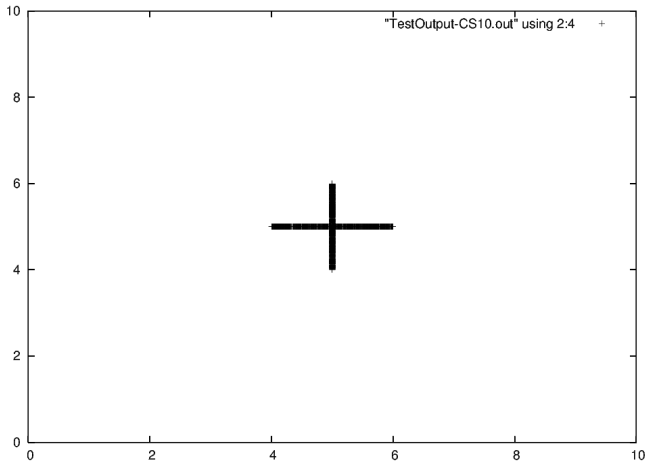

Figure 7.2: Operator: Creep Mutate With Decay, (Creep Size = 10)

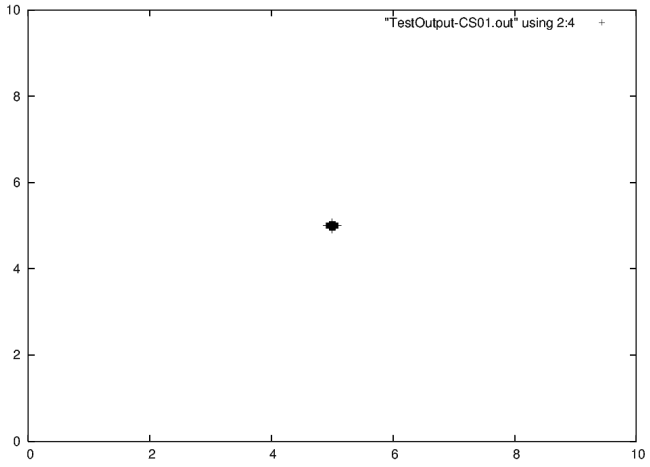

Figure 7.3: Operator: Creep Mutate With Decay, (Creep Size = 1)

From these three figures (7.1, 7.2, 7.3) it can be observed that the new chromosomes form a cross shape, this due to the fact that mutation traditionally affects only one gene at a time. In addition it can be observed that the distribution of the chromosomes on the axis is relatively uniform.

7.2.2 Dynamic Vector Mutate

The Dynamic Vector Mutate operator was tested for the same conditions. However this operator is more complex and has two additional parameters:

- The Iteration Dependency Factor, which controls the compression rate of the cloud of points as a function of the Progress Ratio of the optimisation, increasing this value will increase the concentration rate. The value of this parameter was set to 1.5.
- The Initial Distribution Factor, which controls the initial size of the cloud of points, increasing this value increases the original concentration of the points. The value of this parameter was set to 0.5.

Figure 7.4 shows the new chromosomes generated by this operator for the start of an optimisation (Progress Ratio = 0.0). Figure 7.5 shows the chromosomes generated during the middle of an optimisation run (Progress Ratio = 0.5). Finally figure 7.6 and 7.7 shows the chromosomes generated towards from the end of an optimisation run with a Progress Ratio of respectively of 0.8 and 0.98.

From figure 7.4 one could observe that the distribution of the points does not follow the cross shape any more and forms a cloud covering the whole range of the genes. The fact that the points cover the whole range of genes confirms that the The Dynamic Vector Mutate operator satisfies Radcliffe [136] ergodicity criterion as described in section 6.8.1.3.

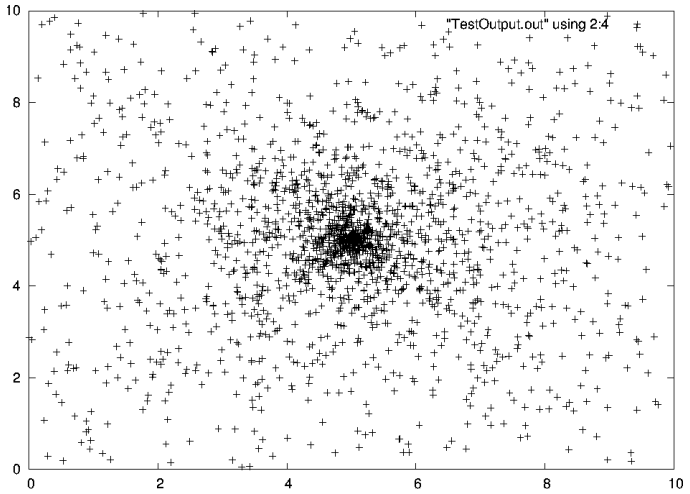

Figure 7.4: Operator: Dynamic Vector Mutate (Progress Ratio = 0.0)

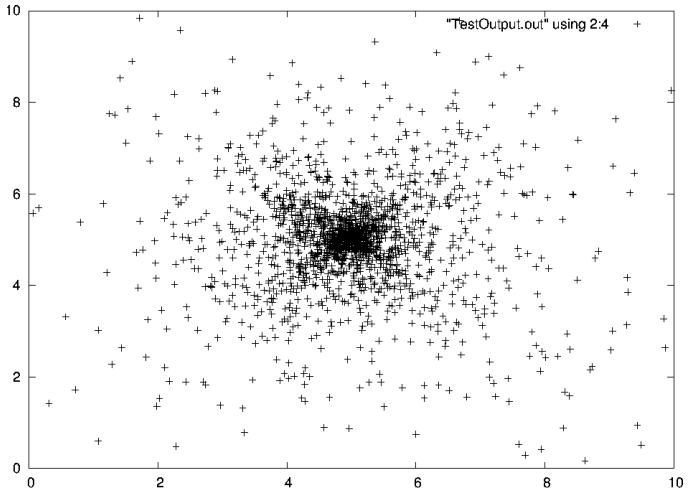

Figure 7.5: Operator: Dynamic Vector Mutate (Progress Ratio = 0.5)

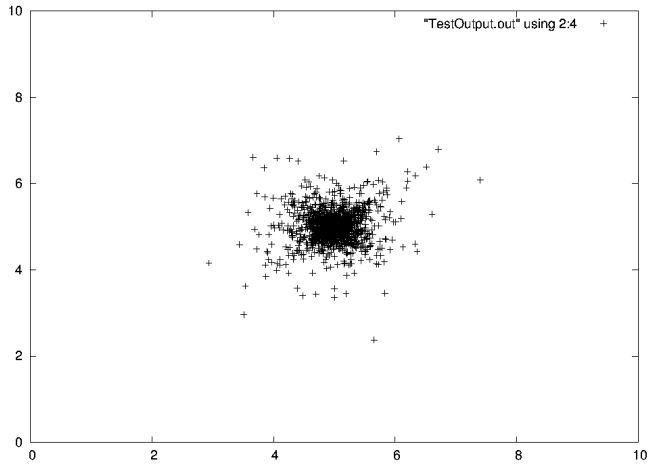

Figure 7.6: Operator: Dynamic Vector Mutate (Progress Ratio = 0.8)

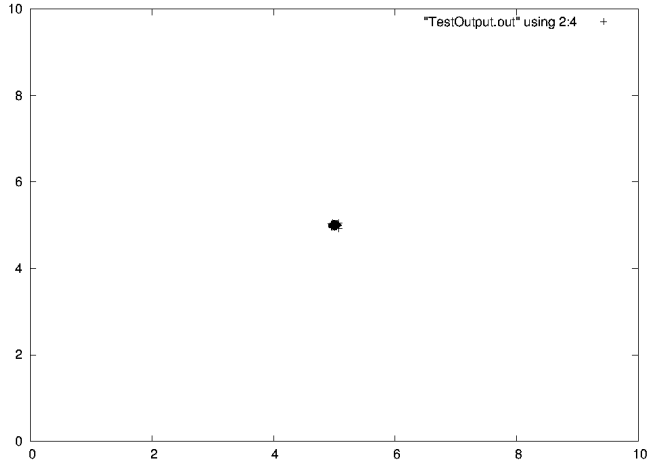

Figure 7.7: Operator: Dynamic Vector Mutate (Progress Ratio = 0.98)

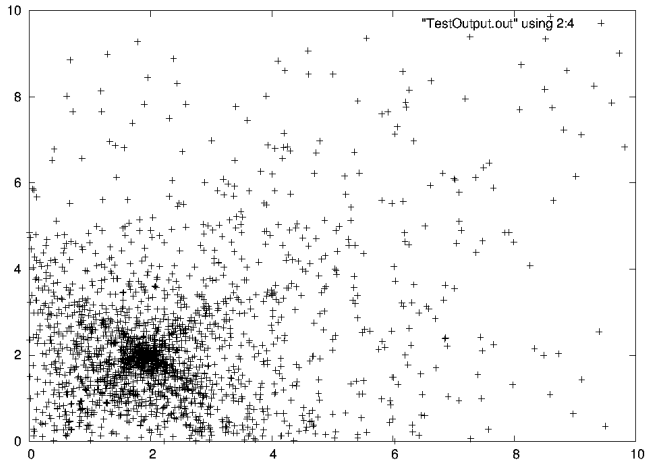

Figure 7.8: Operator: Dynamic Vector Mutate close to a boundary (Progress Ratio = 0.0)

In addition it can be seen that the distribution of the points is not uniform and that the probability appears to be inversely proportional to the distance of the original chromosomes. As the optimisation approaches completion, from figure 7.5, 7.6 and 7.7 it can be seen that the cloud contracts to ultimately form a point where only minimum variations occur.

It is also interesting to observe the behavior of this operator when the original gene is close to the gene boundaries. Figures 7.8, 7.9 and 7.10, shows the distribution of the points for the case where the original gene value is 2.

On figure 7.8 it is possible to observe that the points are more concentrated in the space close to the boundaries, this is due to the fact that the same number of points need to be created in a smaller surface. However the distribution is still dependent on the distance towards the original position. It is interesting to note that the points are not attracted nor repulsed by the boundaries. The absence of bias concerning the boundaries is as expected in section 6.8.1.3.

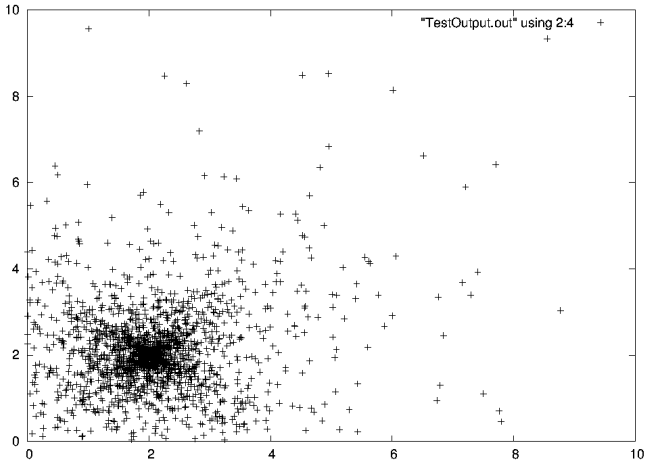

Figure 7.9: Operator: Dynamic Vector Mutate close to a boundary (Progress Ratio = 0.5)

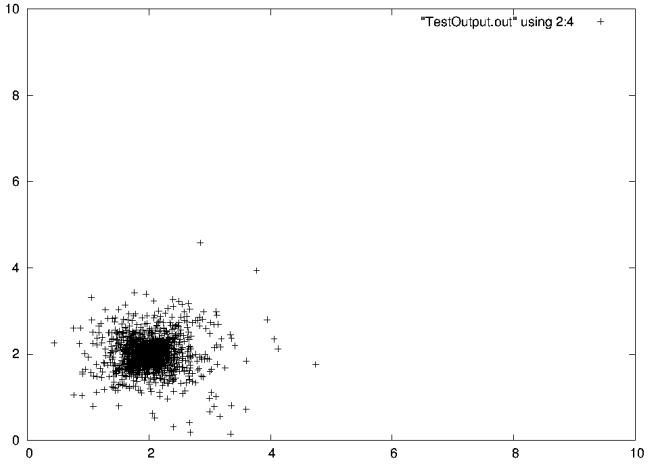

Figure 7.10: Operator: Dynamic Vector Mutate close to a boundary (Progress Ratio = 0.8)

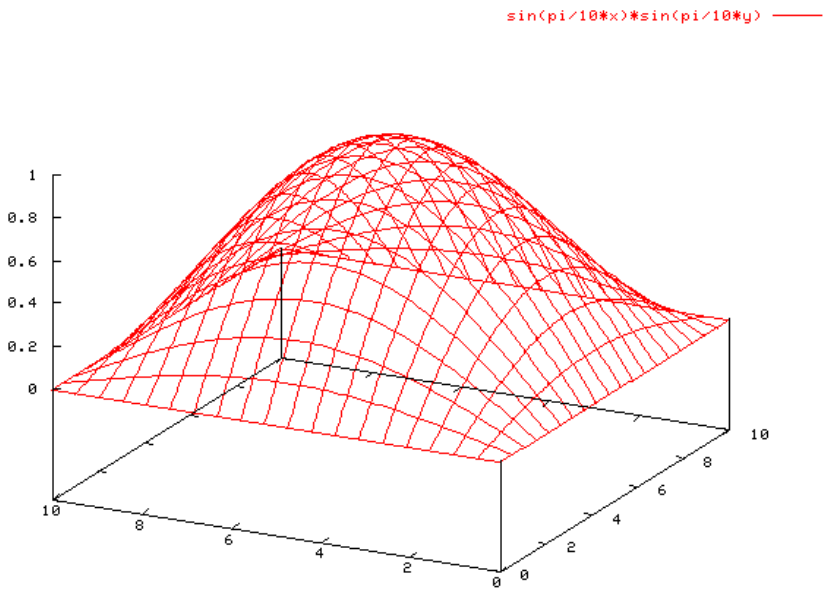

Figure 7.11: Simple Function in 2-dimensions

7.3 Optimisation of a Simple Function

The aim of this section is to test the capacity of the optimiser to find the optimum of a simple function with no local optimum. This series of tests verifies the operation of the different operators and their efficiency for this type of problem.

This simple function needs to have a single optimum, it was decided to maximise a sine wave in the range $[0{:}\pi]$. The function is expressed in 7.1 and can be viewed in figure 7.11, for two dimensions and an arbitrary range r.

$$f(x,y) = sin\left(\frac{\pi}{r}x\right) * sin\left(\frac{\pi}{r}x\right) \tag{7.1}$$

This function gives a single optimum of value 1 located at the center of the

given range. It will be used to setup the optimal control parameters, to define the relative quality of the operators, and to test the capacity of the optimiser to optimise high dimensional problems.

7.3.1 Definition of the optimum control parameters

GA processes include a number of control parameters that can be tuned to achieve optimal performance. Although genetic algorithms are quite sturdy and are able to generate an optimised solution for a very wide range of parameter settings, tuning the parameters of the algorithm can allow dramatic performance improvements.

There is no specific rule to define the optimum parameters, these have to be determined empirically. It is even possible to use a GA as a meta-algorithm to derive the optimum control parameters, see Grefenstette [59]. Some guidelines for the values of the control parameters can be found in literature [27, 35, 64, 54, 56], these suggest the following setup guidelines:

- Initial Population Ratio: This parameter controls the amount of random search performed during the initial phase of the optimisation. There are no guidelines for the setting of this parameter.
- Population Size (P): The population should increase with the complexity of the problem. If the population is too small convergence to a local optima may occur, if the population is too large loss of performance may occur.
- Crossover Probability (CP): Values between 0.1 to 0.6 have been reported to perform well
- Mutation Probability (MP): Values between 0.0 and 0.3 have been reported to perform well, increasing the mutation probability may reduce premature convergence problems.

- Selection Pressure (SP): The value of the selection pressure has to be defined such that the GA stays within the failure boundaries defined by Goldberg [56].

- SBX Distribution Index: This variable controls the spread of the SBX crossover operator, Deb [32] suggested that the optimal value is 1.

- Initial Distribution Factor (IDF0): This parameter controls initial spread of the DVM mutation operator. The allowable range for this operator is $[0 : 1]$.

- Iteration Dependency Factor (IDF1): This is a parameter of the DVM mutation operator. It controls the contraction of the distribution created by the DVM depending of the completion ration of the optimiser. The allowable range for this operator is $[0 : +\infty]$ however expected values would be between 0 and 3.

Tests were performed to define the optimal value of the above parameters, starting with IDF0 and IDF1. The tests consisted of determining the efficiency of the optimisation algorithm depending on the value of the different parameters. Here the efficiency was determined as the number of evaluations required to obtain an optimal solution within 1% of the known optimal solution. Since GA optimisation is a stochastic process the number of evaluations is averaged over a number of runs of the optimisation process (between 100 and 500 depending on the cases) . The plots in addition to the test points show a curve fit to highlight the trends. The parameter tuning was first performed on the parameters for which there was the highest uncertainty in the initial estimate.

- Dynamic Vector Mutation Settings: For IDF0, figure 7.12 shows that the optimal setting is situated between 0.75-0.8, and for IDF1 figure 7.13 shows that the optimal setting is situated around 1.6. Once the

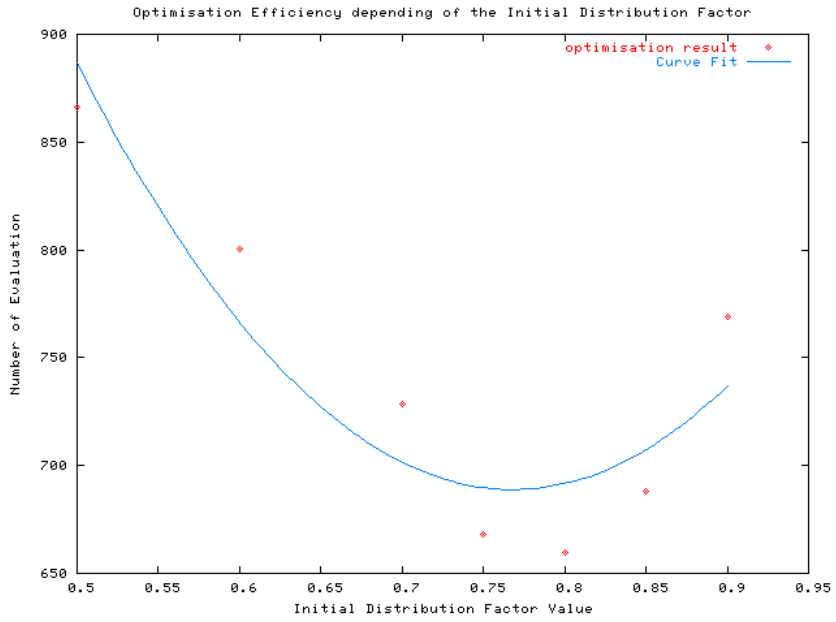

Figure 7.12: Effect of the Initial Distribution Factor (IDF0)

tuning of the mutation operator has been performed and its behavior understood, the crossover and mutation probabilities can be tuned using the same principle.

- Crossover Probability Setting: In figure 7.14, it is interesting to note that for the crossover probability the efficiency becomes relatively constant past 0.15 until 0.3. This is an interesting fact which means that the CP can be freely selected between 0.15-0.3 depending on the convergence characteristics of the problem.

- Mutation Probability Setting: Figure 7.15 shows the effect of the mutation probability on the performance of the optimiser. The observation of this graph shows that, as expected the low but non zero values of mutation probability give the best performance. In addition in figure 7.16, which represents a plot of the low values of the MP, it can be ob-

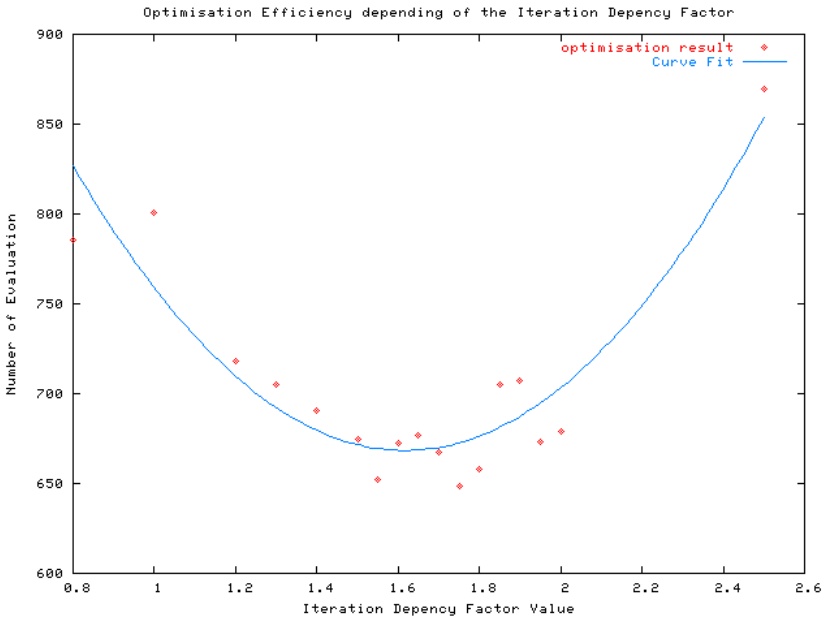

Figure 7.13: Effect of the Iteration Dependency Factor (IDF1)

served, that for very low MP values the the spread of the optimisation performance result is large and can be clearly seen in this graph even though each point is the average of 250 optimisation runs. This can be explained theoretically, the mutation operator is disruptive to the formation of building blocks [54], however it is capable of providing new building blocks which help to maintain the necessary diversity. With these considerations in mind the mutation probability was chosen in the highest value of the zone of optimal performance (0.005-0.01 see figure 7.16).

- Selection Pressure Setting: From the observation of the graph in figure 7.17, the optimal selection pressure range appears to be from 3 to 10. However since a high selection pressure has a tendency to favor premature convergence. The lower value of the range has been selected to provide a more generic setting.

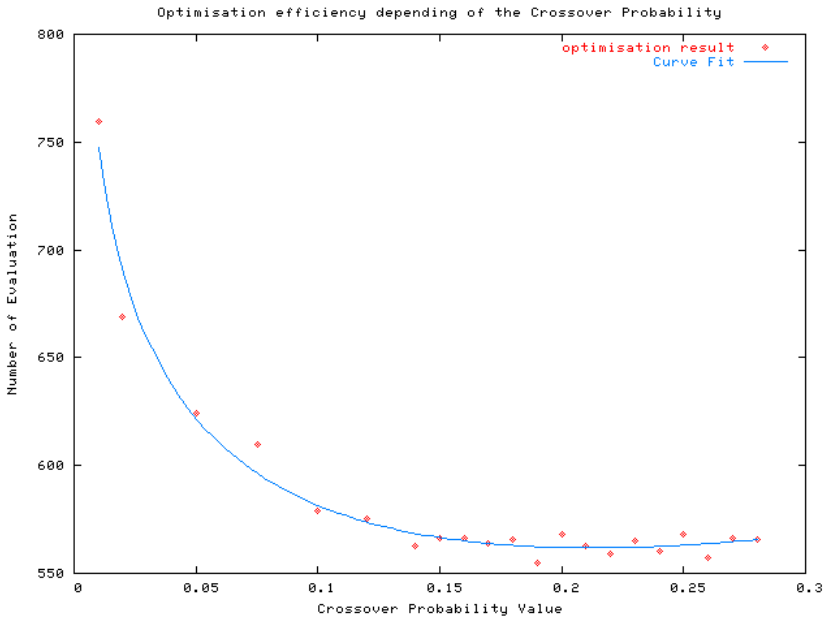

Figure 7.14: Effect of the Crossover probability (CP)

- Population Size and Initial Population Setting: Last but not least the optimal population size and the optimal initial population ratio have to be determined. Figure 7.18 show a plot of the effect of the population size on the the performance of the optimisation. It shows an optimal some for population composed between 25 and 40 individuals, a population size of 30 was selected. Concerning the ratio controlling the size of the initial population, figure 7.19, the optimal value range appears to be between 1.3 and 2 times the nominal population size Even though the data suffers a large spread. However for this type of optimisation the benefits from a large initial population are limited. Benefit will appear on more complicated problems where a good initial population has a higher importance, and on problems were the evaluation method might suffer from a high failure rate during the initial stages of the optimisation.

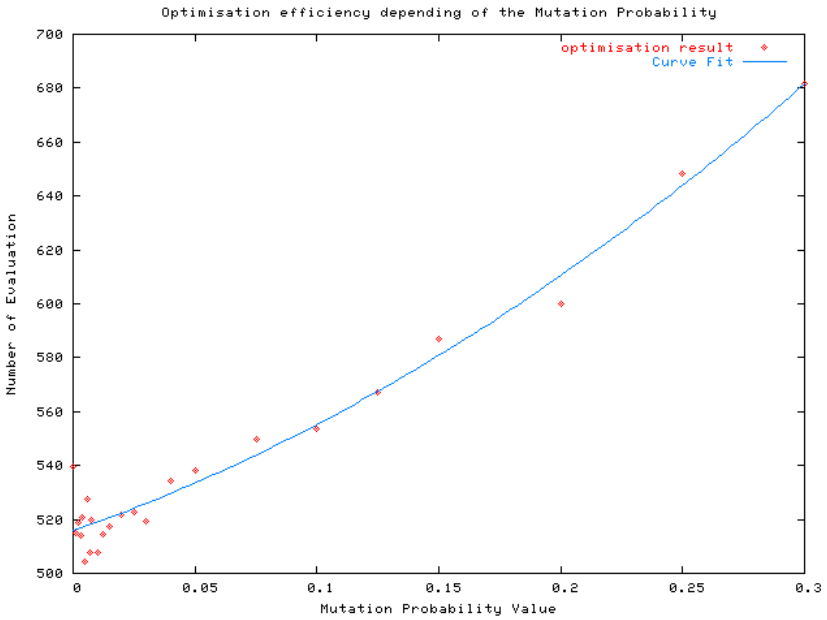

Figure 7.15: Effect of the Mutation Probability (MP)

The optimiser demonstrated reasonable performance, to achieve a solution within 1% of the optimal solution of the simple function test problem in 5 dimensions, The average number of evaluations required to complete the optimisation was close to 400 with a few of the best optimisations completing in less than 200 evaluations. This series of tests has provided some insight into the effect on each parameter on the performance of the optimiser. The control parameter values defined in this section are summarised in Table 7.1. These give near optimal performance for this problem and are generic enough to provide reasonable performance for other cases with a limited need for re-tuning.

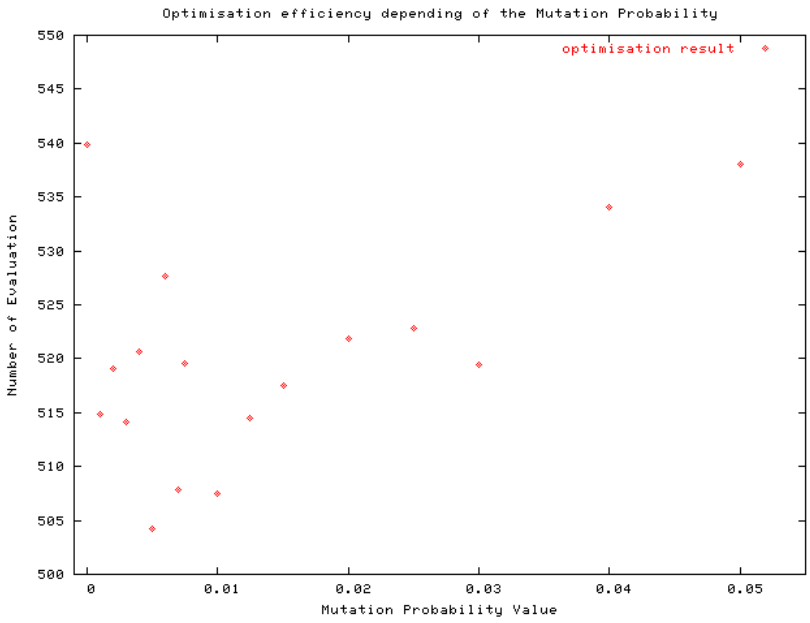

Figure 7.16: Effect of the Mutation Probability (MP) in the low Values Range

Optimisation Parameter	Value
Population Size	30
Initial Population Ratio	1.3
Crossover Probability	0.2
Mutation Probability	0.01
Selection Pressure	3
Distribution Index (SBX)	1
Initial Distribution Factor (DVM)	0.775
Iteration Dependency Factor	1.6

Table 7.1: Final setting of the optimiser for the optimisation of the Simple Function in 5 Dimensions

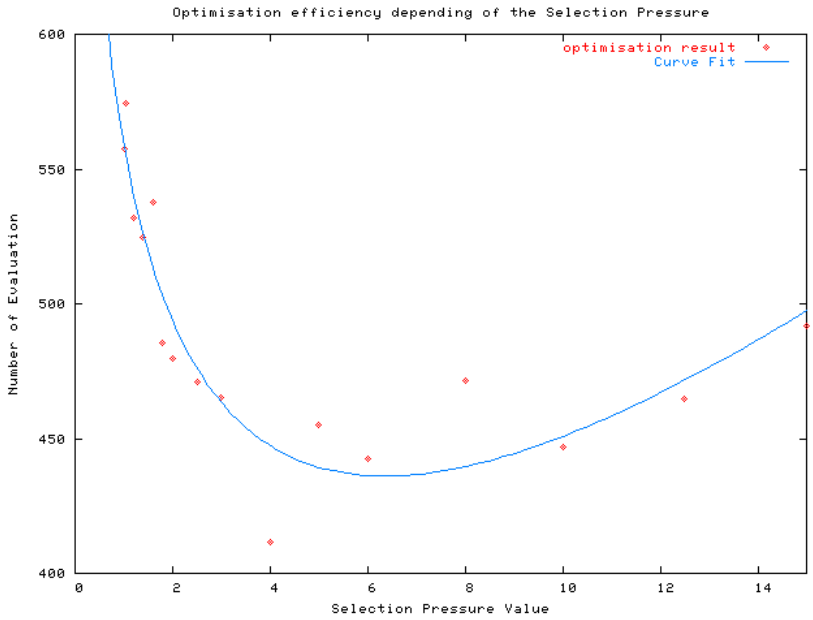

Figure 7.17: Effect of the Selection Pressure (SP)

7.3.2 Test of different operators

The different operators described in chapter 5 have been tested in order to define their relative qualities. The same test consisted of comparing the average number of evaluations to achieve 1% error towards the optimal solution of the simple function problem in 10 dimensions. Results were averaged over 100 optimisations.

The effect of the different Crossover operators on the performance of the optimisation can be observed in table 7.2, there is a clear performance increase between the Weighted Averaging Crossover and the BLX-α operators. Use of the SBX operator allow a further reduction in the average number of evaluations.

The result for the different crossover operators can be found in table 7.3, there the effect of the diverse improvements to the operator can be observed.

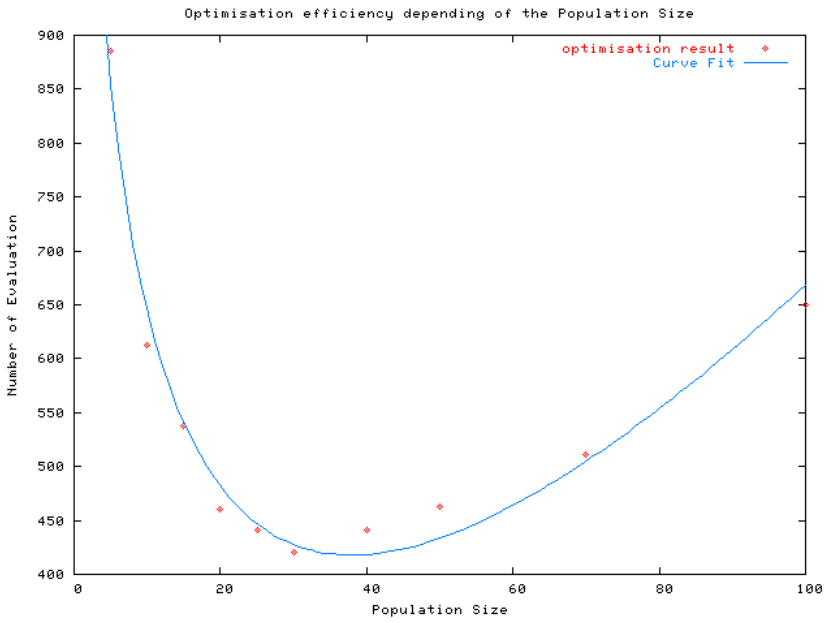

Figure 7.18: Effect of the Population Size (PS)

Crossover Operator	Number of Evaluation
WeightedAveragingCrossover	2552.41
WeightedLinearCrossover (BLX-α)	1556.46
BoundedSBXCrossover	1149.9

Table 7.2: Performance of the different Crossover operators

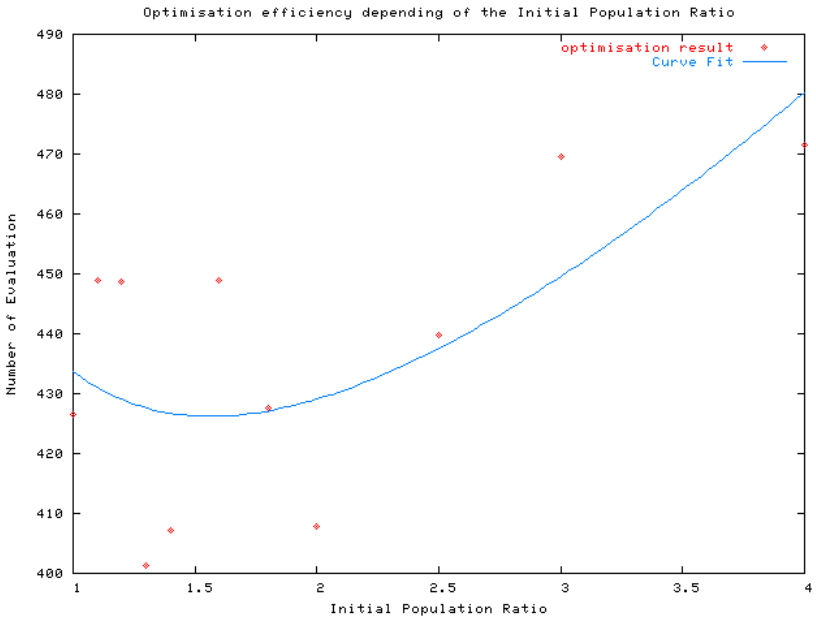

Figure 7.19: Effect of the initial Population Ratio

The proposed operator (DVM) comes out as the most efficient one. However the improvement of all the mutation operators is limited, this is mainly due to the fact that in this simple problem the mutation probability is relatively low.

Mutation Operator	Number of Evaluation
RandomMutate	1366.32
CreepMutate	1225.56
CreepMutateWithDecay	1181.54
DynamicVectorMutate	1163.86

Table 7.3: Performance of the different Crossover operators

7.3.3 Dimensionality effects

Combustor design has a lot of variables that need to be tuned, therefore it is important to determine how the optimiser scales with the dimensionality of the problem. Dimensionality increases cause serious difficulties mainly due to the exponential increase of the search volume.

For problems like the simple function if a range of 1% is desired for a valid solution, the ratio $\mathcal{V}/\mathcal{S}$ of the valid volume $\mathcal{V}$ over the search space volume $\mathcal{S}$ can be used to define the difficulty of the problem. From equation 7.1 assuming a range $[0 : \pi]$. The valid range to achieve 99% of the optimal value which is 1 can be expressed as follow

$$r = \frac{\pi}{2} - asin(0.99) \tag{7.2}$$

Therefore the valid volume $\mathcal{V}$ can be expressed as $\mathcal{V} = 2r$ in one dimension,and the search space as $S = \pi$. For a n-dimension the valid space becomes the volume of an hyper-sphere of radius r. In the case of an even number of dimensions the volume of hyper-sphere containing the valid solution is given in 7.3 and the solution volume will be expressed as the volume of an hypercube given in 7.4.

$$\mathcal{V} = \frac{\pi^{\left(\frac{n}{2}\right)}}{\left(\frac{n}{2}\right)!} r^n \tag{7.3}$$

$$S = \pi^n \tag{7.4}$$

From 7.2, 7.3 and 7.4, the ratio $\mathcal{V}/\mathcal{S}$ can be calculated for different dimensions. It decreases exponentially as n increases. $\mathcal{V}/\mathcal{S} = 0.09$ in one dimension, there is nearly one chance out of ten for a random individual to be valid. In a 30-dimensional search space $\mathcal{V}/\mathcal{S} = 8.9 * 10^{-46}$, the problem has become exponentially harder.

Optimisation has been performed for the simple function in a number of

Number of Dimensions n	3	5	10	20	30
Number of Evaluations (Avg)	59.85	401.25	1131.91	2430.20	3909.9

Table 7.4: Scaling of the optimisation technique depending of the dimensionality of the problem

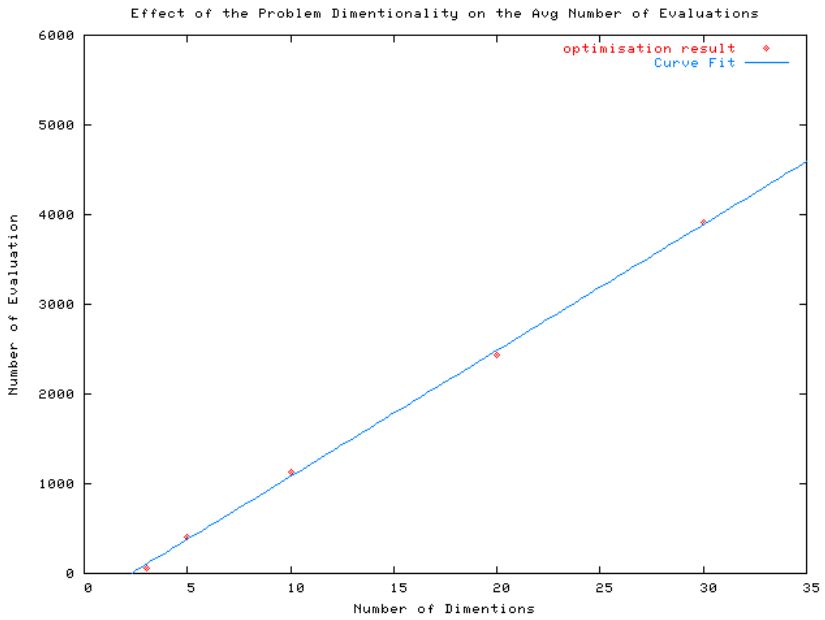

Figure 7.20: Dimensionality effect on the required number of evaluations

dimensions, with the same validity range. The optimisations used the settings defined in 8.2.2 except for the population size which was varied depending upon the dimensionality of the problem. Table 7.4 and figure 7.20 presents the scaling of the average number of evaluations required to achieve the valid range depending on the number of dimensions.

In figure 7.20 it can be observed that the required number of evaluations increases linearly with respect to the number of dimensions and can be approximated for dimensions > 2, as $N = a * n + b$ where $a = 140.447$ and $b = -323.457$. The fact that the optimiser scales linearly with the number

of dimensions (problem variables) is a very interesting advantage compared to other optimisation methods.

7.4 Optimisation with Local Optima

The aim of this section is to test the capacity of the optimiser for problems containing multiple local optimum. To define its capacity to avoid the local optimum and find the global optimum.

For this purpose a function was specially designed, the Hedgehog function, which provides a single global optimum and a number of local optimum that can be varied. This function may be used in any n-dimensions. In two dimensions the Hedgehog function can be expressed as 7.5 which will give the search space shown in figure 7.21.

$$f(x,y) = \left(\cos\left(\frac{\alpha\pi}{r}(x-\delta)\right)\cos\left(\frac{\alpha\pi}{r}(y-\delta)\right)\right)^2 * e^{\left(-s\left((x-\delta)^2+(y-\delta)^2\right)\right)} \quad (7.5)$$

Where δ represents the location of the global optimum in this case $\delta = 5$, r represent the gene range, α represents the number of optima in each direction, and the factor s represents the steepness of the peaks.

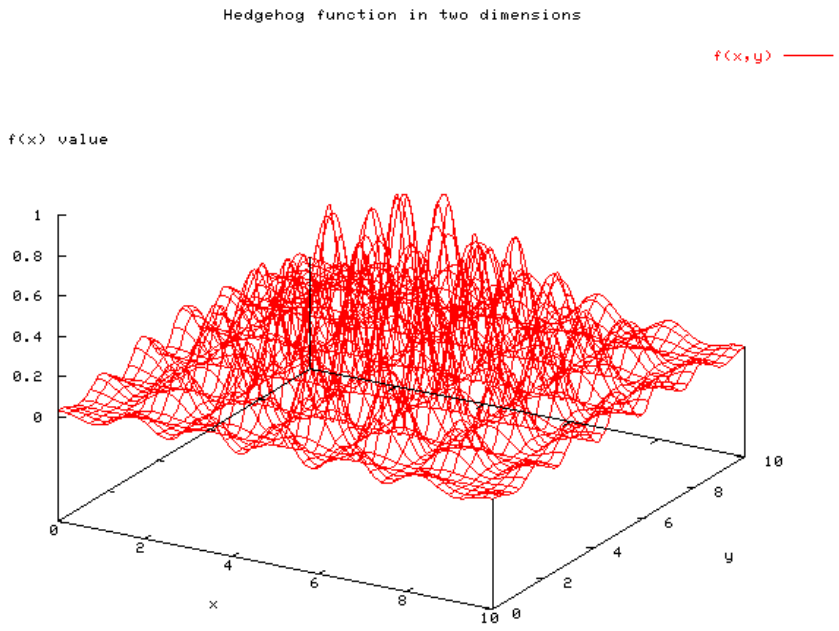

Figure 7.21: Hedgehog function in two dimensions with $\delta = 5$, $\alpha = 6$, and $s = 0.07$

The number φ of sub-optima generated by this function can be calculated as shown in equation 7.6 in the case where $\alpha \in \mathbb{N}$.

$$\varphi = \begin{cases} (\alpha+1)^n - 1 & \text{if } \alpha/2 \in \mathbb{N} \\ \alpha^n - 1 & \text{otherwise} \end{cases} \tag{7.6}$$

Dimensionality has a large effect on the optimisation of problems with sub optima, the difficulty of the problem increasing dramatically as the number of dimensions increases due to both the decrease of the ratio $\mathcal{V}/\mathcal{S}$ and the increase of the number of local optima φ.

The following test consisted of setting the optimiser to optimise the Hedgehog function in different dimensions to test the scaling capacity of the optimiser

Number of Dimensions n	3	5	10	15	20
Number of Local Optima	26	242	$5.9*10^4$	$1.4*10^7$	$3.4*10^9$
Number of Evaluations (Avg)	220.60	622.90	1909.75	3765.80	6091.40

Table 7.5: Scaling of the optimisation technique depending of the dimensionality of the problem

in the presence of local optima. The same procedure was used as in section 7.3.3, the parameters for the Hedgehog function were set as follow, $\delta = 5$, $\alpha = 2$, and $s = 0.07$. Table 7.5 present the scaling of the average number of evaluations required to achieve the valid range and the number of local optima depending of the number of dimensions.

In figure 7.22 it can be observed that the increase in the number of evaluations with regard to the number of dimensions dose not increase linearly anymore although it still has a sub quadratic increase. It can be approximated for dimensions > 2, as $N = a * n^2 + b * n + c$ where $a = 10.247$ $b = 109.509$ and $c = -194.195$.

7.5 Constrained Problems

This section is concerned with the test to verify the capacity of the optimiser to optimise problems with constraints. The problem chosen to highlight the constraints handling capabilities of the optimiser is a constrained function based on a welded beam design problem, the physical description of the problem is outside the scope of this section. It is a relatively hard problem but has been well published [142, 29, 30]. It consists in optimising a set of four design variables $\overrightarrow{x} = (h, l, t, b)$subjected to five inequalities constraints.

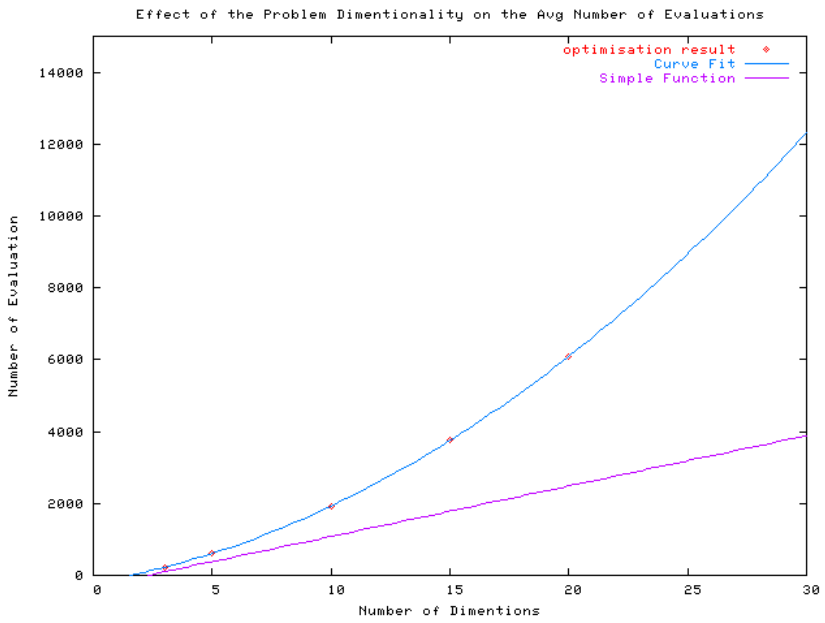

Figure 7.22: Dimensionality effect on oh the required number of evaluations for the Hedgehog function and comparison with the simple function

$$
\begin{array}{lcll}
f_w\left(\overrightarrow{x}\right) & = & 1.10471h^2l + 0.04811tb(14.0+l) & \textit{To be minimized} \\
 & & & \textit{subjected to} \\
g_1\left(\overrightarrow{x}\right) & \equiv & 13,600 - \tau\left(\overrightarrow{x}\right) \geq 0 & \\
g_2\left(\overrightarrow{x}\right) & \equiv & 30,000 - \sigma\left(\overrightarrow{x}\right) \geq 0 & 0.125 \leq h \leq 10 \\
g_3\left(\overrightarrow{x}\right) & \equiv & b - h \geq 0 & 0.1 \leq l,\, t,\, b \leq 10 \\
g_4\left(\overrightarrow{x}\right) & \equiv & P_c\left(\overrightarrow{x}\right) - 6,000 \geq 0 & \\
g_5\left(\overrightarrow{x}\right) & \equiv & 0.25 - \delta\left(\overrightarrow{x}\right) \geq 0 &
\end{array}
\tag{7.7}
$$

Where the terms $\tau\left(\overrightarrow{x}\right)$, $\sigma\left(\overrightarrow{x}\right)$, $P_c\left(\overrightarrow{x}\right)$, and $\delta\left(\overrightarrow{x}\right)$ are as follow:

$$
\tau\left(\overrightarrow{x}\right) = \sqrt{\left(\tau'\left(\overrightarrow{x}\right)\right)^2 + \left(\tau''\left(\overrightarrow{x}\right)\right)^2 + \frac{l\tau'\left(\overrightarrow{x}\right)\tau''\left(\overrightarrow{x}\right)}{\sqrt{0.25\left(l^2 + (h+t)^2\right)}}} \tag{7.8}
$$

$$
\sigma\left(\overrightarrow{x}\right) = \frac{504,000}{t^2 b} \tag{7.9}
$$

$$
P_c\left(\overrightarrow{x}\right) = 64,746.022\left(1 - 0.0282346t\right)tb^3 \tag{7.10}
$$

$$
\delta\left(\overrightarrow{x}\right) = \frac{2.1952}{t^3 b} \tag{7.11}
$$

And, where

$$
\tau'\left(\overrightarrow{x}\right) = \frac{6,000}{\sqrt{2}hl} \tag{7.12}
$$

$$
\tau''\left(\overrightarrow{x}\right) = \frac{6,000\left(14 + 0.5l\right)\sqrt{0.25\left(l^2 + (h+t)^2\right)}}{2\left\{0.707hl\left(\frac{l^2}{12} + 0.25\left(h+t\right)^2\right)\right\}} \tag{7.13}
$$

This results in a difficult to optimise problem with a knife shaped optimal region as shown in figure 7.23 which represent a plot of the valid space in the dimensions h and t.

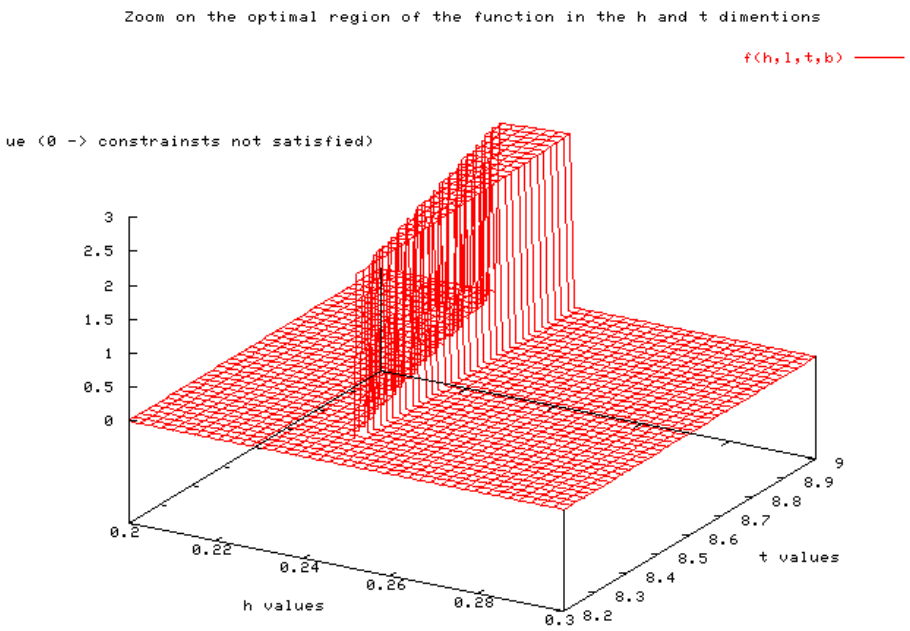

Figure 7.23: Optimal zone of $f_w(\overrightarrow{x})$in the dimensions h and t

The optimal solutions reported in the literature [142] is, $f_w = 2.38116$ with $h = 2.444$, $l = 6.2187$, $t = 8.2915$, and $b = 0.2444$. Using binary GA Deb [29] managed to achieve $f_w = 2.43$ and in a recent work [30] $f_w = 2.38119$. These results were obtained using between 40,000 and 320,000 evaluation of the function.

The test of the constraint technique consisted of optimising equation 7.7 using only 5,000 evaluations since for the application of this software 40,000 to 320,000 evaluations would be totally unrealistic . Out of 10 simulations the best optimisation achieved a value of 2.3906 which is within 99.6% of the best known solution, 5 optimisation were within 1%, 20 optimisation were within 10% and only 2 optimisation resulted in values below 75% of the best known solution. This result clearly shows the capacity of performing quality

optimisation with a relatively small number of evaluations.

7.6 Conclusion

During this chapter the behavior of the optimiser has been observed and described, in order to give a better insight for setting up and tuning the optimisation library.

The capacity of the optimiser to optimise different types of problems has been demonstrated by successfully applying the optimisation technique, to a simple problem with only a single optimum, to a problem composed of a number of local optima, and finally to a difficult constrained problem. The optimiser exhibited good performances in terms of the average number of evaluation required to perform the optimisation. The variation of the dimensionality of the problem exhibited a linear scaling of the evaluation requirement for the simple problem and a sub quadratic scaling for the Hedgehog function test problem containing a number of local optima.

This strong performance in term of the number of evaluations required to perform an optimisation is crucial for the application of the optimiser to real engineering problems where the number of optimisations will be limited due to the cost of the simulation.

Chapter 8

Combustor Design Optimisation

8.1 Introduction

The aim of this chapter is to present the application of the optimisation Toolbox to the preliminary design of a gas turbine combustor. For this purpose, the Toolbox was interfaced with external simulation tools used to define the quality of the design. The optimisation Toolbox was tested for different design conditions in order to experiment with different design approaches. Some possible ways of exploiting the problem knowledge generated during the optimisation will be explored. Finally, the application of the optimisation toolbox to other design problems will be presented.

8.2 The Design Problem

The combustor design method has been described in chapter two, there a set of parameters that can be used to define the quality of a combustor have been presented. These parameters will form the basis of the performance and constraints definition of the combustor design.

To allow the calculation of these performance parameters, the simulation tool Flownet presented in chapter four has been linked to the optimiser and its output will provide the necessary information to calculate the values of most of the performance parameters. In order to determine the level of pollutant emissions of the design, the Toolbox was also linked to a method for pollutant emissions prediction.

8.2.1 Selection of the Design Variables

There are a large number of variables that can be used to optimise a combustor design. These can be classified into three main sets of variables:

- Variables controlling the geometry of the combustor, such as size, shape, volume, number and position of the different features
- Variables controlling the features of the combustor, such as the cooling method, type of injector...
- Variables controlling the setting of the different features of the combustor, size and number of holes of the different ports on the flame tube.

The optimisation technique would be capable of optimising all these three sets of parameters. However the available simulation software limits the optimisation possibility for the two first parameter sets due to two following factors. Firstly, the simulation tool uses a non parametric description of the combustor geometry which is cumbersome and can not be modified easily. In addition, there are concerns about the capacity of the semi empirical tools to give reliable simulation results for the non conventional designs that may be generated by letting the geometry or the features of the combustor vary freely. The first limiting factor can be overcome by the use of a preprocessor that uses a parametric method to describe the combustor (such as the graphical

interface designed for Flownet). In order to overcome the second limiting factor flownet would need to be complemented by others tools capable of describing the mixing in a more refined and physical way.

The optimisation will limit itself with the optimisation of the third type of design variables, the settings of the different features of the combustor. The automation of the optimisation of those settings can be largely beneficial for the design project since a large part of the design time is spent on the tuning of these feature settings, resulting in a potential reduction of the design time and a fine tuning of the designs.

The different features of a combustor consist mainly in different ways to inject cold air in the combustion zone to control the combustion process and to cool the flametube walls. The setting of those features consists mainly in the definition of the number and the size of the openings for the different features, cooling patches, dilution ports. The ratio of the number of holes and the diameter of the hole define the penetration of the jet generated by this hole. The simple mixing relation used in Flownet dose not take into account the variation of the jet penetration. Under these conditions it is preferable to fix either the diameter or the number of holes and and use the other one as a design variable to control the massflow delivered by these ports.

The design variables for the combustor design consists in the diameter of the holes for the following features of the designs:

- The Injector(s)
- The different cooling features of the baseplate
- The different cooling features of the liner
- The different dilution orifices

8.2.2 Definition of the Objectives

The definition of the optimisation objectives is constrained by the available simulation capability. As it has been highlighted in chapter two some of the critical performance parameter of the combustor can not be directly simulated. This require the use of a number of other parameters as constraints in order to control them. The optimisations performed for the test cases will use the following performances parameters as objectives and constraints:

- The overall combustor pressure drop
- The inner and outer flame tube pressure drop
- The AFR for different zones of the combustor
- The Mass Flow of cooling air for the base plate(s) and the flametube region(s)
- The maximum wall temperature for different flametube walls region(s)
- The average wall temperature for different flametube walls region(s)
- The amount of recirculating flow in the primary zone(s)
- The overall S. I. Loading Λ_{SI}
- The Relight Loading χ_{SI}
- The Pollutant emissions
- The cross-flow Mach ratios

This set of objectives are used to define the quality of a given combustor design. One might notice that these performances parameters only are not sufficient to perform a complete design of a combustor. However the optimisation of these parameters allows the design process to be done faster, giving

more time for the designer to tackle other design problems like combustion instabilities or exit temperature profile. This limitation could be partially alleviated by the addition of simulation tools capable of tackling these missing performance parameters such as, a relight model, a model defining the stability of the combustor, a proper mixing model or the use of simple CFD to have information on the quality of th mixing that would allow to have a temperature traverse model, better emissions model.

This set of objectives can be used in two ways, these can be either used as a set of design targets that the optimiser have to reach within a constrained range, or as the optimisation of one or more objective while the other performance parameters are constrained to a specified range.

8.3 Test Cases

8.3.1 Achievement of a set of design targets

The aim of this design exercise is to demonstrate the capacity of the optimizer to generate a combustor design having the same performance parameter as one designed using the traditional methods. In addition this will allow to compare the automatically generated design with the original one.

8.3.1.1 Setting of the optimisation

For this optimisation the optimiser was configured to achieve as precisely as possible a set of target defined as the performances of an existing combustor designed using traditional techniques. The specific combustor is a single annular military combustor that will be referred as generic-combustor-01 (figure 8.1).

An acceptability criteria was introduced in order to define if a proposed design can be accepted. This acceptability criteria, impose arbitrarly that all the

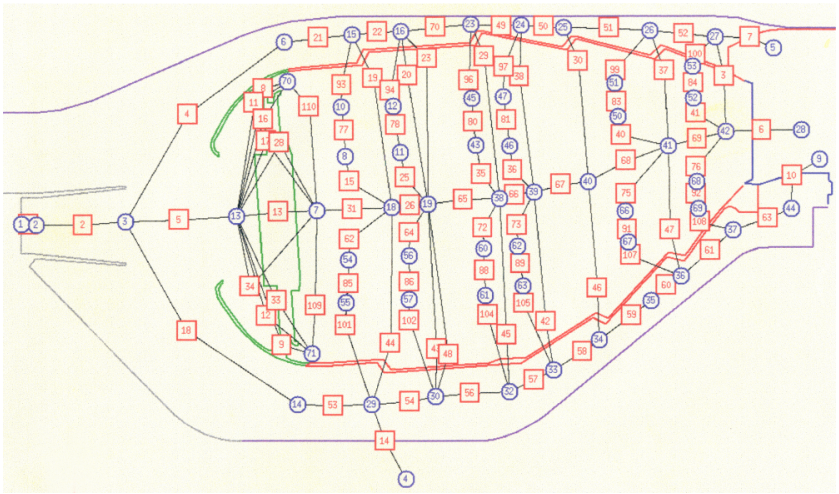

Figure 8.1: A Network Model of the Generic-Combustor-01

performance parameters of a proposed design must be within ± 5% of the desired target value. Using the method of "range error, target achievement and optimisation factor" described in section 6.4.2. The fitness is defined as a function of the distance between the actual design performance parameter value and the acceptance criteria when the later is not satisfied. In the other case, when the criteria is satisfied, the fitness is defined as a function of the distance towards the desired performance target.

This represents a rather hard design problem since the allowable range for each of the 23 design constraint is narrow and therefore limits the freedom of the optimiser.

8.3.1.2 The Optimisation Process

The design optimisation was performed on a Linux workstation using a 650MHz Athlon Processor and 512Mb of RAM. The optimisation process consisting of 10000 evaluations was completed in less than four hours (Figure 8.2).

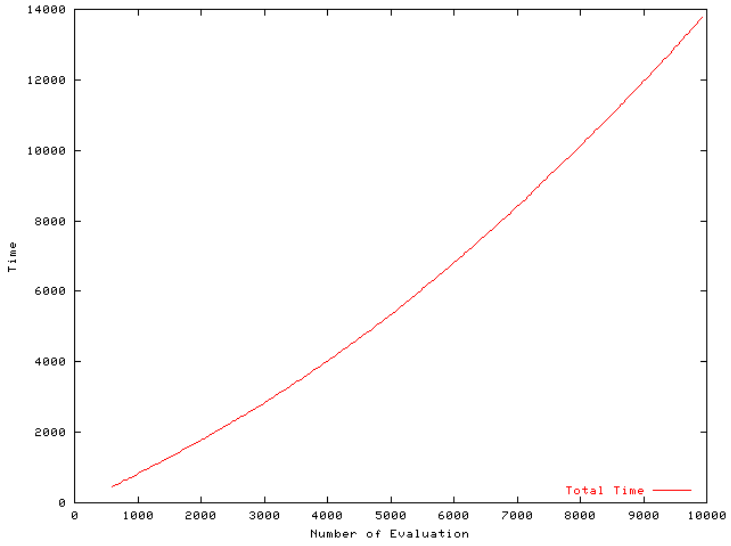

Figure 8.2: Evolution Of the required time in function of the number of evaluations

Figure 8.3 represent the evolution of the fitness as a function of the number of evaluations. The three fitness curves present on the graph represents respectively the best, the average, and the worst fitness present in the population as a function of the number of evaluations.

On the first section of this graph the fitness increase relatively linearly however since the value of the fitness is small at this point it dose not appear on this graph. This section represents the first phase of the optimisation process which could be described as strongly explorative.

On a second section a strong discontinuity can be observed, it represents the achievement of the acceptance criteria. This discontinuity in the fitness function would cause critical difficulties for analytical techniques and the sudden large increase in fitness might even bring traditional genetic algorithms to premature convergence. This is due to the sudden generation of elements with a significantly better fitness than the rest of the population generating an excessive selection pressure which might bring the GA to premature con-

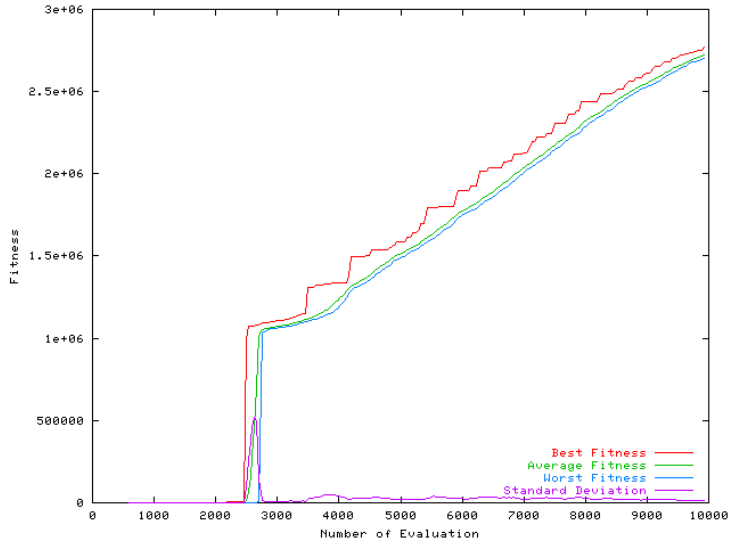

Figure 8.3: Evolution of the fitness as a function of the number of evaluations

vergence. This phenomenon has been strongly reduced thanks to the use of fitness scaling and the combination of SUS sampling and duplicate prevention techniques.

On the last section of the graph, it can be seen that the fitness continue increase linearly until the end of the graph, where the difference between the best and the worst fitness decrease demonstrating that the optimisation is converging. This convergence phase is the final phase of the the optimisation process and consists mostly of exploitation search.

8.3.1.3 The Designed Combustor

Figures 8.4 and 8.5 shows the evolution of two key design parameters the Pressure drop and the zone crossflow Mach ratio as they are reaching their respective target these represent the value of the best element of the population, it can be observed that the parameters tend to oscillate but are are

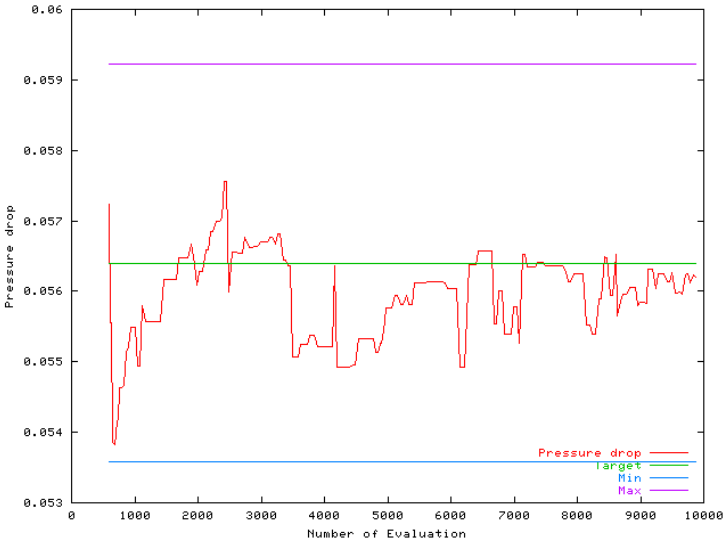

Figure 8.4: Evolution of the pressure drop as a function of the number of evaluations

well directed towards the target value . Table 8.1 shows the final performance parameters achieved by the optimisation. From this table it can be observed that the Toolbox allowed the precise achievement of the targets, the maximum target error being 3.33%.

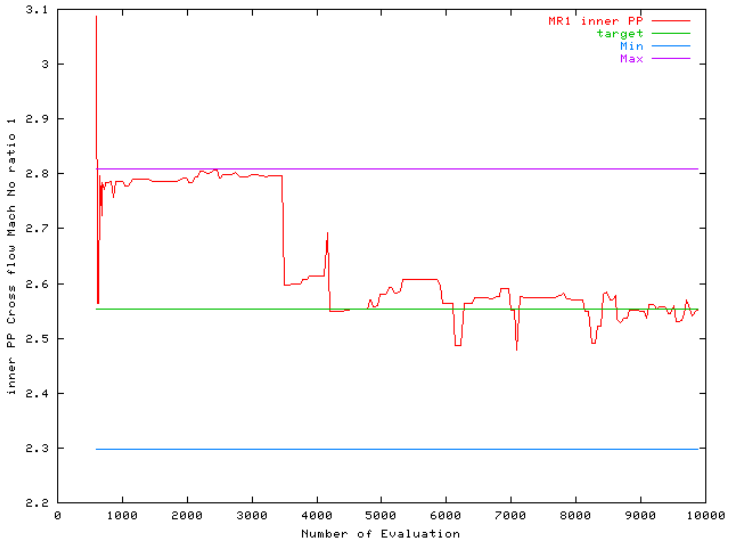

Figure 8.5: Evolution of Mach ratio as a function of the number of evaluations

Target	Desired Range	Achieved Range
Combustor Pressure Drop	±5%	0.27%
Pressure drop wall 1	±5%	0.43%
Pressure drop wall 2	±5%	0.41%
AFR injector	±5%	-2.82%
AFR 1	±5%	0.56%
AFR 2	±5%	0.90%
Flametube cooling	±5%	-1.90%
Base Plate Cool	±5%	1.42%
Max Temp Combustor	±5%	-0.02%
Max Temp Zone 1	±5%	-3.00%
Max Temp Zone 2	±5%	-0.02%
Avg Temp Combustor	±5%	0.15%
Avg Temp Zone 1	±5%	-0.54%
Avg Temp Zone 2	±5%	0.75%
Recirculating flow	±5%	0.31%
SI Loading	±5%	0.03%
Relight loading factor	±5%	0.29%
NOx	±5%	-1.21%
Mach ratio Outer Ports 1	±5%	-1.89%
Mach ratio Inner Ports 1	212 ±5%	0.73%
Mach ratio Outer Ports 2	±5%	-0.93%
Mach ratio Inner Ports 2	±5%	3.33%

Table 8.1: Achievement of a set of 22 design parameters targets

8.3.2 Minimization of the Wall Cooling Flow

The previous section allowed to demonstrate the capacity of the optimiser to generate a design satisfying the same performance criteria as an existing combustor. The aim of this new design task is to demonstrate the capacity of the optimiser to go beyond the performance targets achieved by existing combustors and further push the objectives.

To demonstrate this capacity it was decided to use the optimiser to minimise the air flow required for the cooling of the flametube walls while constraining the others performance parameters to be within the design criteria of the original project. The reduction of cooling flow requirement is a very interesting exercise since it permits to reduce the pollutant formation zone generated by the cold cooling gases, and it as well allows to benefit of more air to be used to control flow properties such as AFR and combustor exit temperature profile.

8.3.2.1 Setting of the Optimisation

For this task the optimiser used the same performance parameters from generic-combustor-01 as in the previous section. However this time the value of the flame tube cooling flow is set as an optimisation parameter and its value will be minimised. In addition The target constraints and range were defined based on the original project targets which, provide a larger range for the constraints values. The range on wall temperatures was reduced to zero the wall temperature should never exceed the maximum prescribed temperature.

8.3.2.2 The Optimisation Process

The optimisation was performed in 4h 30min using a single workstation for 10000 evaluation, all the targets were within the allowable range after 2500

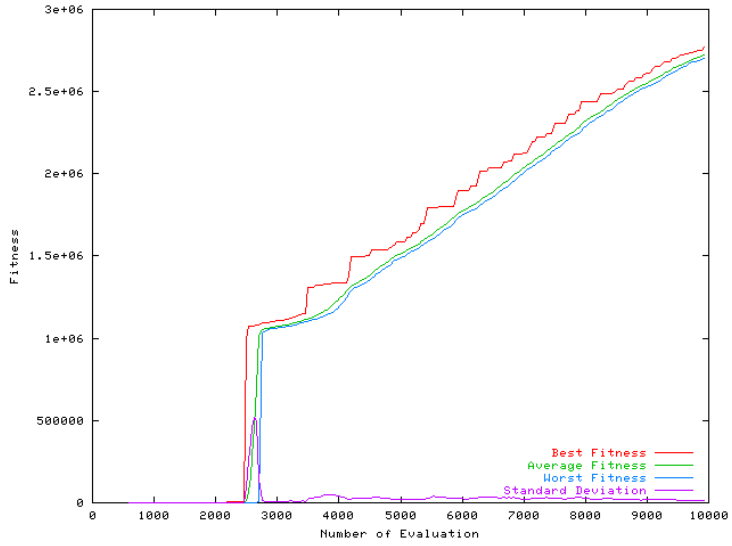

Figure 8.6: Evolution of the fitness as a function of the number of evaluations

evaluations. As in the previous section the optimisation process can be observed in figure 8.6 and shows the same phases.

8.3.2.3 The Designed Combustor

Figure 8.7 shows the evolution of the total cooling flow which is to be minimised, it can bee seen on this figure that no optimisation is performed on the cooling flow until all the targets are within their allowable range. It need to be noted that the cooling flow from the base plate which is to be firmly constrained since heat transfer is not modeled on the baseplate. From this graph a consequent reduction in the combustor cooling flow can be clearly observed. Table 8.2 shows the final performance parameters achieved during this optimisation. From this table it can be observed that the Toolbox allowed a 23.28% reduction of the flametube cooling flow.

It is now interesting to examine the differences between the original and the

Target	Desired Range	Achieved Range
Combustor Pressure Drop	±5%	-0.49%
Pressure drop wall 1	±5%	-0.80%
Pressure drop wall 2	±5%	-0.75%
AFR injector	±5%	3.69%
AFR 1	±5%	0.67%
AFR 2	±5%	2.00%
Flametube cooling	NA	-23.28%
Base Plate Cool	±5%	-4.97
Max Temp Combustor	+0%	-0.70%
Max Temp Zone 1	+0%	-0.89%
Max Temp Zone 2	+0%	-0.70%
Avg Temp Combustor	+5%	1.22%
Avg Temp Zone 1	+5%	-0.73%
Avg Temp Zone 2	+5%	2.34%
Recirculating flow	±5%	1.84%
SI Loading	±5%	-0.05%
Relight loading factor	±5%	1.82%
NOx	+0%	-5.97%
Mach ratio Outer Ports 1	±10%	-5.36%
Mach ratio Inner Ports 1	±10%	-0.17%
Mach ratio Outer Ports 2	±10%	1.17%
Mach ratio Inner Ports 2	±10%	-2.02%

Table 8.2: Achievement of a set of 22 design parameters targets

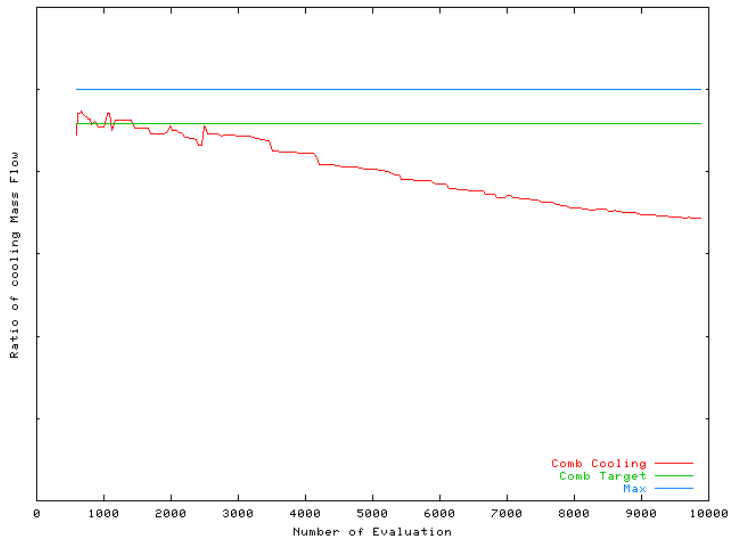

Figure 8.7: Evolution of the flame tube cooling flow as a function of the number of evaluations

optimised design. Figure 8.8 and 8.9 represents the amount of cooling flow injected in the flametube through the cooling orifices for both the original and the optimised design. Figure 8.10 and 8.11 shows the flametube wall temperature as well for both the original and the optimised design. From this set of graph it can be observed that the optimiser manage to reduce the cooling flow requirements by un-constraining the wall temperature and preventing unnecessary overcooling of certain parts of the combustor. It is as well interesting to note that the original design has a section of its outer wall over the maximum temperature limit, and that this was corrected in the optimised section thanks to a small increase of the fourth Z-ring of the outer wall. A careful observer might note the fact that the fifth Z-ring of the outer flametube wall is positioned towards the exit of the combustor and do not seems to have a strong participation in the cooling of the flametube, this was detected by the optimiser which severely reduced its massflow. However this Z-ring might be used to improve the exit temperature profile and its reduction

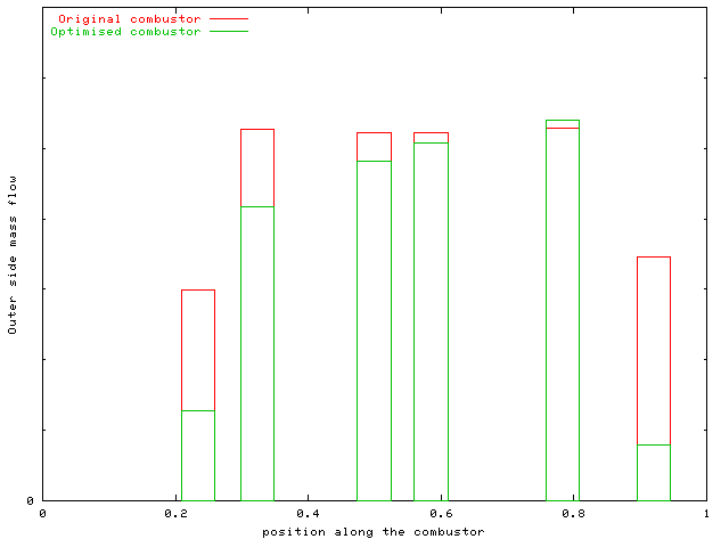

Figure 8.8: Evolution of the cooling flow entering the flametube outer wall along the combustor

might adversely affect this parameter therefore for further optimisation this value should be constrained. It is reassuring to observe that the results from the optimisation are logical and can be simply explained.

8.3.3 Minimisation of the NOx Emissions

The successful application of the Toolbox for the optimisation of the flame tube cooling flow gave confidence in the optimisation technique and the robustness of the simulation code, therefore it was decided to apply the optimisation Toolbox to the more delicate problem of the reduction of NOx.

8.3.3.1 Setting of the Optimisation

The optimisation toolbox was set similarly as for the previous experiments, using the same performance parameters from generic-combustor-01 as in the

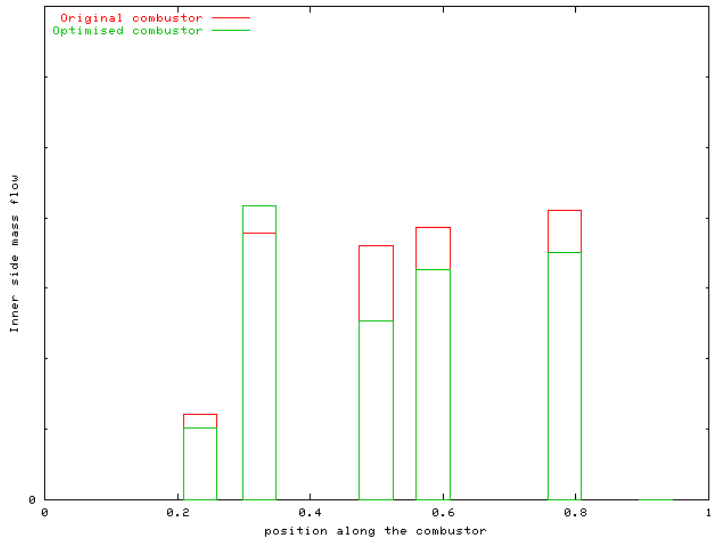

Figure 8.9: Evolution of the cooling flow entering the flametube inner wall along the combustor

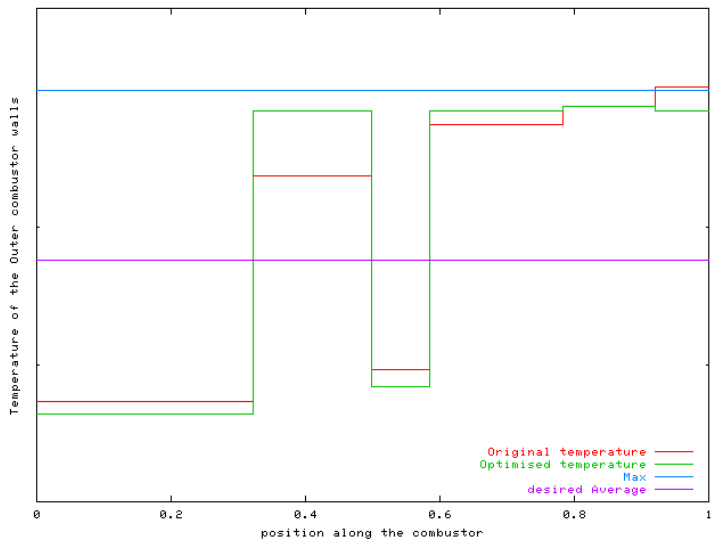

Figure 8.10: Evolution of the flametube outer wall temperature along the combustor

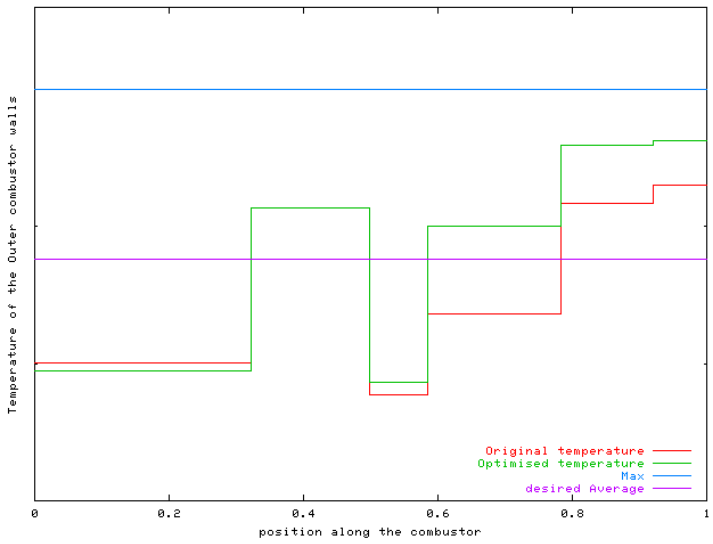

Figure 8.11: Evolution of the flametube Inner wall temperature along the combustor

previous section. However this time the value of the predicted NO emissions is set as an optimisation parameter and its value will be minimised. The target constraints and range were defined similarly to the previous section, but with removing the range constraint for the AFR values.

8.3.3.2 The Optimisation Process

The optimisation was performed in 36 minutes using 10 networked workstations for 10000 evaluation. All the targets were within the allowable range after 2000 evaluations . The optimiser managed to reduce the value of the predicted emission by 18.6%. As in the previous sections the optimisation process can be observed in figure 8.12 and shows the same phases.

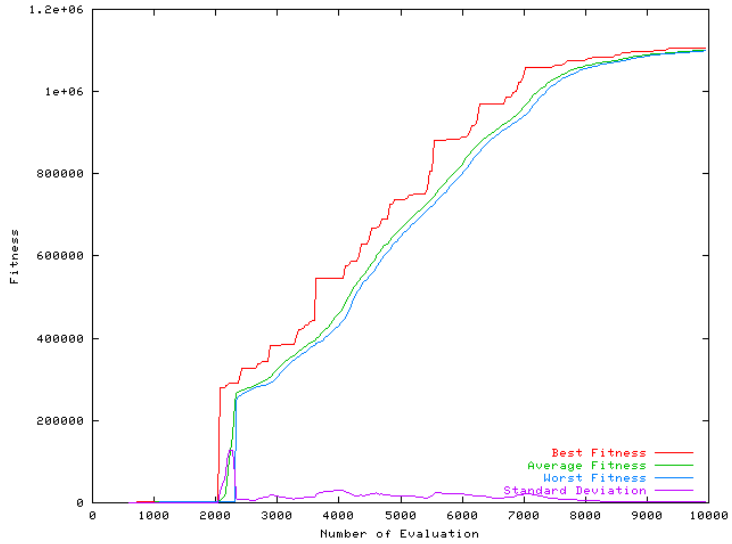

Figure 8.12: Evolution of the fitness as a function of the number of evaluations

8.3.3.3 The Designed Combustor

The results of the optimisation are shown in table 8.3. As before it can be seen that all the parameter were achieved within their allowable range of error.

Figure 8.13 shows the evolution of the NOx emissions along the optimisation, and Figure shows the relight-loading factor for the combustor. From those two graphs it is interesting to note that the reduction of NOx is stopped when the relight-loading factor hits the limit of its range. This shows that the relight loading, which is dependent on the combustor mass flow, seems to be a limiting factor for NOx Reduction. This behavior can be explained by the fact that NOx emission are reduced by a leaner and therefore colder primary zone, in order to make this zone leaner the optimiser increase the amount of mass flow in this zone, this has for effect to increase the amount of recirculating mass flow (equation 2.7), which in turns brings the relight

Target	Desired Range	Achieved Range
Combustor Pressure Drop	±5%	-0.39%
Pressure drop wall 1	±5%	-0.63%
Pressure drop wall 2	±5%	-0.60%
AFR injector	NA	12.80%
AFR 1	NA	4.82%
AFR 2	NA	4.16%
Flametube cooling	±10%	-1.67%
Base Plate Cool	±5%	4.69%
Max Temp Combustor	+0%	-0.10%
Max Temp Zone 1	+0%	-1.95%
Max Temp Zone 2	+0%	-0.10%
Avg Temp Combustor	+5%	1.08%
Avg Temp Zone 1	+5%	-1.21%
Avg Temp Zone 2	+5%	2.39%
Recirculating flow	±5%	4.97%
SI Loading	±5%	-0.04%
Relight loading factor	±5%	4.99%
NOx	+0%	-18.59%
Mach ratio Outer Ports 1	±10%	-1.06%
Mach ratio Inner Ports 1	±10%	8.10%
Mach ratio Outer Ports 2	±10%	3.91%
Mach ratio Inner Ports 2	±10%	7.21%

Table 8.3: Achievement of a set of 22 design parameters targets

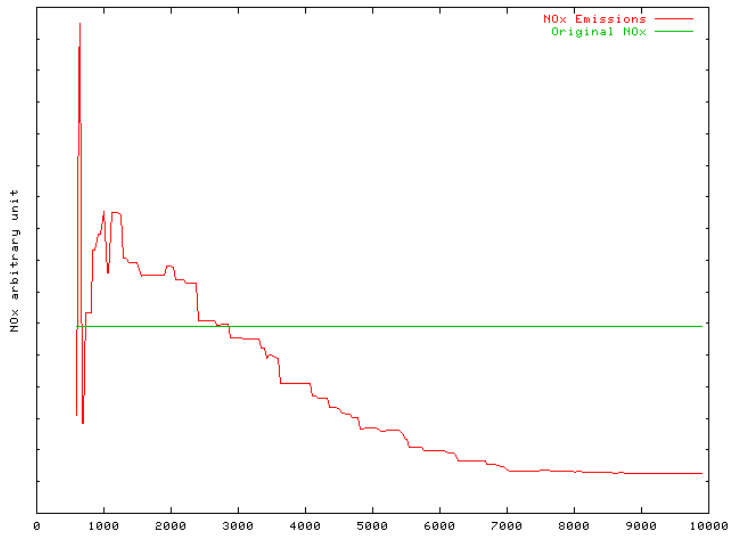

Figure 8.13: Evolution of the predicted NOx emissions as a function of the number of evaluations

loading factor (equation 2.12) towards it's upper limit. The satisfaction of the constraint having a prime importance over the optimisation, the relight loading is stabilised on the edge of it's upper limit.

Figure 8.15 and 8.16 shows the AFR along the flame-path for the manual design and the optimised design respectively as a comparison, and as a difference. From these figures it can be seen that the optimiser went for a slightly leaner primary zone.

In order to highlight this fact Figures 8.17, 8.18, and 8.19 shows the evolution of the AFR of respectively, the injectors, the zone 1 and 2 of the combustor. From those graphs it is interesting to note that the optimiser chooses to go as lean as the relight loading factor allows to reduce the average temperature and therefore the NOx emissions.

In this case as well the design generated by the optimiser seems logical and an explanation of the improvement can be provided.

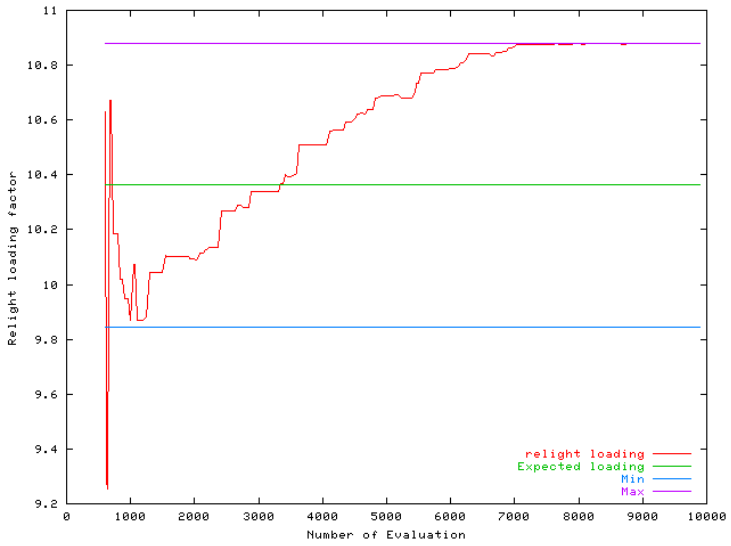

Figure 8.14: Evolution of the Relight Loading factor as a function of the number of evaluations

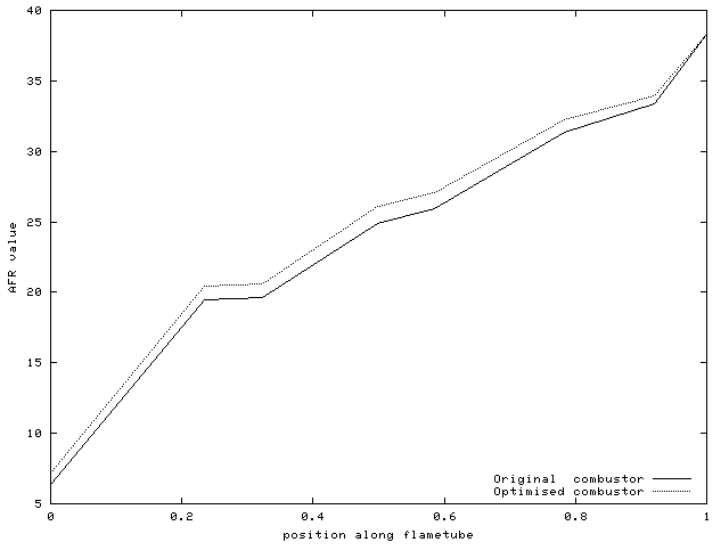

Figure 8.15: Evolution of the AFR along the flametube

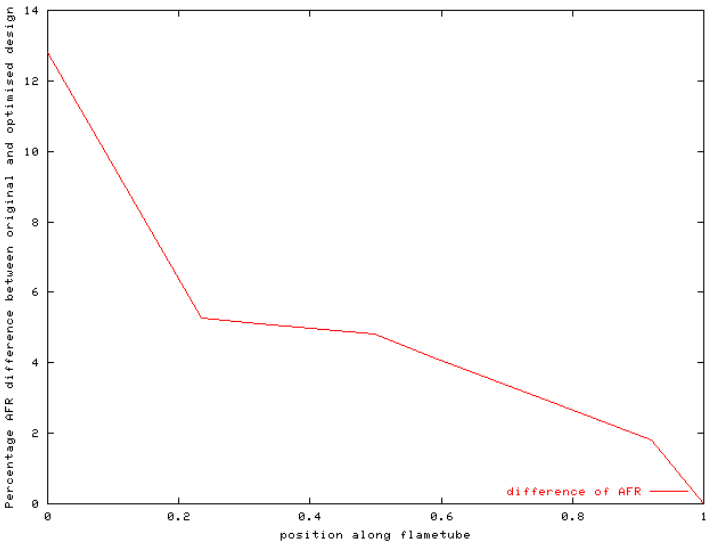

Figure 8.16: Evolution of the AFR difference along the flametube

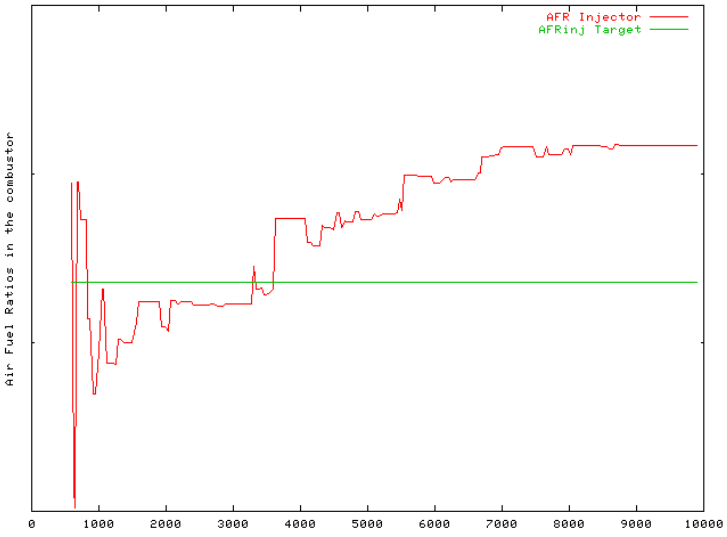

Figure 8.17: Evolution of the Injector AFR as a function of the number of the evaluations

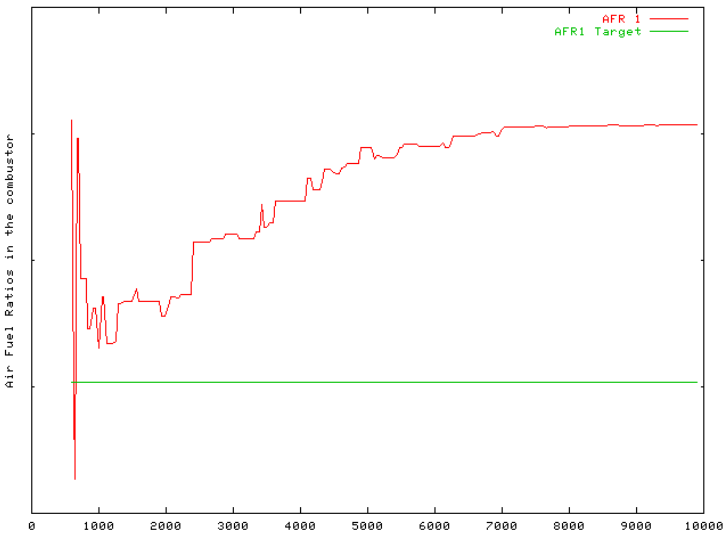

Figure 8.18: Evolution of the zone 1 AFR as a function of the number of evaluations.

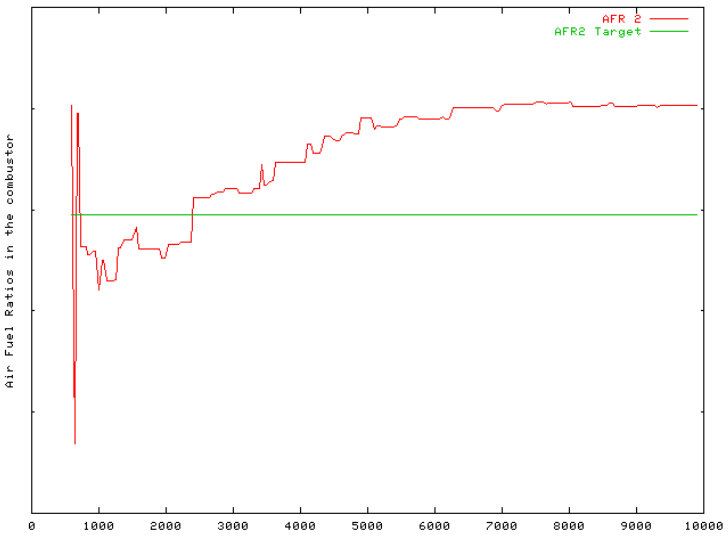

Figure 8.19: Evolution of the zone 2 AFR as a function of the number of evaluations.

8.4 Other applications

The robustness and flexibility of genetic algorithms have allowed to apply this Optimisation Toolbox other design problems. These are briefly described in order to demonstrate the Toolbox capability to be used in a wide range of engineering domains.

8.4.1 Biomass Gasifier

Alcides Codciera-Neto [20] used the Toolbox to optimise the performance of combined cycle power-plant using biomass fuel with a gasifier system. The optimisation was performed on the gasification process order to define the optimum air, steam, and fuel temperature, as well as the steam molar number and the equivalence ratio in order to maximise the fuel Low Calorific Value (LCV) while constraining the ratio of energy losses in the process. Even-though this work took place during the early stage of the development of the design optimisation Toolbox it provided very encouraging results by allowing optimum input parameters to be defined while satisfying the constraint.

8.4.2 Airfoil design

Dimitris Pantazis [125] used the tool box in conjunction with an airfoil simulation tool 'xfoil' to optimise the wing profile of an unmanned air vehicle. The optimisation has been performed by modifying the control point of a spline defining the wing profile. The simulation tool is based on the panel method and allowed to rapidly determine the Cl and Cd of the wing at different angles of attack. Two optimisation approach were tested, the maximisation of Cl over a range of angle of attack while minimizing drag at zero incidence., for the second one the the simulation constrained the achievement of a $Cl = 0.35$ at 0 angle of attack and the minimisation of drag.

This study gave very promising results by generating high performance airfoils that can be adapted to their specific mission.

8.5 Conclusion

This chapter presented the application of the optimisation Toolbox to real life combustor design problems. The use of the Toolbox allowed the relatively precise achievement of the design parameters of an existing combustor designed using traditional methods, demonstrating the capacity of automatic design for targets. The second and the third optimisation exercises allowed the demonstration of the capacity of the optimiser to achieve the performance targets and range desired for the design project while optimising a given performance parameter, leading to better combustor design eliminating some problems present on the existing design and allowing in one case a 22.3% reduction in the cooling flow requirements and in the other cases a 18% reduction in NOx emissions. In addition, the analysis of the design allowed the alleviation of some concerns about the capacity of the optimiser to produce sensible designs.

Finally, the application of the optimisation Toolbox to two others engineering design problems allowed the demonstration of the robustness of the method and its wide range of possible applications.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

This work concerned the development of a novel preliminary design methodology for gas turbine combustor. This method takes advantage of design optimisation techniques to partially automate the preliminary design process, allowing the generation of fully optimised combustor designs in a short time scale.

The initial task consisted of analyzing the combustor preliminary design process, its requirements, its features, and most importantly the identification of the key parameters which define the performance of a combustor these are crucial for any optimisation technique in order to define the quality of the proposed design.

This led to the development of a genetic algorithm based optimisation Toolbox which regroup a Genetic Algorithm based optimisation library and a set of tools to ease its integration within an engineering design environment.

A suitable simulation technique was selected to be interfaced with the Toolbox. Flownet, a semi empirical simulation tool based on network algorithms

was selected for its good compromise between computational requirements and physical representation. However the simulation capability had to be increased to model NOx emission. For this purpose a simple NOx model was implemented.

In order to provide background on Genetic Algorithms, their principles have been presented with some examples of their use in the engineering domain. This was followed by a detailed description of the simple genetic algorithm techniques, and the identification of the shortcomings of these techniques.

The background on genetic algorithms firmly in place, a Java based SGA library was developed A novel constraint handling technique was developed in order to handle efficiently the large number of objectives and constraints present in engineering design problems. In order to tackle the shortcomings of the SGA techniques, an extensive transformation of the SGA library were performed, by implementing the state of the art methods and operators, and developing some novel ones specifically designed for engineering application.

To check the effectiveness of this newly designed GA library, it had to be validated against analytical problems. The validation of the optimisation capability of the Toolbox was performed by applying the optimisation technique to three different types of analytical optimisation problem with known solutions: a simple problem with only a single optimum, a problem composed of a number of local optima, and a difficult constrained problem. These optimisation tests were performed for different number of input variables (dimensionality of the problem)

These test problems were first used to define the most appropriate settings for the optimisation method. The variation of the dimensionality of the problem exhibited a linear scaling of the evaluation requirement for the simple problem and a sub quadratic scaling for the Hedgehog function test problem containing a number of local optima. These results demonstrate the excellent performance of the optimisation method for high-dimensional problems, showing that it is capable of optimising problems with 20 to 30 dimensions

in less than 10 000 evaluations. This strong performance in terms of the number of evaluations required to perform an optimisation is crucial for the application of the optimiser to real engineering problems where the number of optimisations are limited due to the cost of the simulation.

Finally, the the optimisation Toolbox was applied to real life combustor design problems, in order to verify its capacity to partially automate the design process. The three optimisation exercises were performed on a single annular combustor: achievement of an existing design, minimisation of cooling flow under constraints, and minimisation of NOx emissions under constraints.

These optimisation task were successfully performed. The first one allowed the precise achievement of the 23 performance targets; the the second one allowed a 22.3% reduction of the flametube cooling flow; and the third optimisation provided an 18% reduction in NOx emissions for the combustor. For these applications the optimisation time was relatively short; it took an average of 50 minutes on a Linux workstation.

Has it as been hi-lighted by the attempt made to use empirical NOx correlations (section 4.4.1) the GA will use any flaw in the model in order to improve the performance of the final design. Care need to be taken to analyse the optimised design and make sure it is correct, and if it is not add some constraints to prevent the optimiser to use the breach in the model. It should be remembered that: *"The quality of an optimised design is only as good as the model used to asses it."*. The analysis of the design allowed the alleviation these concerns by demonstrating that the combustor designs generated by the optimiser were correct and logical.

These test cases allowed to validate the optimiser's capacity to provide a valuable contribution to the combustor preliminary design process by allowing to dramatically reduce the time required to tune a combustor design to a given set of performance parameters.

Finally, the application of the optimisation toolbox to two others engineering design problems allowed the demonstrate of the robustness of the method

and its wide range of possible applications.

9.2 Recommendations for future work

Although the design methodology presented in this work through the optimisation Toolbox represent a complete and fully functional design tool it is possible to identify three main directions for further work, as follow:

- Improvement of the simulation capabilities.
- Further development of the optimisation capability.
- Extension of the optimisation process.

The following sections will present these three possible ways of improving the usefulness of the Toolbox.

9.2.1 Simulation Capability Extension

It has been mentioned in the previous chapters that the information provided by Flownet are not sufficient to allow the complete preliminary design process to be optimised, in addition the quality of the simulation has a critical effect on the quality of the optimisation. Therefore a more complete and reliable model of the combustor would allow the removal of a number of constraints which currently reduces the optimisation capability, and increase the confidence in the optimised design.

The key design parameters that should modeled are as follow:

- Exhaust temperature profile. This is a critical constraint for the turbine operation. It could be modeled through the inclusion of a relatively crude non reacting CFD model capable of giving information on

trends in the mixing quality, allowing a better calculation of pollutant emissions formation, and to predict an exit temperature profile. Using screening and neural network evaluation (section 6.11) to reduce the number of calls to the CFD simulation, the optimiser will cope with simulations costing up to an hour of CPU time. The use of the power of modern computers coupled to a pre initialised grid that is deformed to match the evaluated design should allow to use a grid fine enough to capture the global trends in the distribution of mixture fraction.

- CO, UHC, and Soot emissions. The emission of pollutant is a critical parameter of modern combustor designs. These could be modeled through the implementation of a chemical kinetics scheme to the existing NOx emission model.

- Combustion instabilities. These are becoming an important design issue for modern "clean" combustors like LPP, which tends to suffers from instability or noise. The availability of simple models to account of such problems is still limited, but would form a valuable contributing to the description of the quality of a design.

- Prediction of Hot Spots. Combustor tend to suffers from of hot spots on the liner walls which can not be simulated by the 1D simulation techniques. It would require the integration of a full CFD model of the combustor.

- Structural analysis and weight. Weight is a critical parameter of any aeronautical equipment, a thermo structural analysis of the combustor design would provide valuable information on lifetime and weight of a specific design.

Some other key parameter are modeled using relatively crude methods and would benefit of some improvement of the model

- NOx model. The NOx model could benefit from the inclusion of mixing information and more detailed chemistry.

- Ignition and relight. The ignition and relight is modeled through an empirical correlation. The optimisation quality would benefit from a better modeling of this critical parameter.

- Mixing model. The actual mixing model is extremely coarse however mixing affects a large number of key parameters, such as, temperature profile, pollutant emissions, amount of recirculating flow, wall cooling, fuel evaporation and distribution. The mixing model should take account of the swirler effect and the liner pressure drop. Ultimately a computational flow-field could be used to define the mixing. The inclusion of a proper mixing model would allow to remove a large number of constraint that are imposed to try to control the mixing to a fixed level. These constraints are for example, the flame tube pressure drop, the crossflow Mach ratio, the amount of recirculating flow in the preliminary zone. All these could be removed giving more freedom to the optimiser, to refine the tuning of the combustor design, giving the opportunity for further improvements in the optimisation capability.

9.2.2 Optimisation Capability Development

In addition to the simulation capabilities, the optimisation performances and capabilities have the potential to be improved in order to reduce the number of design evaluations, therefore allowing faster optimisation or more complex simulation tools to be used.

- Neural Networks could be trained with each simulated points and used to provide an approximate evaluation of the designs. This technique has the potential to dramatically reduce the number of calls to the simulation tool.

- The inclusion of auto adaptive parameters for the optimisation technique would allow a simplification of the use of the optimisation tool while ensuring that all the parameters are optimally set.

- The inclusion of Multi-Objective techniques supporting preferences would be a valuable tool for the design engineer by allowing online interaction between the designer and the optimisation process.

9.2.3 Extension of the optimisation process.

This third way of improvement is probably the most novel and promising in terms of the potential gains for the industry. In section 3.3.1 it has been mentioned that the optimisation process is ultimately the maximisation of the profits that the designed product is capable of generating. Therefore this section will relate research directions that could be taken to bring real optimisation problems to this theoretical goal.

- Simulation of manufacturing cost and life cycle cost of the design. This is an obvious direction which has the potential of providing significant savings by allowing to optimise the performance / cost tradeoff.

- Simulation of manufacturability issues. These are responsible for extra manufacturing costs and quality problems and should be taken into account for a proper estimation of the design quality [157].

- Simulation of the customer behavior. This is the key element that would ultimately allow to relate a design with potential profits, however there is a lack of existing model in this domain.

- Optimisation of the operating envelope. A design should really be optimised over its complete operation envelope to guarantee optimum performance in all operating conditions instead of a single design point.

This route has been successfully attempted for the design of an UAV wing [125] and for the design optimisation of gas turbine design [126].

Bibliography

[1] Adkins R. C. & Gueroui D. (1986), An Improved Method For Accurate Prediction of Mass Flows Through Combustor Liner Holes. ASME Paper, ASME-86-GT-149.

[2] Anand M. S. & Priddin C. H. (2001) Combustion CFD - A Key Driver to Reducing Development Cost and Time, Fifteenth International Symposium on Air Breathing Engines. Bangalore, India. ISABE-2001-1087.

[3] Anderson M. B. (1995), The potential of Genetic Algorithms for Subsonic Wing Design. American Institute of aeronautics and Astronautics paper ref.: AIAA-95-3925.

[4] Andersson J. (2001), Multi Objective Optimisation in Design. PhD Thesis, Institute of Technology Linköping University, Sweden.

[5] Antonisse J. (1989), A New interpretation of Schema Notation that Overturns the Binary Encoding Constraint. proceedings of the Third International Conference on Genetic Algorithms. Cited in [77].

[6] Baker J. E. (1987). Reducing Bias And Inefficiency in the Selection Algorithm. Proceedings of the second international Conference on Genetic Algorithms. Hillsdale, NJ: Lawrence Erlbaum Associates. p 14-21.

[7] Ballal D. R. & Lefebvre A. H. (1973), Film Cooling Effectiveness in the Near Soot Region. Journal of Heat Transfer, pp265-266.

[8] Baukal C. E., Gershtein V. Y., and Li X. (2001), Computational fluid dynamics in industrial combustion,CRC Press, ISBN 0-8493-2000-3.

[9] Beck M. A. & Parmee L. C. (1998), conceptual Design Exploration Using Evolutionary and Adaptive Strategies. Proceedings of ACDM'98, PEDC, University of Plymouth.

[10] Becker T. & Perkavec M. A. (1994), The capability of different semi-analytical equations for estimation of NOx emissions of gas turbines. ASME Paper 94-GT-282, International Gas Turbine and Aero-engine Congress and Exposition, The Hague, Netherlands.

[11] Bethke A. D. (1981), Genetic Algorithms as Function Optimizers, Doctoral Dissertation, University of Michigan. Cited in: Goldberg D. E. [54].

[12] Borman G. L. & Ragland K. W., Combustion Engineering, McGraw-Hill, 1998, ISBN 0-07-006567-5

[13] Bos A. H. W. (1998), Aircraft Conceptual Design by Genetic / Gradient-Guided Optimisation. Engineering Application of Artificial Intelligence Vol. 11, pp377-382.

[14] Bouvier D. (2001), LOWNOX - A European Low Emission Engine Initiative, Air & Space Europe, Vol3, No1/2, pp96-100.

[15] Bowman C. T., Control of combustion-generated Nitrogen Oxide Emissions: Technology driven by regulation. Proceedings of the 24^{th} International Symposium on Combustion, The Combustion Institute, Pittsburgh, 1992.

[16] Bramelette M. F. & Bouchard E. E. (1991), Genetic Algorithms in Parametric Design of Aircraft. In Handbook of Genetic Algorithms [27] pp109-123.

[17] Bullock G. N. et al. (1995), Developments in the use of genetic algorithm in engineering design, Design studies Vol. 16, pp507-524.

[18] Cantu-Paz E. (2002), On random numbers and the performance of genetic algorithms, Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2002 San Francisco, Morgan Kaufmann.

[19] Chankong V. & Haimes Y. Y. (1983), Multi Objective Decision Making Theory and Methodology. North-Holland

[20] Codciera Neto A. (1999), Assessment of Novel Power Generation Systems for the Biomass Industry. PhD Thesis, Cranfield University.

[21] Coello Coello C. A. (1999), A survey of Constraint Handling Techniques used with Evolutionary Algorithms. Technical Report Lania-RI-99-04, Laboratorio Nacional de Informatica Avanzada, Xalapa, Veracruz, Mexico.

[22] Coverstone-Caroll V., Hartmann J. W., Mason W. J. (2000), Optimal Multi-Objective Low-Thrust Spacecraft Trajectories. Computer Methods in Applied Mechanics and Engineering, V186, pp387-402.

[23] Croome J. E. (1995), The application of a Genetic Algorithm Optimisation Technique to Axial Compressor Design. MSc Thesis, School Of Mechanical Engineering, Cranfield University, UK.

[24] Crossley W. A. (1996), Genetic Algorithm Approaches for Multiobjective Design of Rotor Systems. American Institute of aeronautics and Astronautics paper ref: AIAA-96-4025-CP.

[25] Curnock B. (1993), Engine Starting And Stopping, Von Karman Institute Lecture Series on Gas Turbine Engine Transient Behavior. document Ref VKI/LS-1993-06.

[26] Davis L. (1987), Genetic Algorithms And simulated annealing. Pitman, London 1987.

[27] Davis L. (1991), Handbook of Genetic Algorithms, International Thomson Computer Press, ISBN 1-85032-825-0.

[28] Deb K. (1990), Optimal Design of a class of Welded Structures via genetic Algorithms. 31th Conference on Structural Dynamics and Material, pp444-453.

[29] Deb K. (1991), Optimal Design of a Welded Beam Structure Via Genetic Algorithms, AIAA Journal, Vol29, pp2013-2015.

[30] Deb K. (2000), An Efficient Constraint Handling Method for Genetic Algorithms, Computer Methods in Applied Mechanics and Engineering. pp311-338.

[31] Deb K., Anand A. and Joshi D. (2002), A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimisation. KanGAL report Number 2002003, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur.

[32] Deb K. & Agrawal R. (1995), Simulated Binary Crossover for Continuous Search Space. Complex Systems, 9, pp115-148.

[33] Deb K. & Beyer H. G. (1999), Self Adaptive Genetic Algorithms with Simulated Binary Crossover. University of Dortmund Technical Report Number CI-61/99.

[34] Deb K. & Goyal M.(1997), Optimizing Engineering Designs Using a Combined Genetic Search. in Back T. (Ed.), Proceedings of the Seventh International Conference on Genetic Algorithms. Morgan Kaufman, San Francisco. pp521-528.

[35] Dejong K. (1975), An analysis of the behavior of a class of a class of genetic adaptive systems. PhD Thesis, University of Michigan.

[36] Despierre A., Stuttaford P. J., and Rubini P. A. (1997), Preliminary gas turbine combustor design using a genetic algorithm. ASME paper 97-GT-72.

[37] Doorly D. J. & Peiro J. (1997), Supervised Parallel Genetic Algorithms in Aerodynamic Optimisation. American Institute of aeronautics and Astronautics paper ref: AIAA-97-1852.

[38] Eshelman L. J. and Schaffer J. D. (1993), Real coded genetic algorithms and interval schemata. in Foundation of Genetic Algorithms II, Whitley D. (ed). pp 187-202. Cited in [34].

[39] Esping B. (1995), Design Optimisation as an engineering tool. Structural Optimisation Vol. 10, Springer-Verlag, pp. 137-152.

[40] Esquivel S. et All (2002), Enhanced Evolutionary Algorithms for Single and Multiobjective Optimisation in the Job Shop Scheduling Problem. Knowledge-Based Systems, V15, pp13-25.

[41] Fenimore C. P., Formation of Nitric Oxide in Premixed Hydrocarbon Flames. Proceedings of the 13^{th} International Symposium on Combustion, The Combustion Institute, Pittsburgh, 1970.

[42] Fisher P. A. (1930), The Genetical Theory of Natural Selection. Oxford Charendon Press.

[43] Fletcher R. S. & Heywood J. B. (1971), A Model For Nitric Oxide Emissions From Aircraft Gas Turbine Engines. AIAA paper, AIAA-71-123.

[44] Fogarty T. C. (1997), Genetic Algorithms for the Optimization of Combustion in Multiple-Burner Furnances and Boiler Plants. In Handbook of Evolunary Computation, Oxford University Press, Section G3.2.

[45] Fogel D. B. (1988), An evolutionary approach to the traveling salesman problem. Biological Cybernetics, 60, pp138-144.

[46] Fogel D. B. (2000), Evolutionary Computation: Towards a New Philosophy of Machine Intelligence, Second Edition, IEEE Press, ISBN 0-7803-5379-X.

[47] Fonseca C. M. & Fleming P. J. (1993), Genetic Algorithms for Multiobjective Optimisation: Formulation, Discussion and Generalization. Fifth International Conference on Genetic Algorithms, Morgan Kaufman.

[48] Frankenberger E. & Blake-Schaub P. (1998), Modeling design process in industry - Empirical investigations of design work in practice. Automation In Construction, Vol 7, pp139-155.

[49] Morley C. GasEq V0.71, http://www.c.morley.ukgateway.net/gseqmain.htm, 2001.

[50] Ginnakoglou K. C. (2000), Acceleration of genetic Algorithms Using Artificial Neural Networks - Theoretical Background, Von Karman Institute for fluid Dynamics Lectures Series 2000-07.

[51] Giannakoglou K. C. & Giotis A. P. (2000), Acceleration of Genetic Algorithms Using Artificial Neural Networks Application of the Method. Von Karman Institute for Fluid Dynamics Lectures Series 2000-07.

[52] Goldberg D. E. (1980), Adaptive control of gas pipeline systems. University of Michigan, Ann Arbor.

[53] Goldberg D. E. (1987), Simple Genetic Algorithms and the Minimal Deceptive Problem. Genetic Algorithms and Simulated Annealing, L. Davis (Ed.), pp 74-78.

[54] Goldberg D. E. (1989), Genetic Algorithms, In search, Optimisation & Machine Learning. Addison-Wesley, ISBN 0-2011-5767-5.

[55] Goldberg D. E. (1990). Real Coded Genetic algorithms, Virtual Alphabets, and Blocking. ILLIGAL Report No. 90003, University of Illinois .

[56] Goldberg D. E. Deb K. Thierens D. (1993), Toward a Better understanding of mixing in genetic algorithms. Society of Instrument and Control Engineers Journal Vol 32, pp10-16.

[57] Greenhough V. W. & Lefebvre A. H. (1957), Some Application of Combustion Theory to Gas Turbine Development, Sixth International Symposium on Combustion, pp858-869. Cited in [95].

[58] Grefenstette J. J. (1983), Incorporating Problem specific Knowledge into Genetic Algorithms. Genetic Algorithms and Simulated Annealing: An Overview, Davis L (Ed), Hyperion Books, pp-42-60. ISBN 0-2730-8771-1

[59] Grefenstette J. J. (1986). Optimisation of Control Parameters for Genetic Algorithms, IEEE Transactions on Systems, Man, and Cybernetics, Vol 16, pp122-128.

[60] Grefenstette J. J. (1993), Deception Considered Harmful, Foundation of Algorithms, 2, L. Darrell Whitley (Ed.) pp75-91.

[61] Grefenstette J. J. & Baker J. (1989), How Genetic Algorithms Work: a Critical-look at Implicit Parallelism. Third Conference on Genetic Algorithm.

[62] Greyvenstein G. P. & Laurie D. P. (1994), A Segreated CFD Approach to Pipe Network Analysis. International Journal for Numerical Methods in Engineering.

[63] Gupta A. K. (1997), Gas Turbine Combustion: Prospects and Challenges, Energy Conversion Management Vol38, No10-13, pp1311-1318.

[64] Harik G. et all, (1997), The Gamblers Ruin Problem, Genetic Algorithms and the Sizing of Populations. IEEE Conference. on Evolutionary Computation, pp7-12.

[65] Hartley S. J. (1998), Concurrent Programming The Java Programming Language, Oxford University Press USA.

[66] Higuchi T, Tsutsui S and Yamamura M. (2000), Theoretical Analysis of Simplex Crossover for real Coded Genetic Algorithms. Parallel Problem Solving From Nature VI, pp365-374.

[67] Heywood J. B., Internal Combustion Engine Fundamentals,McGraw-Hill, 1988, ISBN 0-07-100499-8

[68] Hollstein R. B. (1971), Artificial Genetic Adaptation in Computer Control Systems. PhD Thesis University Of Michigan, USA. cited in [124]

[69] Holland J. H. (1975), Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, Reprint edition (April 29, 1992), MIT Press, ISBN 0-2625-8111-6.

[70] Homaifar A., Lai S. H. Y., Qi X. (1994), Constrained optimisation via Genetic Algorithms, Simulation, Vol62, No4, pp242-254. Cited in [109].

[71] Homaifar A., Lai S. H. Y., McCormick E. (1994), System Optimisation of Turbofan Engines Using Genetic Algorithms, Applied Mathematical Modelling, Vol18, pp72-83.

[72] Hunter A. (1995), SUGAL User Manual, University of Sunderland, UK

[73] Iacopino G. (2002), Low Emissions Gas Turbine. MSc Thesis, Cranfield University.

[74] ICAO (1999), ICAO Adopts New Aircraft Engine Emissions and Noise Standards, Council of the International Civil Aviation Organization (ICAO), 1 March 1999 document ref: PIO 02/99.

[75] Ilwecicz et all (1996), Cost Optimization Software for Transport Aircraft Design Evaluation (COSTADE) Design Cost Methods. NASA report ref: NASA-96-cr4737.

[76] Jakob W., Gorges-Schleuter M., and Blume C. (1992), Application of Genetic Algorithms to Task Planning and Learning. Parallel Problem Solving from Nature PPSN2, pp291-310.

[77] Janikow C. Z. & Michalewicz Z. (1991), An Experimental Comparison of Binary and Floating Point Representation in Genetic Algorithms. fourth Conference on Genetic Algorithms.

[78] Jasuja A. K. (1982), Plain-Jet Airblast Atomization of Alternative Liquid Petroleum Fuels. ASME Paper, ASME-82-GT-32.

[79] Jaszkiewicz A. (2000), On The Computational Effectiveness of Multiple Objectives Metaheuristics. Fourth International Conference on Multi-Objective and Goal Programming, MOPGP-00.

[80] Joines J. A. & Houck C. R. (1994), On the use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimisation Problems with GA's, Proceedings of the first IEEE Conference on Evolutionary Computation, IEEE Press, pp579-584

[81] Johnson W. M. (1991), Superscalar Microprocessor Design, Prentice-Hall, Englewood Cliffs, NJ.

[82] Jones B. (1999), Aircraft Gas Turbine Emissions: Legislation and Control Technology, IMechE Seminar on Gas Turbine Pollutant Emissions: Technology advances and Update, Cranfield University Ref S697.

[83] Jones M., Curnock P., Bradbrook S. J., and Brich N. (2001) Evolutions in Aircraft Engine Design and Vision for the Future, Fifteenth International Symposium on Air Breathing Engines. Bangalore, India.

[84] Kirkpatrick S., Gelatt C. D., Vecchi M. P. (1983), Optimization by Simulated Annealing, Science 220 (4598), pp671-680.

[85] Kita H., Ono I. and Kobayashi S. (1999), Multi Parental extension of the unimodal normal distribution crossover for real coded genetic algorithms. proceedings of the Congress on Evolutionary Computation. pp1581-1587

[86] Knuth D. E. (1973), The art of computer programming, fundamental Algorithms, 2nd ed. , Vol1, Addison-Wesley.

[87] Knuth D. (1981), Seminumerical Algorithms,2nd Edition, The Art of Computer Programming, Addison Wesley, Vol 2, Section 3.2.1.

[88] Koza J. R. (1989), Hierarchical Genetic Algorithms operating on populations of computer programs. 11th Conference on artificial Intelligence, pp687–774.

[89] Kumar K. K. (1992), Genetic Algorithms: An Introduction and Overview of their Capabilities. American Institute of aeronautics and Astronautics paper ref: AIAA-92-4462-CP

[90] Lefebvre A. H. (1966), Theoretical Aspects of Gas Turbine Combustion Performance, CoA Note Aero No. 163, Cranfield University. Cited in [95].

[91] Lefebvre A. H. (1983), Gas turbine combustion first edition. Mc Graw Hill.

[92] Lefebvre A. H. (1984), Fuel effects on gas turbine combustion - Linear temperature, pattern factor and Pollutant Emissions. AIAA-84-1491, also Journal Of Aircraft Vol.21, No.11, 1984.

[93] Lefebvre A. H. (1985), Fuel Effects on Gas Turbine Combustion - Ignition, Stability, and Combustion Efficiency. Journal of Engineering for Gas Turbine and Power, Vol107, pp24-37.

[94] Lefebvre A. H. (1989), Atomization and Sprays. Hemisphere Publishing.

[95] Lefebvre A. H. (1999), Gas turbine combustion second edition. Taylor & Francis, ISBN 1-56032-673-5.

[96] Levine D. (1996), Users Guide to the PGAPack Parallel Genetic Algorithms Library. Argonne National Laboratory, USA, doc ANL-95/18.

[97] Louis S. G. (1993), Syntactic Analysis of Convergence in Genetic Algorithms, Foundations of genetic algorithms 2, Morgan Kaufman, San Mateo, CA, pp141–151

[98] Lowe D. (1998), Design Parameters, Private Communication.

[99] Lowe D. (1999), Altitude Relight Methodology, Private Communication.

[100] Michielssen E. Ranjithan S. and Mittra R. (1992), Optimal Multi Layer Filter Design Using Real Coded Genetic Algorithms. IEEE Proceedings journal Vol 139, No 6, pp413-420.

[101] Min S., Nishiwaki S., Kikuchi N. (2000), Unified Topology Design of Static and Vibrating Structures Using Multiobjective Optimisation. Computers & Structures, V75, pp93-116.

[102] Malecki R. E., Rhie C. M., Colket M. B., Mababushi R. K. (2001), Application of an Advanced CFD-Based Analysis System to The PW6000 Combustor to Optimize Exit Temperature Distribution - Part1: Description and Validation of the Analysis Tool. Proceedings of the ASME Turboexpo 2001, Paper 2001-GT-0062.

[103] Man K.F. Tang K. S. and Kwong S. (1999), Genetic Algorithms: Concepts and Designs, Springer-Verlag, ISBN 1-85233-072-4.

[104] Mandavilli S. & Patnaik L. M. (1997), On Exact populationnary Model of Genetic Algorithms, Information Sciences, pp37-67.

[105] Mason W. J. Optimal Earth Orbiting Satellite Constellations via Pareto Genetic Algorithm. AIAA-98-4381.

[106] Mari C. (2001), Trends in the Technological Development of AreoEngines: an Overview. Fifteenth International Symposium on Air Breathing Engines, Bengalore, India. ISABE-2001-1012.

[107] Meysenburg M. M. and Foster J. A. (1997), The quality of pseudorandom number generators and simple genetic algorithm performances. Proceedings of the Seventh international conference on genetic algorithms. Morgan Kaufman, pp. 521-528.

[108] Michalewicz Z. (1995), Genetic Algorithms, Numerical Optimization and Constraints, Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburgh, pp151-158.

[109] Michalewicz Z. (1996), Genetic Algorithms + Data Structure = Evolution Programs, Third edition, Springer-Verlag, ISBN 3-540-60676-9.

[110] Michalewicz Z. & Attia N. (1994), Evolutionary Optimization of Constrained Problems, Proceedings of the Third Conference on Evolutionary Programming.

[111] Michalewicz Z., Nazhiyath G. (1995), GENOCOP III: A Co-Evolutionary Algorithm for Numerical Optimization Problems With Nonlinear Constraints. International Conference on Evolutionary Computation, IEEE, Vol2, pp647-651.

[112] Michalewicz Z., & Schoenauer, M. (1996), Evolutionary Algorithms for Constrained Parameter Optimization Problems, Evolutionary Computation, Vol.4, No.1, pp.1-32.

[113] Mongia H. (2001), A synopsis of gas turbine combustor design methodology evolution of last 25 years. Fifteenth International Symposium on Air Breathing Engines, Bangalore, India. ISABE-2001-1086.

[114] Murthy J. N. Gas Turbine Combustor Modeling For Design. PhD Thesis, Cranfield University, UK.

[115] Murthy J. N. Gas Turbine Combustor Modeling For Design. PhD Thesis, Cranfield University, UK.

[116] Nadon L. J. J., Kramer S. C., and King P. I. (1998), Multi Objective Optimization of Mixed-Stream Turbofan Engines.

[117] Nicol D., Malte P. C., Lai J., Marinov N. N., and Pratt O. T. (1992), NOx Sensitivities for gas turbine engines Operated on lean Premixed Combustion and Conventional Diffusion Flame. ASME Paper, ref: 92-GT-115

[118] Nightingale P. (2000), The Product-Process-Organization Relationship in Complex Development Projects. Research Policy, Elsevier, Vol29, pp913-930.

[119] Nomura T. (1997), An Analysis on Linear Crossover for Real Number Chromosomes in an Infinite Population Size. ICEC 97 (IEEE), pp111-114.

[120] Obayashi S. (2000), Multi Objective Evolutionary Computation For Supersonic Wing Design. Von Karman Institute for fluid Dynamics Lectures Series 2000-07.

[121] Oyama A. Obayashi S. and Nakamura T. (2000), Real coded Adaptive Range Genetic Algorithm Applied to Transonic Wing Optimisation, 6th International conference on Parallel Problem Solving From Nature, Paris, Springer pp712-721. ISBN 3-540-41056-2.

[122] Pardalos P. M. et al (2000), Recent developments and trends in global optimisation, Journal of computational and applied Mathematics, Vol. 124, pp209-228.

[123] Parmee L. C. (1997), Evolutionary and adaptive strategies for engineering design. Seventh Conference on Genetic Algorithms. Morgan Kaufman, pp373-378.

[124] Parmee L. C. (2001), Evolutionary and Adaptive Computing in Engineering Design, Springer-Verlag.

[125] Pantazis D. (2002), Two Dimensional Airfoil Aerodynamics Optimisation with the Use Of Genetic Algorithms. MsC Thesis, Cranfield University, UK.

[126] Patnaik S. N., Guptill J. D., Hopkins D. A., and Lavelle T. M. (2001), Optimization for Aircraft Engines with Regression and Neiral-Network Analysis Approximators. Journal of Propulsion and Power, Vol17, No1, pp85-91.

[127] Periaux J. & Winter G. (1995), Genetic Algorithms In Engineering And Computer Science, John Wiley & Sons.

[128] Pierret S., Deumeulenaere A., Gouberneur B, and Hirch C. (2001), A Flexible and Automatic Design Environment Applied to the Optimisation of Turbomachinery Blades. Fifteenth International Symposium on Air Breathing Engines. Bangalore, India. ISABE-2001-1054.

[129] Powell D. & Skolnick M. M. (1993), Using Genetic Algorithms in Engineering and Design Optimisation with Non-Linear Constraints. Pro-

ceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufman, pp424-430.

[130] Pratchet T. Stewart I. and Cohen J. (2002), The Science of the Diskworld, second edition, Ebury Press, London. ISBN 0-09-188657-0.

[131] Press W. H. Teukolsky S. A. Wetterling W. T. Flannerey B. P. (1992), Numerical Recipes in C: The Art of Scientific Computing second Edition, Cambridge University Press, Chapter 7, pp274-316, ISBN-0-521-43108-5.

[132] Quagliarella D & Vinci A. (2000), GAs For Aerodynamic Shape Design II: Multi Objective Optimisation and Multi Criteria Design. Von Karman Institute for Fluid Dynamics Lectures Series 2000-07.

[133] Radcliffe A. (1960), Fuel Injection. in High Speed Aerodynamics and Jet Propulsion, Hawthorne W. R., Princeton University Press.

[134] Rogero J. M. (2002), Development of Emission Prediction Methods for whole Engine Performance and Engine Control Studies - EmisCalc V1.0, Cranfield University UTC Report, No.PE051.

[135] Roy G. D., Propulsion Combustion: Fuels to Emissions, Taylor & Francis, 1998, ISBN 1-56032-431-7

[136] Radcliffe N. J. (1991), Forma Analysis and Random Respectful Recombination, Proceedings of the Fourth International Conference on Genetic Algorithms.

[137] Radcliffe N. J. (1992), Non-Linear Genetic Representations, in Parallel Problem Solving from Nature II, Elsevier Science.

[138] Radcliffe N. J. & Surry P. D. (1995), Fundamental Limitations on Search Algorithms: Evolutionary Computing in Perspective, Lecture Notes in Computer Science, Vol. 1000, Springer Verlag, pp275-291.

[139] Radding C. (1982), Homologous Pairing and Strand Exchange in Genetic Recombination. Annual Review of Genetics, pp405-437.

[140] Rasheed K. Hirsh H. and Gelsey A. (1997), A genetic algorithm for continuous design space search. Artificial Intelligence in Engineering Vol 11, pp295-305.

[141] Rdolph G. (1994), Convergence Analysis of Canonical Genetic Algorithms. IEEE Transactions on Neural Networks, pp96-101.

[142] Reklaitis G. V., Ravindran A., and Ragsdell K. M. (1983), Engineering Optimization: Methods and Applications, Wiley.

[143] Rehman S. & Guenov M. D. (1998), A methodology for modeling manufacturing costs at conceptual design. Computers Ind. Engng, Elsevier, Vol35, No3-4, pp623-626.

[144] Reynolds D. & Gomatan J. (1996), Stochastic Modeling of Genetic Algorithms, Artificial Intelligence, pp303-330.

[145] Richardson J. T. et all (1989), Some Guidelines for Genetic Algorithms with Penalty Functions. Proceedings of the Third International Conference on Genetic Algorithms, pp191-197.

[146] Rizk N. K. & Mongia H. C. (1991), Gas Turbine Combustor Performance Evaluation, AIAA Paper, AIAA-91-0640.

[147] Rizk N. K. & Mongia H. (1992), NOx Model for lean Combustion Concept, AIAA paper AIAA-92-3314.

[148] Rizk N. K. & Mongia H. C. (1993), Semianalytical Correlations for NOx, CO and UHC Emissions, Journal of Engineering for Gas Turbine and Power, Vol115, No3, pp612-619.

[149] Rizk N. K. & Mongia H. (1994), Emissions Predictions of different Gas Turbine Combustors, AIAA paper AIAA-94-0118.

[150] Robins Sir. R. (1996), The Trent Program, a further step in engineering evolution. The 1996 Christopher Hilton Lecture at the Royal Institute Of Engineers, 1st October 1996. Cited in: [118]

[151] Rogero J. M. (1998), Study of a Mach 2 Cold Flow Spike Nozzle and use of Artificial Intelligence Techniques in the Aim of Shape Optimisation. Master Thesis, University of Strathclyde, Glasgow, UK.

[152] Rogero J. M. & Rubini P. A. (2001), Optimisation of Combustor Wall Heat Transfer and Pollutant Emissions for Preliminary Design Using Evolutionary Techniques. Fifteenth International Symposium on Air Breathing Engines, Bengalore, India. ISABE-2001-1122.

[153] Rogero J. M., Tiwari A., Muneaux O., Rubini P. A., Roy R., and Jared G. (2000) Application of Evolutionary Algorithms for Solving Real Life Design Optimisation Problems. Parallel Problem Solving From Nature (PPSNVI) Workshop. Paris.

[154] Roozenburg N. and Eekels J. (1995), Product Design: Fundamentals and Methods, John Wiley & Sons. ISBN 0-471-94351-7.

[155] Roy R. Parmee I. C. and Purchase G. (1996) Integrating the genetic Algorithms with the Preliminary Design of Gas Turbine Cooling Systems. second International Conference on Adaptive Computing in Engineering Design and Control, University of Plymouth.

[156] Roy R., Jared G., Tiwari A., and Muneaux O. (2000), Real Life Design Optimisation Features And Techniques. IEEE Proceedings of the Fifth Online World Conference on Soft Computing in Industrial Applications (WSC5) Finland.

[157] Roy R., Jared G., Tiwari A., and Muneaux O. (2000), Design Optimisation: A Survey of British industries. Flexo Report 5, SIMS, Cranfield University.

[158] Russell, S. & Norvig, P. (1995), Artificial Intelligence A Modern Approach. Prentice-Hall, ISBN 0-1310-380-52.

[159] Saenger W. (1984), Principle of Nucleic Acid Structure, Springer-Verlag, ISBN 0-3879-0761-0.

[160] Schipper Y. & Rietveld P. (1997), Economics and Environmental Effects of Airline Deregulation, Tinbergen Institute Discussion Papers No:97-031/3.

[161] Schmitt L. M. (2001), Fundamental Study: Theory of Genetic Algorithms, Theoretical Computer Science 259, pp1-61.

[162] Schmitt L. M. Nhaniv C. L. and Fujii R. H. (1998), Fundamental Study: Linear Analysis of Genetic Algorithms, Theoretical Computer Science 200, pp101-134.

[163] Schwefel H. P. (1977), Numerische optimerung von computer-modelen mittels der evolution strategie. Interdisziplinare Systemforschung, Basel and Stuttgart (Ed), Birkhauser Verlag, pp-5-8. Cited in [124].

[164] Schwefel H. P. (1981), Numerical Optimization of Computer Models, John Wiley & Sons, GB.

[165] Schoenauer M. & Xanthakis S. (1993), Constrained GA Optimisation, Proceedings of the Fifth International Conference on Genetic Algorithms.

[166] Simon H. (1969), The Science of the Artificial, MIT Press. Cited in [4]

[167] Sobieszczanski-Sobieski J. & Haftka R. T. (1997), Multidisciplinary aerospace design optimization: survey of recent developments, Structural Optimization, Springer-Verlag, Vol14, pp1-23.

[168] Stollery J. L. & El Ehwany A. A. M. (1965), A Note on the Use of a Boundary Layer Model for Correlating Film-Cooling Data. International Journal of Heat and Mass Transfer, Vol8 pp55-65.

[169] Stuttaford P. J.(1997), Preliminary gas turbine combustor design using a network approach. PhD Thesis School Of Mechanical Engineering Cranfield University, UK.

[170] Stuttaford P. J. & Rubini. P. A. (1996), Preliminary gas turbine combustor design using a network approach. ASME paper 96-GT-135.

[171] Sunderam V. S. & Geist G. A. (1999), Heterogenous parallel and distributed computing. Parallel Computing,, Elsevier Vol25, pp1699-1721.

[172] Svirezhev Y. M. & Passekov V. P. (1989), Fundamentals of Mathematical Evolutionary Genetics, Mathematics and its Applications, Vol 22, Kluwer Academic Publishers. London.

[173] Syed K, (2001), Private Communication, Alstom Power Industrial Gas Turbines.

[174] Thierens D. & Goldberg D. E. (1993), Mixing in Genetic Algorithms, Proceedings of the Fifth International Conference on Genetic Algorithms, pp38-45.

[175] Toffolo A. & Lazzaretto A. (2002) Evolutionary Algorithms for Multi-Objective Energetic and Economic Optimization in Thermal System Design. Energy, V27, pp549-567.

[176] Tokumasu Y., Miazawa K., and Fujitsuna Y. (2001), Overview of "Research and Development of Environmentally Compatible Propulsion System for Next-Generation Supersonic Transport (ESPR Project)", Fifteenth International Symposium on Air Breathing Engines, Bengalore, India. ISABE-2001-1179.

[177] Tonouchi J. H., Held T. J., and Mongia H. C. (1998), A Semi-Analytical Finite Rate Two Reactor Model forGas Turbine Combustors. Journal of Engineering For Gas Turbine and Power,Vol120, pp495-501.

[178] Truelove A. M. & Whitaker K. W. (1993), Rocket Stage Optimisation Using A simple Genetic Algorithms. 29th joint propulsion Conference, Monterey, AIAA-93-1778.

[179] Odgers J. & Carrier C. (1973): Modeling of Gas Turbine Combustors;Consideration of Combustion Efficiency and Stability, ASME paper, ASME-72-WA/GT-1

[180] Odgers J. & Kretschmer D. (1985), The Prediction of Thermal NOx in Gas Turbines, ASME paper, ASME-85-GT-126.

[181] Uelschen M. & Lawrenz M. (2000), Design of Axial Compressor Airfoils with Artificial Neural Networks and Genetic Algorithms. American Institute of aeronautics and Astronautics paper, Fluids 2000, ref: AIAA-2000-2546.

[182] Ulizar L. (2000), Customer oriented design. presented at Cranfield University.

[183] Vavak F. Jukes K. and Fogarty T. C. (1997). Adaptive Combustion Balancing in Multiple Burner Boiler Using a Genetic Algorithm with Variable Range of Local Search. Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan Kaufman, San Francisco. pp719-726.

[184] Vose M. D. & Rowe J. E. (2000), Random Heuristic Search: Applications to GAs and Functions of Unitation. Computer Methods in Applied Mechanics and Engineering, pp195-220.

[185] Wellens M. & Sing R. (2002), Propulsion System Optimisation for Minimal Global Warming Potential. Proceedings of the ICAS Congress, ICAS-2002-7112.

[186] Wellens M. & Sing R. (2003), Genetic algorithms Based Optimisation of Cooled Recuperated Turbofan Design. To be publisshed AIAA-2003-1210.

[187] Westerberg J. (2000), Air Transport System Sensibilities. Air & Space Europe, Vol 2, No 3, pp38-40.

[188] Whitley D. (1989), The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best, Proceedings of the Third International Conference on Genetic Algorithms.

[189] Wienke D., Lucasius C., and Kateman G. (1992), Multicriteria target vector optimisation of analytical Procedures Using a Genetic Algorithm. Acta Analytica Chimica, V265/2, pp211-225.

[190] Wright A. H. (1991), Genetic Algorithms for real parameter optimisation. in Foundation of Genetic Algorithms, Rawlins G. J. E. (ed). pp 205-222. Cited in [34].

[191] Zedda M. & Singh R. (1999), Gas Turbine Engine and Sensor Fault Diagnosis Using Optimisation Techniques. American Institute of aeronautics and Astronautics paper ref: AIAA-99-2530.

[192] Zeldovich Y. B., Sadovnikov P. Y., Frank-Kamenetskii, Oxidation of Nitrogen in Combustion, Academy of Science of USSR, Moscow,1947.

Appendix A

Documentation of the JGA Toolbox V1.0

JGA Optimisation Toolbox Documentation

Jean-Michel Rogero (j.m.rogero@free.fr)

22nd April 2003

Contents

Chapter 1

Introduction

1.1 Description

This is version 1.0 of the documentation for the JGA optimisation toolbox version 1.0.x. This documentation intends to describe the usage and the setting of the JGA optimisation toolbox in order to perform optimisation with a special emphasis for optimisation on gas turbine combustor.

1.2 Aim of this program

The JGA Optimiser code is an optimisation toolbox that has been design to perform optimisation of engineering problems, with a special emphasis on the gas turbine combustor preliminary design. It allows to simulate any problems as long as some way of evaluating the performances of the designs is provided. In addition some special features have been added for the use of this optimiser fro combustor preliminary design, It is able to run the network simulator Flownet as well as emissions and relight models.

The aim of this tool in it's use for preliminary design is to perform local refinements and tuning for a combustor design by modifying some design parameters. It is not intended as a complete design tool starting from a blank sheet and giving an optimised combustor as an output. It should be use to reduce the design time by performing the optimisation and tweaking that are normally performed manually.

The JGA optimiser is based on genetic algorithms or more generally evolutionary optimisation, some basic knowledge of these techniques are required to take full advantage of the optimisation features. The genetic algorithms

powering the optimisation code have been extensively been adapted to engineering design. More details about genetic algorithms can be found in the first and second year review of the project and in the thesis.

1.3 The optimisation toolbox design

The aim of the toolbox is to provide a set of methods and tools suitable for engineering design. Which will allow the designer to optimise numerous parameters of a complex problem using the designer's traditional simulation software to evaluate the solutions. To be useful the tool needs to be as efficient as possible and achieve results in a reasonable amount of time. It also needs to be flexible to permit improvements of existing methods and the addition of new tools. Finally to be used in real life it needs to be user friendly and allows the designer to interact with the optimisation process without requiring heavy training in the optimisation domain.

In order to achieve these objectives it will be needed to use a modular object oriented architecture that will permit tools to be added thereby enhancing the versatility and the performance. As well, to allow the engineer to use his traditional analysis software. An interfacing tool able to communicate with a wide range of third-party software will need to be developed. In order to be applied to engineering problems the optimisation tools will need to be very robust, and efficient on a wide range of problems.

Genetic algorithms are used for the main optimisation tool because of their robustness. But these need to be adapted to their use in engineering design to maximise the performance of the optimiser. As well, to reduce the computational overhead of simulation it was found necessary to design a tool distributing the evaluation of the analysis code over a network of heterogenous computers.

The last part of the tool box concerns the program / user interaction, the addition of a user friendly graphical interface where the designer can follow the evolution of all the problem parameters during the optimisation allows the user to get a 'feeling' of the optimisation process with out requiring a high knowledge of optimisation.

1.4 Toolbox architecture

A high degree of modularity was required in order to achieve the desired aims of versatility and expendability of the tool box. This was performed

by the use of Java as the main programming language, being object oriented it readily allows development of this type of modular architecture. As well it's platform independence avoids the burden of porting the code and eases the problems associated with working on a heterogenous set of computers. Another useful feature of Java is it's advanced support for networking and graphics. However this language suffers from being relatively slow compared to C/C++ and as well it is not memory efficient. The tool box is to be used for engineering design where simulation is performed by external analysis codes typically written in C / Fortran. Since the simulation process takes a very high proportion of the computational time the relative slowness of Java will not be a handicap as well the capacity of modern computer systems do not put heavy constraints on memory usage.

The object oriented design allows modules to be loaded at runtime depending on the toolbox requirements, it even permits new modules to be added without recompilation of the code. These modular tools are organised to support the interchangeable optimisation modules, by providing functionalities like interfacing with other software, distributing the evaluation over a network. The optimisation modules are themselves composed of interchangeable sub-modules. These implement the basic functions of the optimiser. Figure 3 shows an example of the toolbox modular organisation.

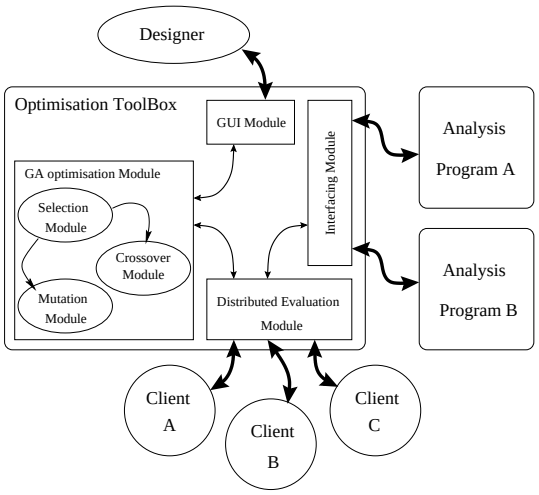

Figure 1.1: Example of the organisation of the modules.

Chapter 2

Installation

The installation is rather straight forward, the JGA Directory simply needs to be copied on the host machine, no further settings are required to run the code. In or der to compile it, the Makefile needs to be adapted to the environment.-

- Flownet should be compiled on each of the different system that compose the network (the make file will automatically create a different file extension. for each system)
- Update the version of Flownet compiled for your environment in the fl directory.
- Set up the runclient.sh script to suit your network (if distributed evaluation is required)
- The JGA Toolbox can be compiled by typing make in this directory.

Chapter 3

Usage & Example

The aim of this chapter is to describe the use of the optimisation toolbox through a worked example. For that we will take a combustor that will be named "GenericCombustor01".

3.1 The Generic Combustor 01 Example

3.1.1 Flownet Model

The first thing needed to perform an optimisation on a combustor is a Flownet model of this combustor. For the optimisation to be performed correctly the Flownet model should be complete and include wall heat transfer. No specific modifications are required on the Flownet model, however making sure that warnings are disabled increase the speed of execution.

3.1.2 JGA Configuration File

The second step consists in creating the configuration file for the optimisation tool box. This is the place where all the settings for the optimiser will be concentrated, the JGA toolbox parameters as well as the settings describing what to read and write in the Flownet input and output files.

No order is required for the setting of the different variables and data of the configuration file. however to allow easy understanding of this file it is important to stick to a way of describing the optimisation problem in this configuration file. That is why the setting of the file is separated in 3 section.

- The first one dedicated to set the optimisation toolbox for the problem.

- The second section is used to link the genes to features on the network input file.
- The third one is used to link the result of Flownet (the network file) to the parameters used for the calculation of the constraints.

3.1.2.1 Setting the optimiser

This is where the setting of the optimisation toolbox is performed. All the settings are described in the next chapter so this section will simply go through the input file and comment out on some settings.

```
// object JGA
variable chromosomeType         = GenericFlownetChrom
variable initialPopulationRatio = 1.5
variable populationSize         = 100
variable debugLevel             = 0
variable optimisationMethod     = GAOptimizer
variable randomSeed             = 151324
```

The chromosome to use to perform optimisation on a combustor is the generic Flownet chromosome. The debug level is normally set to zero. The random seed should be changed between two similar optimisation to use a different random sequence.

```
// Object JgaClient
variable serverName = osiris

// object DistributedEvaluation
variable JGAPath    = jga/
variable ServerPort = 5000
```

These setting are only needed when running the evaluation distributed.

```
// object Optimiser
variable maxNumberOfGenerations = 0
variable maxNumberOfEvaluations = 10000
variable maxFitness             = 1
variable finalPrintPopulation   = false
variable printPopInReport       = false
variable reportInterval         = 1
variable OutputFilesName        = OptimisationOutput
```

The zero value for the maximum fitness means that the optimiser will not stop depending of the fitness. It is not advised to print the population when it has a large size.

```
// object GAOptimiser
variable objPopInitialiser      = RandomInitialisation
variable objCrossoverSelector   = StochasticUniversalSampling
variable objMutationSelector    = RandomSelection
variable objReplacementOperator = TournamentReplacement
variable objCrossoverOperator   = BoundedSBXCrossover
variable objMutationOperator    = DynamicVectorMutate
variable objEvaluationOperator  = SerialEvaluation
// variable objEvaluationOperator = DistributedEvaluation
```

The settings can be toggled using the comment symbol //

```
// object chromosomeModifier/Selector
variable storeChromosomes = true
variable keepTrackOfId    = true
```

Use the store chromosome feature only when the evaluation take a relatively long time (more than a second).

Keeping track of ID allows to keep track of the ID of the parents of a chromosome

```
// Parameters fixed to 500 for genericflownet chromosome
// Constraints fixed to 49 for genericflownet chromosome
variable selectionPressure = 2
variable printEvaluationResults = false
variable chromosomeLength = 25
variable numberOfParameters = 500
variable numberOfConstraints = 49
// object sons of Chromosome
variable networkPath      = /tmp/fl/
variable originalFileName = network1b.net
```

Putting the Flownet code and network in the temporary directory increase the speed dramatically compared than running through the network file system.

The when using the generic Flownet chromosome the number of parameters should be set to 500 and the number of constraints should be set to 49.

```
    // object StochasticUniversalSampling
    variable SUSRate = 0.20
    // object RouletteSelection
    variable rouletteSelectionRate = 0.20
    //object BoundedSBXCrossover 0 = max exploration + -> decrease
explo
    variable distibutionIndex = 1.0
    // object RandomSelection
    variable randomSelectionRate = 0.05
    // object DynamicVectorMutate
    variable initialDistribFact = 0.775
    variable iterDependancyFactor = 1.6
    // object ReplacementOP
    variable elitism = true
```

Those are the value that works well on this case, elitism works generally well in all cases.

3.1.2.2 Setting The Constraints

When using the generic Flownet chromosome there is 31 constraints listed and described individually in chapter 4.17.8, these constraints might not all be useful to the particular case.

These constraints might be used for different purpose:

- converging a constraint towards a target value.
- constraining a constraint to a range.
- optimising a constraint by maximising or minimising it.

So these constraints can be switched on or off for each of these uses in the input file.

```
// Constraints               |---------- pressure drops ----------|--------------- AFRs ----------------...  -----|
//                             1     2     3     4     5     6    7     8     9      10    11    12    ...  31
data targetConstraintFlag   = { true  false false false false false true  true  true  false false false ...  false }
data rangeConstraintFlag    = { true  false false false false false true  true  true  false false false ...  false }
data optimiseConstraintFlag = { false false false false false false false false false false false false ...  false }
data maxOptimiseConstFlag   = { false false false false false false false false false false false false ...  false }
data constraintsTarget      = { 0.056 0     0     0     0     0     6.357 24.03 25.94 0     0     0     ...  0     }
data constraintsLowerRange  = { 0.950 0     0     0     0     0     0.950 0.950 0.950 0     0     0     ...  0     }
data constraintsUpperRange  = { 1.050 0     0     0     0     0     1.050 1.050 1.050 0     0     0     ...  0     )
.
```

The settings are only shown partially due to space problems.When the target constraint flag is set to true or when the optimise constraint flag is set to true the target for this constraint should be set. As well when the range flag is set at least one of the range for this constraint should be set.

3.1.2.3 Setting the genes to be written in the Flownet input file

The setting of the genes allows the toolbox to interact with Flownet through the ".net" input file. This is done through the "generic text file interface" which is able to process a text file and modify a number of key parameters.

This file interface search a file (normally the Flownet input file) for a given text string either in the whole file or in a designated column or in the designates line.

Please refer to section 4.18 for more detailed informations on the syntax.

For example we are looking to modify the diameter of outer primary dilution port (OPDP) labelled as element "20" in a Flownet input file. This element is located in the Cooling/Dilution Holes (HO) section of the input file, so a lock will be made on the "HO" string which defines the start of this section. We know that element numbers are located in the first column. As well we know that the diameter of the holes is specified in the 5th column

The configuration line link the gene 1 to the element 20 will be the following:

```
vect key {type name string line column linoffset coloffset geneNo}
vect i1  {lock      "HO"  0      1      0          0           }
vect i2  {gene PDP  "20"  0      1      0          4       1   }
```

The search is done sequentially in the file so the values searched have to be in the same order as in the file.

Here is a complete linking of genes to the input file:

```
// finding the cooling /dilution holes section.
vect i1  { lock        HO  0 1 0 0    }
// Baseplate Cooling
vect i2  { gene IBPC1 "34" 0 1 0 4 18 }
vect i3  { gene OBPC3 "11" 0 1 0 4 21 }
vect i4  { gene OBPC4 "8"  0 1 0 4 23 }
vect i5  { gene OBPC2 "16" 0 1 0 4 20 }
vect i6  { gene IBPC3 "12" 0 1 0 4 22 }
vect i7  { gene OBPC1 "17" 0 1 0 4 17 }
vect i8  { gene IBPC2 "33" 0 1 0 4 19 }
// Fuel Injector
vect i9  { gene FI    "13" 0 1 0 4 16 }
vect i10 { gene OBPC5 "28" 0 1 0 4 25 }
```

```
vect i11 { gene IBPC4 "9"  0 1 0 4 24 }
// Outer primary and secondary dilution port
vect i12 { gene OPDP  "20" 0 1 0 4  1 }
vect i13 { gene OIDP  "38" 0 1 0 4  3 }
// Inner Primary and secondary dilution port
vect i14 { gene IPDP  "43" 0 1 0 4  2 }
vect i15 { gene IIDP  "42" 0 1 0 4  4 }
// Inner and Outer primary effusion cooling
vect i16 { gene OPEC  "23" 0 1 0 4  5 }
vect i17 { gene IPEC  "48" 0 1 0 4  6 }
// Outer Zrings
vect i18 { gene OZR1  "19" 0 1 0 4  7 }
vect i19 { gene OZR2  "29" 0 1 0 4  9 }
vect i20 { gene OZR3  "30" 0 1 0 4 11 }
vect i21 { gene OZR4  "37" 0 1 0 4 13 }
vect i22 { gene OZR5  "3"  0 1 0 4 15 }
// Inner Zrings
vect i23 { gene IZR1  "44" 0 1 0 4  8 }
vect i24 { gene IZR2  "45" 0 1 0 4 10 }
vect i25 { gene IZR3  "46" 0 1 0 4 12 }
vect i26 { gene IZR4  "47" 0 1 0 4 14 }
```

3.1.2.4 Setting the parameters to be read in the flownet output file

The parameters are read in much the same way as the genes are written, There the parameter number replace the gene number and values are read and not written.

3.1.3 Performing the Optimisation

The optimisation toolbox can be run using two different modes. The text mode or the graphic mode

3.1.3.1 The text mode

To launch the optimisation using the text mode the command is “jga nameOfConfigurationFile”

The text mode is very basic, all the data about the optimisation followed the display of the progress of the optimisation

will scroll on the terminal. The optimiser will exit automatically when the stopping conditions will be reached. In order to kill the optimisation while it is ongoing use the “control C” combination.

3.1.3.2 The graphic mode

To launch the optimisation using the graphic mode the command is “jgat nameOfConfigurationFile”

The graphic mode allows to visualise the progress of the optimisation by displaying onto one graph the best, average and worst fitness,

and onto an other graph one of he followings, the value of the genes, the target or range error, the optimisation bonus, the value of each of the constraints or the value of each of the parameters.

Once the optimisation toolbox is launched the optimisation can be started using the start button and can be stopped using the stop button. The list in the middle allows th chose the data that will displayed on the left graph.

Chapter 4

Description of the Settings & Parameters

This chapter aims to describe in details each parameters and settings for all of the implemented modules. It is intended to be a quick reference when building a new input file, or developing an existing one.

For each objects (modules) their functionalities are described and their parameters are listed and explained in details with when possible the range of acceptable values and the usual settings.

4.1 The jga Object

This is the main object of the optimiser, it is where the optimisation technique is set and lunched and where the system is set up.

4.1.1 Debug Level "variable debugLevel (Integer)"

The debug level is used to setup the amount of information displayed in the console:

- 0 -> No debug information printed.
- 1 -> Minimal debug informations displayed.
- 2 -> Extended debug informations.
- 3 -> Very specific debug informations.

4.1.2 Random Seed "variable randomSeed (Long Integer)"

The setting of the random seed allow to reproduce optimisation:

- Two optimisation performed with the same Random Seed and exactly the same parameters will give the same results.
- Two optimisation performed with a different Random Seed and exactly the same parameters will give different results.
- Two optimisation performed with the same Random Seed and different parameters will give different results. Due to the fact that even a slight change in the parameter will affect the pseudo random number suite changing totally the suite.

The available range for the Random seed variable is a long integer from: 1 to 1E+19.

4.1.3 Statistics analysis "variable numberOfTrials (Integer)"

This variable is set when performing statistical analysis of the optimisation, it allows to run the optimiser a given number of times with a different random seed.

4.1.4 Store Chromosome "variable storeChromosome (Boolean)"

This setting allow to store all generated chromosomes in a database usable by the optimiser to improve the efficiency of the optimisation.

- "true" to store the chromosomes. Recommended when the evaluation of the chromosome is time consuming.
- "false" not to store the chromosomes. Recommended when evaluation is short or the number of evaluation is high.

4.1.5 Initial Population Size "variable initialPopulationRatio (Double)"

Defines the Initial size of the population as a ratio of the population size. The initial population being randomly generated an initial size of the population larger than the population size can improve the performances. As well the evaluation by flownet of combustors with random parameters results in flownet failing to give a result for a high proportion of the chromosomes (ie, combustor designs). Increasing the size of the initial population allows to compensate for the flownet failures.

4.1.6 Population Size "variable populationSize (Integer)"

The population size defines the number of chromosomes (ie, combustor designs) in the population at the end of each generation. There are no strict rules defining the size of the population for a problems. Usually it has been found that the size of the population should increase with the number of genes. a problem with 10 genes (variables) should have a population around 50 to 200 chromosomes and a problem with 20 genes should have a population between 150 and 500 chromosomes. These numbers will varies as well depending of the capacity of the GA operators to maintain diversity.

4.1.7 Optimisation Method "variable optinisationMethod (String)"

This is where the type of optimisation is selected by choosing the appropriate optimisation object. At the time of writing this documentation two optimisation objects exist:

- "RandOptimiser" The Random Optimisation object performs optimisation using the random search method. This method is very inefficient and therefore it is only intended to be used for comparison purpose.
- "GAOptimiser" The Genetic Algorithm Optimiser is the main Optimisation method, This methods is based on the genetic algorithms first developed by John Holland. However it has been extensively modified and the name of Evolutionary Strategies would be more appropriate. for more information on this object please refer to the 1^{st} and 2^{nd} year review and the thesis to be published. GAOptimiser is the preferred optimisation method for the time being.

- "HybridOptimizer" This is a test optimiser using GAOptimiser as the main optimiser and a hill climbing type technique towards the end.
- "OperatorTester" The operator tester is used to test new operators in known conditions.

New optimisation objects can be added by the users.

4.1.8 Chromosome Type "variable chromosomeType (string)"

This is where the type of chromosome is selected. The chromosome is the place where the design variables of the combustor to be optimised are stored as well as the methods for evaluating the quality of the designs. At the time being it is necessary to create a new chromosome object for each type of optimisation job. The objects available for the time being are:

- "FunctionChromosome" This a test chromosome where the optimisation consist in maximising one of three predefined functions of known optimum.
- "GenericFlownetChrom" This is the chromosome chromosome used to perform optimisation on any flownet combustor network.

This list might increase as new object needs to be created for new type of optimisations.

4.2 The jgaClient Object

This object is the client object that is run on remote computers when running with distributed evaluation. The setting of theses variables is only necessary when running distributed evaluations.

4.2.1 Server Name "variable serverName (string)"

This variable stores the name of the computer running the server of the optimisation.

4.2.2 Server Port "variable serverPort (Integer)"

This variable stores the port number where the server and the clients will be communicating.

4.3 The Optimiser Object

The Optimiser object is the parent object of all the optimisation methods. it is where the general settings used by all the optimisation methods are made.

4.3.1 Maximum Number of Generations "variable maxNumberOfGenerations (Integer)"

This variable allows to define the maximum number of generation to be processed. The optimiser will stop when the set number of generation have been performed. If the set number of generation is 0 the optimiser will not take into account the maximum number of generation.

4.3.2 Maximum Number of Evaluations "variable maxNumberOfEvaluations (Integer)"

This variable allows to define the maximum number of evaluations to be processed. The optimiser will stop when the set number of evaluation have been performed. If the set number of evaluation is 0 the optimiser will not take into account the maximum number of generation. Using the maximum number of evaluations as a stopping parameter is interesting because it allows to compare the performance of different settings, and as well it is proportional to the CPU time requirement.

4.3.3 Maximum Fitness "variable maxFitness (Double)"

This variable allows to define the maximum fitness of an optimisation to be performed. The optimiser will stop when the set fitness has been achieved. if the set fitness value is 0 the optimiser will not take into account the maximum fitness. This is useful on optimisations where the desired fitness is known.

4.3.4 Final Printing of the Population "variable finalPrintPopulation (Boolean)"

This setting allow the display in the console of the entire population at the end of the optimisation.

- "true" prints the final population.
- "false" do not print the population.

4.3.5 Print the Population In the Report "variable printPopInReport (Boolean)"

This setting allow the display in the console of the entire population during the report.

- "true" prints the population in the report.
- "false" do not print the population in the report. (the best member of the population will still be printed)

4.3.6 Reporting Interval "variable reportInterval (Integer)"

Define the interval between each reports in terms of number of generations.

4.3.7 Output Files Base Name "variable OutputFilesName (String)"

This defines the base name of your output files. For example If the base name is "combustorTest"

- The optimisation log will be "combustorTestOutput.out"
- The Population File will be "combustorTestPop.out"

4.4 The RandOptimiser and the GAOptimiser Objects

These objects are implementations of the optimisation methods.

The random optimisation object performs optimisation using the random search method. This method is very inefficient and therefore it is only intended to be used for comparison purpose.

The Genetic Algorithm Optimiser is the main Optimisation method, This methods is based on the genetic algorithms first developed by John Holland. However it has been extensively modified and the name of Evolutionary Strategies would be more appropriate. for more information on this object

please refer to the 1st and 2nd year review and the thesis to be published. GAOptimiser is the preferred optimisation method for the time being.

They share some common parameters settings that is why they are regrouped under a same section.

4.4.1 Evaluation Method Selection "variable objEvaluationOperator (String)"

This setting is common for the RandOptimiser and the GAOptimiser. The selection of the evaluation method allows tree settings:

- "SerialEvaluation" To evaluate all the chromosomes serially on a single computer
- "ParallelEvaluation" To evaluate the chromosomes in parallel on a multi CPU workstations when the operating system allows it.
- "DistributedEvaluation" To evaluate the chromosomes in parallel over a distributed network, the most efficient method for big optimisation jobs.

4.4.2 Population Initialiser Method "variable objPopInitialiser (String)"

This setting is common for the RandOptimiser and the GAOptimiser. It allows to select the method used to initialise the initial population. For the time being the only setting available is "RandomInitialisation" which randomly initialise the initial population within the genes upper and lower bounds.

4.4.3 Replacement Operator Selection "variable objReplacementOperator (String)"

This setting is common for the RandOptimiser and the GAOptimiser. It allows to select the method used at the end of a generation to replace some of the old members of the population by some of the newly generated chromosome.

- "TournamentReplacement" This method repeatedly create a tournament between some randomly picked chromosomes among the newly generated chromosomes and the old chromosomes, the chromosome with the best fitness will be kept and the other one will be discarded. This method might be a bit less efficient than the ranked replacement but allows to maintain diversity in the population,

- "RankedReplacement" This method creates a table including the new and the old chromosomes where the chromosomes are ordered depending of their fitness. The chromosomes with the best fitness are then integrated back in the population. This method is a bit more efficient than the Tournament Replacement but can suffer from premature convergence due to the reduction of diversity.

4.4.4 Crossover Selection Method "variable objCrossoverSelector (String)"

This setting is particular to the GAOptimiser. It allows to define the way the chromosomes are selected for the crossover operator. At the time of writing this documentation only two selection objects have been implemented:

- "RouletteSelection" This method consists in affecting to each chromosome an area proportional to their relative fitness on a virtual roulette wheel, this wheel is then spun repeatedly to select the chromosomes for the crossover. This method allows to give a chromosome a chance to be picked proportional to it's relative fitness.

- "StochasticUniversalSampling" This method is similar to the RouletteSelection method except that the wheel is spun only once and chromosomes are picked from pointers around the wheel. This method allows to reduce bias in the selection, this is the preferred selection method for crossover.

- "RandomSelection" This method consists in randomly picking chromosomes in the population for crossover. This method is not very suitable for crossover since it is not biased towards the fittest members.

4.4.5 Mutation Selection Method"variable objMutationSelector (String)"

This setting is particular to the GAOptimiser. It allows to define the way the chromosomes are selected for the mutation operator. At the time of writing

this documentation only two selection objects have been implemented:

- "RouletteSelection" This method consists in affecting to each chromosome an area proportional to their relative fitness on a virtual roulette wheel, this wheel is then spun repeatedly to select the chromosomes for the mutation. This method allows to give a chromosome a chance to be picked proportional to it's relative fitness. This method is not very suitable for mutation since it is biased towards the fittest members.
- "RandomSelection" This method consists in randomly picking chromosomes in the population for crossover. This is the preferred selection method for mutation.

4.4.6 Crossover Operator Object "variable objCrossoverOperator (String)"

This setting is particular to the GAOptimiser. It allows to define the operator used to perform the crossover. At the time of writing this documentation five crossover methods have been implemented:

- "OnePointCrossover" For this method a point of section of the chromosome is randomly selected then the first part of the first chromosome is paired with the second part of the second chromosome. This method is most suited for binary type chromosomes.
- "NPointCrossover" For this method a number N of section points of the chromosome are randomly selected then the parts of each chromosome are paired together. This method is most suited for binary type chromosomes.
- "UniformCrossover" For this method each gene is randomly selected from one or an other of the parents and included in the new chromosome. this method is suited for both binary or real type encoding.
- "WeightedAveragingCrossover" For this method a randomly weighted average between the two parents genes is performed. this is repeated for all the genes. This is the most efficient method for real number coded chromosomes like the chromosomes encoding the combustors.
- "WeightedLinearCrossover" sometimes referred as BLX-α, This is a method where a randomly weighted point is selected on a virtual line linking the two parents genes. The new point is selected on a range defined by the two parents genes and an exploration factor. this is repeated for all the genes.

- "BoundedSBXCrossover" This is the bounded Simulated Binary Crossover, which is based on the BLX-α crossover where a spread factor and a polynomial distribution function are used to perform the blending between the two parent genes values. This is the most efficient method for real number coded chromosomes like the chromosomes encoding the combustors.

4.4.7 Mutation Operator Object "objMutationOperator (String)"

This setting is particular to the GAOptimiser. It allows to define the operator used to perform the mutation. At the time of writing this documentation three mutation methods have been implemented:

- "RandomMutate" This operator randomly mutate a random gene over it's all range. It is the most basic form of nutation, it's explorative properties are quite good however it has been found to be disruptive for genes that have very sensitive settings like ports diameters and etc..

- "CreepMutate" This operator randomly mutate a random gene over a given portion of it's range. It's effect is close to hill climbing, it's explorative properties are limited in range than pure random mutation but it is much less disruptive.

- "CreepMutateWithDecay" This operator randomly mutate a random gene over a given portion of it's range with the portion of this range reducing with the number of evaluations. It's effect is close to hill climbing with annealing, it's explorative properties are less limited in range than the creep mutation due to a larger range at the beginning of the optimisation and it is much less disruptive due to the reduction of this range when the optimisation tend towards the end and some genes becomes critical.

- "DynamicVectorMutate" This operator use a vector of random direction and magnitude to mutate a chromosome. The magnitude is defined using a bounded polynomial spread function to reduce the range of mutation as the optimisation progresses. This is the preferred mutation operator for real number optimisation.

4.5 RandomSelection Object

The random selection operator contains the methods to randomly select chromosomes from a population. It is mainly used to select chromosomes to be mutated

4.5.1 Random Selection Rate "variable randomSelectionRate (Double)"

This variable allows to set the desired random selection rate, which is the average fraction of the population that will be selected. The range of this value is between 0 and 1. When used to pick chromosomes for the mutation operator the optimum values are usually between 0.05 and 0.20, so only a small fraction of the population is picked up for the mutation.

4.6 RoluetteSelection Object

The roulette selection operator contains the methods to select chromosomes from a population using the roulette wheel technique, which consists in affecting to each chromosome an area proportional to their relative fitness on a virtual roulette wheel, this wheel is then spun repeatedly to select the chromosomes for the mutation. This method allows to give a chromosome a chance to be picked proportional to it's relative fitness. This method is normally used to select chromosomes for crossover because it is biased towards the fittest chromosomes.

4.6.1 Roulette Selection Rate "variable rouletteSelectionRate (Double)"

This variable allows to set the desired roulette selection rate, which is the average fraction of the population that will be selected. The range of this value is between 0 and 1. When used to pick chromosomes for the crossover operator the optimum values are usually between 0.15 and 0.40, so a relatively large fraction of the population is picked up for crossover.

4.6.2 Stochastic Universal Sampling Rate "variable SUS-Rate (Double)"

This variable allows to set the desired SUS selection rate, which is the average fraction of the population that will be selected. The range of this value is between 0 and 1. When used to pick chromosomes for the crossover operator the optimum values are usually between 0.10 and 0.30, so a relatively large fraction of the population is picked up for crossover.

4.7 RandomReplacement and RankedReplacement Objects

The RandomReplacement and the RankedReplacement objects contains the methods used at the end of a generation to replace some of the old members of the population by some of the newly generated chromosome.

The tournament replacement method repeatedly create a tournament between some randomly picked chromosomes among the newly generated chromosomes and the old chromosomes, the chromosome with the best fitness will be kept and the other one will be discarded. This method might be a bit less efficient than the ranked replacement but allows to maintain diversity in the population,

The ranked replacement method creates a table including the new and the old chromosomes where the chromosomes are ordered depending of their fitness. The chromosomes with the best fitness are then integrated back in the population. This method is a bit more efficient than the Tournament Replacement but can suffer from premature convergence due to the reduction of diversity.

4.7.1 Elitism "variable elitism (Boolean)"

This variable apply to both RandomReplacement and RankedReplacement, it allows to enable or disable the elitism in the replacement method. The elitism means that the best member of the old population is always kept ant brought back in the new population. Enabling elitism often improve the performances of the optimiser.

- "true" elitism enabled
- "false" elitism disabled

4.8 General Crossover and Mutation Operators Objects

4.8.1 Keep Track Of ID "variable keepTrackOfId (Boolean)"

This setting is common for all crossover and mutation operators, it allows to keep track of the parents ID of a chromosome.

"true" enable the tracking of ID

"false" disable the tracking of ID

The crossover operators objects contains the methods allowing the genetic algorithms to perform recombination of some of the fittest members of the population. This means that the crossover operators take a minimum of two chromosomes, and recombines theirs genes using different techniques. At the time of writing this documentation five crossover methods have been implemented

4.9 OnePointCrossover Object

The one point crossover method consists in randomly selecting a point on the chromosomes gene string then the first part of the first chromosome is paired with the second part of the second chromosome. This method is most suited for binary type chromosomes.

No settings are associated with this method.

4.10 NPointCrossover Object

The N point crossover is a method where a number N of section points of the chromosome are randomly selected then the parts of each chromosome are paired together. This method is most suited for binary type chromosomes.

4.10.1 Number of Crossover Points ¨variable numXover-Points (Integer)"

This setting defines the number of points used for the crossover. The value is an Integer from 0 to the number of genes.

4.11 UniformCrossover Object

The uniform crossover is a method where each gene is randomly selected from one or an other of the parents and included in the new chromosome. this method is suited for both binary or real type encoding.

No settings are associated with this method.

4.12 WeightedAveragingCrossover Object

The weighted averaging crossover is a method where a randomly weighted average between the two parents genes is performed. this is repeated for all the genes. This is an efficient method for real number coded chromosomes like the chromosomes encoding the combustors.

No settings are associated with this method.

4.13 WeightedLinearCrossover Object (BLX-α)

The BLX-α crossover is a method where a randomly weighted point is selected on a virtual line linking the two parents genes. The new point is selected on a range defined by the two parents genes and an exploration factor. this is repeated for all the genes. This is the most efficient method for real number coded chromosomes like the chromosomes encoding the combustors.

4.13.1 Exploration Factor "variable explorationFactor (Double)"

This variable allows to set the exploration factor of the weighted linear crossover. the value to give is a double number superior to 0.

- A value of the factor < 1 will have a tendency to collapse the population around a particular point (reduce diversity)
- A value of the factor $= 1$ will keep the spreading of the population constant. (still a small reduction of diversity)
- A value of the factor > 1 will have a tendency to expand the population. (increasing the diversity)

The usual range of the exploration factor is from 0.8 to 1.5 low values tends to converge the population prematurely.

4.14 BoundedSBXCrossover Object

This is a crossover operator that creates two children from two parents with a probability distribution of the gene values similar to the probability distribution generated by the binary crossover method. It relies on the same basic principle of blending the characteristics of the two parent chromosomes in a manner similar to the BLX-α or the weighted linear averaging. However its uniqueness comes from the definition of a spread factor β and the use of a polynomial distribution function $\overline{\beta}$ to perform the blending between the two parent genes values.

//object BoundedSBXCrossover 0 = max exploration + -> decrease explo

4.14.1 Exploration Factor "variable distibutionIndex (Double)"

This variable allows to set the exploration factor of the weighted linear crossover. the value to give is a double number superior to 0.

- A value of the Index < 1 will have a tendency to expand the population. (increasing the diversity)
- A value of the Index $= 1$ will keep the spreading of the population constant. (still a small reduction of diversity)
- A value of the Index > 1 will have a tendency to decrease the exploration

An Index value of 1 have been reported to give the best results.

4.15 Mutation Operators Objects

The mutation operator objects are used to perform mutation of one gene some selected chromosomes. The mutation correspond to one random change to one random chromosome

4.16 The random mutate object

Is an operator that randomly mutates a random gene over it's all range. It is the most basic form of nutation, it's explorative properties are quite good

however it has been found to be disruptive for genes that have very sensitive settings like ports diameters and etc.. No settings are associated with this object.

The creep mutate object, is an operator that randomly mutates a random gene over a given portion of it's range. It's effect is close to hill climbing, it's explorative properties are limited in range than pure random mutation but it is much less disruptive.

4.17 The creep mutate with decay object

Is an operator that randomly mutates a random gene over a given portion of it's range with the portion of this range reducing with the number of evaluations. It's effect is close to hill climbing with annealing, it's explorative properties are less limited in range than the creep mutation due to a larger range at the beginning of the optimisation and it is much less disruptive due to the reduction of this range when the optimisation tend towards the end and some genes becomes critical.

4.17.1 Creep Size "variable creepSize (Double)"

The creepSize variable is associated with the creep mutate object and the creep mutate with decay object. This variable defines the initial variation range of the mutation in percent. The allowable range for this variable is from 0 to 100. The usual range of this variable varies from 10 to 20 for the creep mutate operator and 20 to 40 for the creep mutate with decay operator.

4.17.2 Creep Decay "variable creepDecay (Double)"

The creepDecay variable is associated with the creep mutate with decay object. This variable defines the reduction of range that takes place at the end of each generation. It is expressed in terms of the percentage reduction of the range. The allowable range for this variable is from 0 to 100. The usual range of this variable is between 0.1 to 5, this decay should be reduces as the number of generation planed for the optimisation increase.

4.18 The Dynamic Vectored Mutation Object

This operator use a vector of random direction and magnitude to mutate a chromosome. The magnitude is defined using a bounded polynomial spread

function to reduce the range of mutation as the optimisation progresses. This is the preferred mutation operator for real number optimisation.

This object has two settings

4.18.1 Initial Distribution Factor "variable initialDistribFact (Double)"

This variable define the initial spread of the mutation, optimum values are around 0.6 to 0.9.

4.18.2 Iteration Dependency Factor "variable iterDependancyFactor(Double)"

This factor define the spread reduction in function of the completion of the optimisation, optimal values are around 1.1 to 2.0.

4.19 Evaluation Objects

The evaluation objects allows different methods for the evaluation of the chromosomes depending on the computer resources available and the problem type.

The serial evaluation object allows to evaluate all the chromosomes serially on a single computer this is the most suited technique for the evaluation of small chromosomes (chromosomes that are not linked with flownet or any other programs). No settings are associated with this object.

The parallel evaluation object allows to evaluate the chromosomes in parallel on a multi CPU workstations when the operating system allows it. No settings are associated with this object.

The distributed evaluation object allows to evaluate the chromosomes in parallel over a distributed network, the most efficient method for big optimisation jobs (ie when the chromosomes are linked to flownet or authors external programs).

4.19.1 Server Name "variable serverName (string)"

The serverName variable is associated with the distributed evaluation object. This variable stores the name of the computer running the server of the optimisation.

4.19.2 Server Port "variable serverPort (Integer)"

The serverPort variable is associated with the distributed evaluation object. This variable stores the port number where the server and the clients will be communicating.

4.19.3 JGA Path "variable JGAPath (String)"

The JGAPath variable is associated with the distributed evaluation object. This variable stores the local path to the jga code.

4.20 Generic Chromosome Objects

An object based on the generic chromosome type is the place where the design variables of the problem to be optimised are stored as well as the methods for evaluating the quality of the designs. At the time being it is necessary to create a new chromosome object for each type of optimisation job. All objects derived from the generic chromosome (ie, all combustor chromosomes and normally all other chromosome type) share the following settings.

4.20.1 Chromosome Length "variable chromosomeLength (Integer)"

This variable defines the number of genes that are contained in the chromosome (ie, the number of variables allowed to change for the optimisation). It's value is a positive integer superior to 0. The maximum number of variables depends only on the capacity of the computer used and the time you are ready to wait for. an optimisation with 25 genes (ie, 25 variables) will take around 4 hours on a 600Mhz Athlon Linux machine.

4.20.2 Number Of Parameters "variable numberOfParametters (Integer)"

This variable defines the number of parameters that will be read from an external file, like a flownet output file. Theses parameters will then be used to calculate the performance parameters used for the different constraints and optimised values to define the fitness of the chromosome. There is no upper limit to the number of parameters.

4.20.3 Number Of Constraints "variable numberOfConstraints (Integer)"

This variable defines the number of constraints constraining the problem. A constraint allows to define a target and a validity range for a performance parameter. For example a pressure drop constraint of 5% with an allowable range of ±5% will constraint the allowable pressure drop between 4.75% and 5.25%.

4.20.4 Number Of Optimised Value "variable numberOfOptimisedValue (Integer)"

This variable defines the number of values that will be optimised (ie, the optimiser will try to maximise or minimise these values).

4.21 Generic Flownet Chromosome Object

This is the general chromosome to be used for optimisations of combustors using flownet. This chromosome should allow to optimise any kind of combustors.

For that purpose it uses 31 constraints that can be switched individually to be range constrained, target constrained, optimised or can be switched off. To calculate the constraints a large list of 500 parameters need to be used. The different constraints and the parameters will be described in details.

For the parameters only the parameters actually used for the calculation of the active constraints needs to be set. As well when the calculation are performed on a large list of parameters like averaging of temperatures or summation of cooling mass flow the number of parameters to be set will depends of the number of elements describing the feature on the model.

4.21.1 Target Constraint Flag list "data targetConstraintFlag (list of Booleans)"

This parameter is a list of booleans of the size of the number of constraints where for each constraint the value true defines that it will be used as a target for the optimisation and the value false means it will not be used as a target.

4.21.2 Range Constraint Flag list "data rangeConstraint-Flag (list of Booleans)"

This parameter is a list of booleans of the size of the number of constraints where for each constraint the value true defines that it will be used as a range constraint for the optimisation and the value false means it will not be used as a range constraint.

4.21.3 Optimisation Constraint Flag list "data optimiseC-onstraintFlag (list of Booleans)"

This parameter is a list of booleans of the size of the number of constraints where for each constraint the value true defines that it will be used as a value to be optimised for the optimisation and the value false means it will not be used as a value to optimise. The optimiseConstraintFlag and the targetConstraintFlag of a constraint should not be set to true together it is either one or the other.

4.21.4 Maximise Optimised Constraint Flag List "data maxOptimiseConstFlag(list of Booleans)"

This parameter is a list of booleans of the size of the number of constraints where for each constraint the value true defines that the constraint to be optimised need to be maximised and the value false means it will need to be minimised.

4.21.5 Constraints Target "data ConstraintsTarget (list of Double)"

This parameter is a list of Doubles of the size of the number of constraints where for each constraint the value defines the target to be that the optimiser try to reach. when the targetConstraintFlag is set the optimiser will try to reach this value exactly or when the optimiseConstraintFlag is set the optimiser will try to reach this value and improve it if it can.

4.21.6 Constraints Lower Range "data ConstraintsLowerRange (list of Double)"

This parameter is a list of Doubles of the size of the number of constraints where for each constraint the value defines the lower allowable range for the value as a fraction of the ConstraintsTarget value. A zero for a value means no check for the lower range.

4.21.7 Constraints Upper Range "data ConstraintsUpperRange (list of Double)"

This parameter is a list of Doubles of the size of the number of constraints where for each constraint the value defines the Upper allowable range for the value as a fraction of the ConstraintsTarget value. A zero for a value means no check for the Upper range.

4.21.8 The generic performances constraints

- Constraint 1 -> Combustor overall pressure drop. Calculation: (P3 -Pcombustor exit)/P3 uses parameters 1 and 2
- Constraint 2 -> Generic pressure drop 1. Calculation: $\frac{P_A - P_B}{P_A}$ uses parameters 10 and 11
- Constraint 3 -> Generic pressure drop 2. Calculation: $\frac{P_A - P_B}{P_A}$ uses parameters 12 and 13
- Constraint 4 -> Generic pressure drop 3. Calculation: $\frac{P_A - P_B}{P_A}$ uses parameters 14 and 15
- Constraint 5 -> Generic pressure drop 4. Calculation: $\frac{P_A - P_B}{P_A}$ uses parameters 16 and 17
- Constraint 6 -> Generic pressure drop 5. Calculation: $\frac{P_A - P_B}{P_A}$ uses parameters 18 and 19
- Constraint 7 -> Calculate the Injector 1 AFR (Air Fuel Ratio). Calculation: MflowOfAirInjector / MflowFuel uses parameters 20 and 22
- Constraint 8 -> Calculate the Injector 2 (pilot) AFR. Calculation: MflowOfAirPilotInjector / MflowPilotFuel uses parameters 21 and 23

- Constraint 9 -> Calculate the zone 1 AFR. Calculation: MflowOfAirZone1 / MflowFuel uses parameters 20 and 24
- Constraint 10 -> Calculate the pilot zone 1 AFR. Calculation: MflowOfAirPilotZone1 / MflowPilotFuel uses parameters 21 and 25
- Constraint 11 -> Calculate the zone 2 AFR. Calculation: MflowOfAirZone2 / TotalMflowFuel uses parameters 20,21 and 26
- Constraint 12 -> Calculate the zone 3 AFR. Calculation: MflowOfAirZone3 / TotalMflowFuel uses parameters 20,21 and 27
- Constraint 13 -> Calculate the zone 4 AFR. Calculation: MflowOfAirZone4 / TotalMflowFuel uses parameters 20,21 and 28
- Constraint 14 -> Calculate the zone 5 AFR. Calculation: MflowOfAirZone5 / TotalMflowFuel uses parameters 20,21 and 29
- Constraint 15 -> Calculate the Combustor Cooling Mass Flow. Calculation: $\sum$ MflowCooling uses parameters 100 to 199
- Constraint 16 -> Calculate the ratio of Wall 1 Cooling Mass Flow. Calculation: $\sum$ MflowW1cooling / TotalMflow uses parameters 100 to 119 and 5
- Constraint 17 -> Calculate the ratio of Wall 2 Cooling Mass Flow. Calculation: $\sum$ MflowW2cooling / TotalMflow uses parameters 120 to 139 and 5
- Constraint 18 -> Calculate the ratio of Wall 3 Cooling Mass Flow. Calculation: $\sum$ MflowW3cooling / TotalMflow uses parameters 140 to 159 and 5
- Constraint 19 -> Calculate the ratio of Wall 4 Cooling Mass Flow. Calculation: $\sum$ MflowW4cooling / TotalMflow uses parameters 160 to 179 and 5
- Constraint 20 -> Calculate the ratio of Wall 5 Cooling Mass Flow. Calculation: $\sum$ MflowW5cooling / TotalMflow uses parameters 180 to 199 and 5
- Constraint 21-> Calculate the Maximum Combustor wall temperature. Calculation: max WallTemp uses parameters 200 to 199
- Constraint 22-> Calculate the Maximum Wall 1 temperature. Calculation: max Wall1Temp uses parameters 200 to 219

- Constraint 23 -> Calculate the Maximum Wall 2 temperature. Calculation: max Wall2Temp uses parameters 220 to 239
- Constraint 24 -> Calculate the Maximum Wall 3 temperature. Calculation: max Wall3Temp uses parameters 240 to 259
- Constraint 25 -> Calculate the Maximum Wall 4 temperature. Calculation: max Wall4Temp uses parameters 260 to 279
- Constraint 26 -> Calculate the Maximum Wall 5 temperature. Calculation: max Wall5Temp uses parameters 280 to 299
- Constraint 27 -> Calculate the Average Combustor wall temperature. Calculation: $\sum$ WallTemp / NumberOfTemp uses parameters 200 to 299
- Constraint 28 -> Calculate the Average Wall 1 temperature. Calculation: $\sum$ Wal1lTemp / NumberOfTemp uses parameters 200 to 219
- Constraint 29 -> Calculate the Average Wall 2 temperature. Calculation: $\sum$ Wall2Temp / NumberOfTemp uses parameters 220 to 239
- Constraint 30 -> Calculate the AverageWall 3 temperature. Calculation: $\sum$ Wall3Temp / NumberOfTemp uses parameters 240 to 259
- Constraint 31 -> Calculate the AverageWall 4 temperature. Calculation: $\sum$ Wall4Temp / NumberOfTemp uses parameters 260 to 279
- Constraint 32 -> Calculate the AverageWall 5 temperature. Calculation: $\sum$ Wall5Temp / NumberOfTemp uses parameters 280 to 299
- Constraint 33 -> Calculate the The mass flow of air recirculating in zone 1 ($\mathrm{W_{P/}}_{Z.RECIRC}$). Calculation: $\sum$ MflowBPPorts + $\frac{1}{2}\sum$ MflowPrimPorts + $\frac{1}{3}\sum$ MflowZ1cooling uses parameters 40 to 44, 100 to 139
- Constraint 34 -> Calculate the The mass flow of air recirculating in pilot zone 1 ($\mathrm{W_{PilotP}}_{Z.RECIRC}$). Calculation: $\sum$ MflowPilotBPPorts + $\frac{1}{2}\sum$ MflowPilotPrimPorts + $\frac{1}{3}\sum$ MflowPilotZ1cooling uses parameters 45 to 49, 160 to 199
- Constraint 35 -> Calculate the overall SI loading. Calculation: $\Lambda_{SI} = \frac{W_3 * 10^9}{\left(P_3^{1.8} * V_{Comb}\right) e^{T_3/300}}$ uses parameters 2,5,6 and 9
- Constraint 36 -> Calculate the Relight loading. Calculation: $\chi_{SI} = \frac{W_{P/Z.RECIRC} * 10^6}{\left(P_{3wm}^{1.3} * V_{PZ}\right) e^{T_{3wm}/300}}$ uses constraint 33 and parameters 30,31and 7. To

calculate this constraint the constraint 33 will be automatically calculated so, the parameters required for the constraint 33 needs to be set.

- Constraint 37 -> Calculate the Relight loading. Calculation: $\chi_{SI} = \frac{W_{PilotPZ.RECIRC}*10^6}{\left(P_{3wm}^{1.3}*V_{PilotPZ}\right)e^{T_{3wm}/300}}$ uses constraint 34 and parameters 30,31 and 7. To calculate this constraint the constraint 34 will be automatically calculated so, the parameters required for the constraint 34 needs to be set.

- Constraint 38 -> Calculate the NOx emissions. Calculation from tables of creation rate uses parameters 300 to 449 and 3

- Constraint 39 -> Calculate the CO emissions. Calculation from tables of creation rate uses parameters 300 to 449 and 3

- Constraint 40 -> Calculate the UHC emissions. Calculation from tables of rates uses parameters 300 to 449 and 3

- Constraint 41 -> Calculate the Soot emissions. Calculation from tables of creation rate uses parameters 300 to 449 and 3

4.21.9 The parameters list

- Parameter 1 -> Total Pressure at compressor delivery P_3.

- Parameter 2 -> Total Pressure at combustor exit.

- Parameter 3 -> Mass flow of fuel.

- Parameter 4 -> Mass flow of air through injector.

- Parameter 5 -> Mass flow of air at combustor inlet (usually element 1).

- Parameter 6 -> Volume of combustor (combustion zone) V_{Comb}.

- Parameter 7 -> Volume of Zone 1 (Recirculation Volume) V_{PZ}.

- Parameter 8 -> Volume of Zone 1 (Recirculation Volume) $V_{PilotPZ}$.

- Parameter 9 -> Compressor Delivery Temperature T_3.

- Parameter 10 -> P_A for generic pressure drop calculation 1.

- Parameter 11 -> P_B for generic pressure drop calculation 1.

- Parameter 12 -> P_A for generic pressure drop calculation 2.

- Parameter 13 -> P_B for generic pressure drop calculation 2.
- Parameter 14 -> P_A for generic pressure drop calculation 3.
- Parameter 15 -> P_B for generic pressure drop calculation 3.
- Parameter 16 -> P_A for generic pressure drop calculation 4.
- Parameter 17 -> P_B for generic pressure drop calculation 4.
- Parameter 18 -> P_A for generic pressure drop calculation 5.
- Parameter 19 -> P_B for generic pressure drop calculation 5.
- Parameter 20 -> Fuel Mass Flow from injector 1.
- Parameter 21 -> Fuel Mass Flow from injector 2 (pilot).
- Parameter 22 -> Mass flow of air through injector 1.
- Parameter 23 -> Mass flow of air through injector 2 (pilot).
- Parameter 24 -> Mass flow of air at end of zone 1.
- Parameter 25 -> Mass flow of air at end of pilot zone 1.
- Parameter 26 -> Mass flow of air at end of zone 2.
- Parameter 27 -> Mass flow of air at end of zone 3.
- Parameter 28 -> Mass flow of air at end of zone 4.
- Parameter 29 -> Mass flow of air at end of zone 5.
- Parameter 30 -> Total Pressure at compressor delivery P_3 for Wind Milling conditions.
- Parametter 31 -> Compressor Delivery Temperature T_3 for Wind Milling conditions.
- Parameter 40 -> Mass flow of air through primary port 1.
- Parameter 41 -> Mass flow of air through primary port 2.
- Parameter 42 -> Mass flow of air through primary port 3.
- Parameter 43 -> Mass flow of air through primary port 4.
- Parameter 44 -> Mass flow of air through primary port 5.

- Parameter 45 -> Mass flow of air through pilot primary port 1.
- Parameter 46 -> Mass flow of air through pilot primary port 2.
- Parameter 47 -> Mass flow of air through pilot primary port 3.
- Parameter 48 -> Mass flow of air through pilot primary port 4.
- Parameter 49 -> Mass flow of air through pilot primary port 5.
- Parameter 100 to 119 -> Wall 1 cooling mass flow (Base Plate).
- Parameter 120 to 139 -> Wall 2 cooling mass flow (Zone1 Walls).
- Parameter 140 to 159 -> Wall 3 cooling mass flow.
- Parameter 160 to 179 -> Wall 4 cooling mass flow (Pilot Base Plate if double annular).
- Parameter 180 to 199 -> Wall 5 cooling mass flow (Pilot Zone1 wall if double annular).
- Parameter 200 to 219 -> Wall 1 temperature (Base Plate).
- Parameter 220 to 239 -> Wall 2 temperature (Zone1 Walls).
- Parameter 240 to 259 -> Wall 3 temperature.
- Parameter 260 to 279 -> Wall 4 temperature (Pilot Base Plate if double annular).
- Parameter 280 to 299 -> Wall 5 temperature (Pilot Zone1 wall if double annular).
- Parameter 300 to 249 -> Flame Path mixed mass flow.
- Parameter 350 to 399 -> Flame Path element Length.
- Parameter 400 to 449 -> Flame Path velocity.

4.22 Gene Writing Interface Object

The setting of the genes allows the toolbox to interact with flownet through the ".net" input file. This is done through the "generic text file interface" which is able to process a text file and modify a number of key parameters.

This file interface search a file (normally the flownet input file) for a given text string either in the whole file or in a designated column or in the designates line.

The general syntax in the configuration file will be:

```
vect key {type name string line column linOffset colOffset geneNo}
```

The locking syntax is:

```
vect key {lock string line column linOffset colOffset}
```

The gene linking syntax is:

```
vect key {gene name string line column linOffset colOffset geneNo}
```

where:

- key - is of the form ix where i means input and x is the number of this input, for the first gene configuration the key will be i1 for the second it will be i2 etc...
- type - defines the type of command to be done lock means to lock on a specified string to go in a specified part of the file and gene means that it will go and modify the specified gene.
- string - is the searched string, either a section when locking on a specified section or an element number when affection a gene to a flownet element.
- line - search for the string in the specified line if line is 0 don't specify line.
- column - search for the string in the specified column if column is 0 don't specify column. practice to search for flownet element numbers which are in column 1.
- linOffset - once the string has been found move line by the specified offset.
- colOffset - once the string has been found move column by the specified offset.
- geneNo - the number of the gene to be affected to this position.

For example we are looking to modify the diameter of outer primary dilution port (OPDP) labelled as element "20" in a flownet input file. This element is located in the Cooling/Dilution Holes (HO) section of the input file, so a lock will be made on the "HO" string which defines the start of this section. We know that element numbers are located in the first column. As well we know that the diameter of the holes is specified in the 5th column

The configuration line link the gene 1 to the element 20 will be the following:

```
vect key {type name string line column linoffset coloffset geneNo}
vect i1  {lock        HO    0     1        0         0           }
vect i2  {gene PDP  "20"  0     1        0         4       1   }
```

The search is done sequentially in the file so the values searched have to be in the same order as in the file.

4.23 Parameter Reading Interface Object

The setting of the parameters allows the toolbox to interact with flownet through the ".res" output file. This is performed through the "generic text file interface" which is able to process a text file and read a number of key parameters.

This file interface search a file (normally the flownet output file) for a given text string either in the whole file or in a designated column or in the designates line.

The general syntax in the configuration file will be:

```
vect key {type name string line column linOffset colOffset paramNo}
```

The locking syntax is:

```
vect key {lock string line column linOffset colOffset}
```

The parameter reading syntax is:

```
vect key {param name string line column linOffset colOffset paramNo}
```

The parameter setting syntax is:

```
vect key {parset name value paramNo}
```

The parameter linking to an other parameter syntax is:

```
vect key {parlink name sourceParamNo paramNo}
```

where:

- key - is of the form rx where r means result and x is the number of this read, for the first read configuration the key will be r1 for the second it will be r2 etc...
- type - defines the type of command to be done lock means to lock on a specified string to go in a specified part of the file, param means that it will go and read in the result file the value of the specified parameter, parset means it will set the specified parameter with the given value, and finally parlink means that the specified parameter will be linked with a source parameter.
- name - is the name given to this parameter.
- string - is the searched string, either a section when locking on a specified section or an element number when the result of an element is affected to a parameter.
- line - search for the string in the specified line if line is 0 don't specify line.
- column - search for the string in the specified column if column is 0 don't specify column. practice to search for flownet element numbers which are in column 1.
- linOffset - once the string has been found move line by the specified offset.
- colOffset - once the string has been found move column by the specified offset.
- value - the value to store in the parameter when type of command is "parset".
- sourceParamNo - the number of the source parameter to be linked with the specified parameter when type of command is "parlink".
- paramNo - the number of the parameter where the value read at this position is to be stored.

For example we are looking to read the mass flow of fuel stored in element "7" in a flownet output file and store it in the parameter 3. This element is located in the elements results section of the result file, so a lock will be made on the "ELEMENT_RESULTS" string which defines the start of this section. We know that element numbers are located in the first column. As well we know that the temperature is specified in the 4th column

The configuration line to store the mass flow of fuel from the element 7 to the parameter 3 will be the following:

```
   vect key {type  name  string       line column linoffset coloffset
paramNo}
   vect r1  {lock        NODE_RESULTS  0      2      0          0           }
   vect r2  {param Mfuel "7"         0      1      0          4
3   }
```

The search is done sequentially in the file so the values searched have to be in the same order as in the file.

Chapter 5

Toolbox Configuration file example

5.1 An example of the toolbox configuration file "AED-Test.cfg" for the optimisation of the "generic combustor 01"

```
// JGA Configuration file
// reading is case sensitive
// comments symbol is -> //
// strings needs the ""
// boolean variable are true or false
// object jga
variable chromosomeType = GenericFlownetChrom
variable initialPopulationRatio = 1.5
variable populationSize = 100
variable optimisationMethod = GAOptimizer
variable debugLevel = 0
// variable randomSeed = 1436134
variable numberOfTrials = 5
// Object JgaClient
variable serverName = osiris
// object Optimizer
variable maxNumberOfGenerations = 0
variable maxNumberOfEvaluations = 10000
variable maxFitness = 1
variable finalPrintPopulation = false
```

```
variable printPopInReport = false
variable reportInterval = 1
// variable OutputFilesName = AED-SBX-1000K
// object GAOptimiser
variable objPopInitialiser = RandomInitialisation
variable objCrossoverSelector = StochasticUniversalSampling
variable objMutationSelector = RandomSelection
variable objReplacementOperator = RankedReplacement
variable objCrossoverOperator = BoundedSBXCrossover
variable objMutationOperator = DynamicVectorMutate
variable objEvaluationOperator = SerialEvaluation
// variable objReplacementOperator = TournamentReplacement
// variable objCrossoverSelector = RouletteSelection
// variable objEvaluationOperator = DistributedEvaluation
// variable objCrossoverOperator = WeightedLinearCrossover
// variable objMutationOperator = CreepMutateWithDecay
// object chromosomeModifier/Selector
variable storeChromosomes = true
variable keepTrackOfId = true
// object DistributedEvaluation
variable JGAPath = jga/
variable ServerPort = 5000
// object sons of Chromosome
// Parameters fixed to 500 for genericflownet chromosome
// Constraints fixed to 49 for genericflownet chromosome
variable selectionPressure = 2
variable printEvaluationResults = false
variable chromosomeLength = 25
variable numberOfParameters = 500
variable numberOfConstraints = 49
variable networkPath = /tmp/fl/
variable originalFileName = network1b.net
// object StochasticUniversalSampling
variable SUSRate = 0.20
// object RouletteSelection
variable rouletteSelectionRate = 0.20
//object BoundedSBXCrossover 0 = max exploration + -> decrease explo
variable distibutionIndex = 1.0
// object RandomSelection
variable randomSelectionRate = 0.05
```

// object DynamicVectorMutate

variable initialDistribFact = 0.775

variable iterDependancyFactor = 1.6

// object ReplacementOP

variable elitism = true

// Constraints |———- pressure drops ——————|———————— AFRs ————————— —|—————- Cooling Massflow ————-|———– Maximum wall temperature ———-|———– Average Wall Temperature ———-|—- WPZRE —-|- DSI -|—– XSI —–|—– Emissions Modelling —–|—————— ——– Mach number ratio ——————-|

// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49

data targetConstraintFlag = { true false true true false false true false true false true false false true true true false false false false false false false false false false true false true true false false true false true true false false false false false true true true true false false false false }

data rangeConstraintFlag = { true false true true false false true false true false true false false true true true false false false false true false true true false false true false true true false false true false true true false true false false false true true true true false false false false }

data optimiseConstraintFlag = { false }

data maxOptimiseConstFlag = { false true false false false false false false false false false false false false }

data constraintsTarget = { 0.0564 0 0.0358 0.0376 0 0 6.358 0 24.877 0 25.951 0 0 38.306 0.430 0.1951 0.0482 0.1864 0 0 850.00 0 850.00 850.00 0 0 788.00 0 788.0 788.0 0 0 9.578 0 1.007 10.363 0 25.910 0 0 0 2.274 2.555 3.204 3.846 0 0 0 0 }

data constraintsLowerRange = { 0.950 0 0.900 0.900 0 0 0.950 0 0.950 0 0.950 0 0 0.900 0.900 0.900 0.900 0.900 0 0 0.100 0 0.100 0.100 0 0 0.800 0 0.800 0.800 0 0 0.900 0 0.900 0.950 0 0.100 0 0 0 0.800 0.800 0.800 0.800 0 0 0 0 }

data constraintsUpperRange = { 1.050 0 1.100 1.100 0 0 1.050 0 1.050 0 1.050 0 0 1.100 1.100 1.000 1.100 1.100 0 0 1.000 0 1.000 1.000 0 0 1.050 0 1.050 1.050 0 0 1.100 0 1.100 1.050 0 1.000 0 0 0 1.200 1.200 1.200 1.200 0 0 0 0 }

// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

// 25 genes boundaries OPDP IPDP OIDP IIDP OEC IEC OZR1 IZR1 OZR2 IZR2 OZR3 IZR3 OZR4 IZR4 OZR5 FI OBPC1 IBPC1 IBPC2 OBPC2 OBPC3 IBPC3 OBPC4 IBPC4 OBPC5

data geneLowerBound = { 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 }

//data geneLowerBound = { 1.0e-5 1.0e-5 1.0e-5 1.0e-5 7.0e-8 7.0e-8 1.1e-7 1.1e-7 1.1e-7 1.1e-7 1.1e-7 1.1e-7 1.1e-7 1.1e-7 2.0e-7 9.7e-5 1.5e-6 2.2e-6 1.1e-6 2.4e-6 3.0e-6 2.5e-6 3.2e-7 3.6e-7 1.5e-5 }

data geneUpperBound = { 8.0e-4 8.0e-4 8.0e-4 8.0e-4 7.9e-7 7.9e-7 3.0e-7 3.0e-7 3.0e-7 3.0e-7 3.0e-7 3.0e-7 3.0e-7 3.0e-7 2.5e-6 3.0e-4 5.0e-6 7.0e-6 3.5e-6 1.4e-5 8.0e-6 1.6e-5 8.0e-7 8.0e-7 4.0e-5 }

// original gene values

// data geneLowerBound = { 2.75e-4 2.85e-4 7.60E-5 1.10e-4 1.98E-7 1.96E-7 2.21E-7 2.21E-7 2.19E-7 2.21E-7 2.20E-7 2.22E-7 2.20E-7 2.21E-7 1.06E-6 1.98e-4 3.25E-6 4.57E-6 2.36E-6 5.02E-6 6.23E-6 5.24E-6 6.56E-7 7.38E-7 3.11E-5 }

// data geneLowerBound = { 0.00027521 0.000285088 7.60E-05 0.000110036 1.98E-07 1.96E-07 2.21E-07 2.21E-07 2.19E-07 2.21E-07 2.20E-07 2.22E-07 2.20E-07 2.21E-07 1.06E-06 0.000198098 3.25E-06 4.57E-06 2.36E-06 5.02E-06 6.23E-06 5.24E-06 6.56E-07 7.38E-07 3.11E-05 }

// data geneUpperBound = { 0.00027521 0.000285088 7.60E-05 0.000110036 1.98E-07 1.96E-07 2.21E-07 2.21E-07 2.19E-07 2.21E-07 2.20E-07 2.22E-07 2.20E-07 2.21E-07 1.06E-06 0.000198098 3.25E-06 4.57E-06 2.36E-06 5.02E-06 6.23E-06 5.24E-06 6.56E-07 7.38E-07 3.11E-05 }

```
// max zr5 <= 8.929e-5 oderwise flownet will crash
// flownet generic chromosome
// 0 -> search on all line/column
// vector i1,i2,i3,...,ix
// Input File Writing
// lock -> input file locator
// gene -> position for writing the gene there must be as many genes as
// specified in chromosomeLength.
// key Str Str Str Int Int Int Int Int
// vect i1 { lock position line column linOfset colOfset }
// vect i2 { gene name position line column linOfset colOfset geneNo}
vect i1 { lock HO 0 1 0 0 }
// Baseplate Cooling
vect i2 { gene IBPC1 "34" 0 1 0 4 18 }
vect i3 { gene OBPC3 "11" 0 1 0 4 21 }
vect i4 { gene OBPC4 "8" 0 1 0 4 23 }
vect i5 { gene OBPC2 "16" 0 1 0 4 20 }
vect i6 { gene IBPC3 "12" 0 1 0 4 22 }
vect i7 { gene OBPC1 "17" 0 1 0 4 17 }
vect i8 { gene IBPC2 "33" 0 1 0 4 19 }
// Fuel Injector
vect i9 { gene FI "13" 0 1 0 4 16 }
vect i10 { gene OBPC5 "28" 0 1 0 4 25 }
vect i11 { gene IBPC4 "9" 0 1 0 4 24 }
// Outer primary and secondary dilution port
vect i12 { gene OPDP "20" 0 1 0 4 1 }
vect i13 { gene OIDP "38" 0 1 0 4 3 }
// Inner Primary and secondary dilution port
vect i14 { gene IPDP "43" 0 1 0 4 2 }
vect i15 { gene IIDP "42" 0 1 0 4 4 }
// Inner and Outer primary effusion cooling
vect i16 { gene OPEC "23" 0 1 0 4 5 }
vect i17 { gene IPEC "48" 0 1 0 4 6 }
// Outer Zrings
vect i18 { gene OZR1 "19" 0 1 0 4 7 }
vect i19 { gene OZR2 "29" 0 1 0 4 9 }
vect i20 { gene OZR3 "30" 0 1 0 4 11 }
vect i21 { gene OZR4 "37" 0 1 0 4 13 }
vect i22 { gene OZR5 "3" 0 1 0 4 15 }
// Inner Zrings
```

```
vect i23 { gene IZR1 "44" 0 1 0 4 8 }
vect i24 { gene IZR2 "45" 0 1 0 4 10 }
vect i25 { gene IZR3 "46" 0 1 0 4 12 }
vect i26 { gene IZR4 "47" 0 1 0 4 14 }
// Output Fle Reading
// Rlock1 -> output file locator
// param1 -> position for reading the param
// key Str Str Str Int Int Int Int Int
// vect r1 { lock position line column linOfset colOfset }
// vect r2 { param name position line column linOfset colOfset paramNo}
// combustor and Primary Zone volume calculated
// vect r1 { parset Vcomb 0.02747 6 }
// vect r2 { parset VPZ 0.01009 7 }
// combustor and Primary Zone volume RR data
vect r1 { parset Vcomb 0.03848 6 }
vect r2 { parset VPZ 0.02329 7 }
// pressure and temperature fot wind milling conditions P3(Psi) T3(K)
vect r3 { parset P3wind 700000 30 }
vect r4 { parset T3wind 400 31 }
// length of flame path elements
vect r4 { parset LPZ1 0.01318 400 }
vect r5 { parset LPZ2 0.01318 401 }
vect r6 { parset LPZ3 0.04215 402 }
vect r7 { parset LPZ4 0.01608 403 }
vect r8 { parset LPZ5 0.03179 404 }
vect r9 { parset LIZ1 0.01545 405 }
vect r10 { parset LIZ2 0.00001 406 }
vect r11 { parset LDZ1 0.03562 407 }
vect r12 { parset LDZ2 0.02481 408 }
vect r13 { parset LDZ3 0.01445 409 }
// Volume of flame path elements
vect r14 { parset VPZ1 0.00003124 450 }
vect r15 { parset VPZ2 0.00002059 451 }
vect r16 { parset VPZ3 0.00715707 452 }
vect r17 { parset VPZ4 0.00288877 453 }
vect r18 { parset VPZ5 0.00586843 454 }
vect r19 { parset VIZ1 0.00283199 455 }
vect r20 { parset VIZ2 0.00000172 456 }
vect r21 { parset VDZ1 0.00522367 457 }
vect r22 { parset VDZ2 0.00260195 458 }
```

```
vect r23 { parset VDZ3 0.00103643 459 }
vect r24 { lock ELEMENT_RESULTS 0 2 5 0 }
// compressor delivery temperature
vect r25 { param T3 "1" 0 1 0 11 9 }
// Outer casing temperature
vect r26 { param OCT1 "77" 0 1 0 8 220 }
vect r27 { param OCT2 "78" 0 1 0 8 221 }
vect r28 { param OCT3 "80" 0 1 0 8 240 }
vect r29 { param OCT4 "81" 0 1 0 8 241 }
vect r30 { param OCT5 "83" 0 1 0 8 242 }
vect r31 { param OCT6 "84" 0 1 0 8 243 }
// Inner casing temperature
vect r32 { param ICT1 "85" 0 1 0 8 222 }
vect r33 { param ICT2 "86" 0 1 0 8 223 }
vect r34 { param ICT3 "88" 0 1 0 8 244 }
vect r35 { param ICT4 "89" 0 1 0 8 245 }
vect r36 { param ICT5 "91" 0 1 0 8 246 }
vect r37 { param ICT6 "92" 0 1 0 8 247 }
// lock the second part of the elements results
vect r38 { lock ELEM 0 1 0 0 }
//flame path flow velocity
vect r39 { param VDZ3 "6" 0 1 0 5 359 }
//outer pp Mratio
vect r40 { param MOPP "20" 0 1 0 8 60 }
vect r41 { param MOPPL "22" 0 1 0 9 70 }
vect r42 { param VPZ4 "26" 0 1 0 5 353 }
vect r43 { param VPZ3 "31" 0 1 0 5 352 }
// M ratio
vect r44 { param MOIP "38" 0 1 0 8 62 }
vect r45 { param MIIP "42" 0 1 0 8 63 }
vect r46 { param MIPP "43" 0 1 0 8 61 }
vect r47 { param MOIPL "49" 0 1 0 9 72 }
vect r48 { param MIPPL "54" 0 1 0 9 71 }
vect r49 { param MIIPL "57" 0 1 0 9 73 }
vect r50 { param VPZ5 "65" 0 1 0 5 354 }
vect r51 { param VIZ1 "66" 0 1 0 5 355 }
vect r52 { param VIZ2 "67" 0 1 0 5 356 }
vect r53 { param VDZ1 "68" 0 1 0 5 357 }
vect r54 { param VDZ2 "69" 0 1 0 5 358 }
vect r55 { param VPZ2 "109" 0 1 0 5 351 }
```

```
vect r56 { param VPZ1 "110" 0 1 0 5 350 }
// jump to the node result part
vect r57 { lock NODE_RESULTS 0 2 0 0 }
// input total pressure
vect r58 { param P0in "1" 0 1 0 1 1 }
vect r59 { param Pa1 "6" 0 1 0 1 12 }
// fuel mass source
vect r60 { param Mfuel "7" 0 1 0 4 20 }
vect r61 { param Pa2 "14" 0 1 0 1 14 }
vect r62 { param Pb1 "28" 0 1 0 1 2 }
// jump to the summary table_1
vect r63 { lock SUMMARY_TABLE_1 0 2 0 0 }
// total massflow through combustor
vect r64 { param Mftot "1" 0 1 0 1 5 }
// base plate cooling air mass flow
vect r65 { param MfIBPC1 "34" 0 1 0 1 100 }
vect r66 { param MfOBPC3 "11" 0 1 0 1 101 }
vect r67 { param MfOBPC4 "8" 0 1 0 1 102 }
vect r68 { param MfOBPC2 "16" 0 1 0 1 103 }
vect r69 { param MfIBPC3 "12" 0 1 0 1 104 }
vect r70 { param MfOBPC1 "17" 0 1 0 1 105 }
vect r71 { param MfIBPC2 "33" 0 1 0 1 106 }
// injector air mass flow
vect r72 { param MfInj "13" 0 1 0 1 22 }
vect r73 { param MfOBPC5 "28" 0 1 0 1 107 }
vect r74 { param MfIBPC4 "9" 0 1 0 1 108 }
// Outer Ports
vect r75 { param MfOPEC "20" 0 1 0 1 40 }
vect r76 { param MfOPEC "38" 0 1 0 1 50 }
// flame tube cooling air mass flow Mflow
vect r77 { param MfOPEC "23" 0 1 0 1 140 }
vect r78 { param MfOZR1 "19" 0 1 0 1 120 }
vect r79 { param MfOZR2 "29" 0 1 0 1 141 }
vect r80 { param MfOZR3 "30" 0 1 0 1 142 }
vect r81 { param MfOZR4 "37" 0 1 0 1 143 }
vect r82 { param MfOZR5 "3" 0 1 0 1 144 }
// Inner Ports
vect r83 { param MfIPEC "43" 0 1 0 1 41 }
vect r84 { param MfIPEC "42" 0 1 0 1 51 }
vect r85 { param MfIPEC "48" 0 1 0 1 145 }
```

```
vect r86 { param MfIZR1 "44" 0 1 0 1 121 }
vect r87 { param MfIZR2 "45" 0 1 0 1 146 }
vect r88 { param MfIZR3 "46" 0 1 0 1 147 }
vect r89 { param MfIZR4 "47" 0 1 0 1 148 }
//flame path mixed flow
vect r90 { param MfPZ1 "110" 0 1 0 2 300 }
vect r91 { param MfPZ2 "109" 0 1 0 2 301 }
vect r92 { param MfPZ3 "31" 0 1 0 2 302 }
vect r93 { param MfPZ4 "26" 0 1 0 2 303 }
vect r94 { param MfPZ5 "65" 0 1 0 2 304 }
vect r95 { param MfIZ1 "66" 0 1 0 2 305 }
vect r96 { param MfIZ2 "67" 0 1 0 2 306 }
vect r97 { param MfDZ1 "68" 0 1 0 2 307 }
vect r98 { param MfDZ2 "69" 0 1 0 2 308 }
vect r99 { param MfDZ3 "6" 0 1 0 2 309 }
// generic pressure drop b 1
vect r100 { parlink pb2 62 13 }
// generic pressure drop b 2
vect r101 { parlink pb2 62 15 }
// primary zone exit massflow elem 65
vect r102 { parlink MfPZ 94 24 }
// intermeditezone exit massflow elem 67
vect r103 { parlink MfIZ 96 26 }
// total mass flow out of combustor Mflow elem 6
vect r104 { parlink Mfout 99 29 }
```

5.2 The network file of the "generic combustor 01"

```
Flownet Input File
============================== GENERAL DATA ==================================================
Tamb         InitPres    InitTemp    Elevation   CalcAmbTemp
[K]          [Pa]        [K]         [m]         (y/n)
0            1E+06       674         0           y
VarGasProp   Rgas        Density     Mu          Cp          Prandtl     ThermConduc
(y/n)        [J/kg/K]    [kg/m^3]    [kg/m/s]    [J/kg/K]    number      [W/m/K]
y            288.21      1           0           0           0           0
Compflow     SolveTemp   NumIterMain NumIterPres NumIterTemp FlowConverge TempConverge
y            y           100         50          3           1E-07       1E-07
```

```
RelaxPres   RelaxRHO    RelaxMass   RelaxOR     RelaxmDG    RelaxCAU    RelaxEnergy
0.4         0.4         0.4         0.4         0.4         0.8           0.4
DiamFact    RoughFact   Nx          C/H_Ratio   Flame_wall_emis Annuli_wall_emis Case_emis
1           1           28          6.21        0.7             0.7              0.7
WarningMes  CalcOrifice RefNode
n           n           28
RecircEntr  Swirl       SwirlerDiam HubDiameter
(y/n)       number      [m]         [m]
y           1.0         0.07        0.04
NumComAirSt(0,1,2)  Mixing(y/n) EquilibriumCalc(y/n) FracTempFile1 FracTempFile2
1                   y           y                   ./data/fuelfrac1.dat
============================== CLASS DEFINITION ============================================
Class     Name   {up to 100 classes allowed
1         Diffuser section
2         Baseplate
3         Outer flametube
4         Outer annulus
5         Inner flametube
6         Inner annulus
7         Combustion zone
8         NGV
9         Inner annulus liner cooling slots
10        Baseplate cooling slots
11        HP Turbine Bleed
12        Outer FHT
13        Outer CD
14        Outer AHT
15        Inner FHT
16        Inner CD
17        Inner AHT
18        **** No Description ****
19        **** No Description ****
20        **** No Description ****
0
============================== ELEMENT DATA ================================================
DW ------------------------------ Darcy-Weisbach Pipes (Type 1) --------------------------------
Elem  Class N1    N2    Length  Diam  Rough  SigK  OrRat  Flag
-     -     -     -     [m]     [m]   [m^-6] -     -      1 Heat[kW]
-     -     -     -     -       -     -      -     -      2 WallTh  InsTh  Ho          Kpipe    Kins     To
-     -     -     -     -       -     -      -     -        [m]     [m]    [W/m^2.K]  [W/m.K]  [W/m.K]  [K]
```

-	-	-	-	-	-	-	-	-	3 Texit[K]

0

HW ------------------------------ Hazen-Williams Pipes (Type 12) ------------------------------

Elem	Class	N1	N2	Length	Diam	Rough	SigK	OrRat	Flag					
-	-	-	-	[m]	[m]	Coeff	-	-	1 Heat[kW]					
-	-	-	-	-	-	-	-	-	2 WallTh	InsTh	Ho	Kpipe	Kins	To
-	-	-	-	-	-	-	-	-	[m]	[m]	[W/m^2.K]	[W/m.K]	[W/m.K]	[K]
-	-	-	-	-	-	-	-	-	3 Texit[K]					

0

DG ------------------------------ Duct with area change (Type 3) ------------------------------

Elem	Class	N1	N2	Length	InA/D	InP	OutA/D	OutP	Rough	SigK	Flag	
-	-	-	-	[m]	[m^2]	[m]	[m^2]	[m]	[m]	-	1	Heat[kW]
-	-	-	-	-	-	-	-	-	-	-	2	-
1	1	1	2	0.006595	0.03059	2.653	0.03059	2.653	1E-26	0	1	0
2	1	2	3	0.03696	0.03059	2.653	0.04065	2.654	1E-26	0	1	0
21	4	6	15	0.02798	0.03367	3.587	0.03508	3.621	1E-26	0	1	0
22	4	15	16	0.01634	0.02506	3.653	0.02296	3.66	1E-26	0	1	0
70	4	16	23	0.029118	0.02296	3.66	0.0198	3.673	1E-26	0	1	0
49	4	23	24	0.02043	0.0198	3.673	0.01625	3.686	1E-26	0	1	0
50	4	24	25	0.01836	0.01625	3.686	0.02262	3.663	1E-26	0	1	0
51	4	25	26	0.03607	0.02262	3.663	0.02916	3.641	1E-26	0	1	0
52	4	26	27	0.02734	0.02916	3.641	0.04073	3.603	1E-26	0	1	0
53	6	14	29	0.03757	0.02714	1.729	0.02831	1.735	1E-26	0	1	0
54	6	29	30	0.02689	0.02831	1.735	0.02526	1.714	1E-26	0	1	0
56	6	30	32	0.03469	0.02526	1.714	0.02349	1.758	1E-26	0	1	0
57	6	32	33	0.02221	0.02349	1.758	0.01987	1.895	1E-26	0	1	0
58	6	33	34	0.02367	0.01987	1.895	0.01933	2.071	1E-26	0	1	0
59	6	34	35	0.02688	0.01933	2.071	0.01799	2.257	1E-26	0	1	0
60	6	35	36	0.01604	0.01799	2.257	0.02184	2.395	1E-26	0	1	0
61	6	36	37	0.02842	0.02184	2.395	0.0261	2.626	1E-26	0	1	0
63	6	37	44	0.02507	0.0261	2.626	0.04008	2.724	1E-26	0	1	0
110	7	70	7	0.01318	0.00237008	3.4	0.00237008	3.4	1E-26	0	2	
109	7	71	7	0.01318	0.00156202	1.987	0.00156202	1.987	1E-26	0	2	
31	7	7	18	0.04215	0.1603	2.694	0.1793	2.728	1E-26	0	2	
26	7	18	19	0.01608	0.1793	2.728	0.18	2.746	1E-26	0	2	
65	7	19	38	0.03179	0.18	2.746	0.1892	2.806	1E-26	0	2	
66	7	38	39	0.01545	0.1892	2.806	0.1774	2.83	1E-26	0	2	
67	7	39	40	1E-05	0.1774	2.83	0.1658	2.907	1E-26	0	2	
68	7	40	41	0.03562	0.1658	2.907	0.1275	3.038	1E-26	0	2	
69	7	41	42	0.02481	0.1275	3.038	0.08225	0.08141	1E-26	0	2	

```
6     7     42    28    0.01445     0.08225     0.08141     0.0612      3.144       1E-26       0           2
7     8     27    5     0.02331     0.04073     3.603       0.04659     3.531       1E-26       0           1
10    8     44    9     0.02151     0.04008     2.724       0.04811     2.973       1E-26       0           1
0
GE ------------------------------ General Empirical Relationship (Type 10) ---------------------
Elem  Class N1    N2    Ck          Beta        Alpha       Heat[kW] {delp0 = ck * (rho^beta)
* (q^alpha)}
0
RD ------------------------------ Restrictor with discharge coefficient (Type 15) --------------
Elem  Class N1    N2    Cd          Area[m^2]   Number
-     -     -     -     -           -Diam[m]    -
0
RL ------------------------------ Restrictor with loss coefficient (Type 16) -------------------
Elem  Class N1    N2    Cc          Cl          Area[m^2]   Number
-     -     -     -     -           -           -Diam[m]    -
0
OR ------------------------------ British Standard Orifice (Type 17) ---------------------------
Elem  Class N1    N2    D[m]        d[m]        Number      Flag
-     -     -     -     -           -           -           1  Corner tappings
-     -     -     -     -           -           -           2  D & D/2 tappings
-     -     -     -     -           -           -           3  Flange tappings
0
PD ------------------------------ Pressure drop with loss coefficient (Type 26) ----------------
Elem  Class N1    N2    Cl          Area[m^2]   J1    InitMass    InitPres    PresProf    PresData    TempProf
-     -     -     -     -           -Diam[m]    -      [kg/s]       [pa]       (0-3)       -            (0-3)
5     1     3     13    0.2         0.06904     2     0           0           0           0           0
18    1     3     14    0.41        0.03048     1     0           0           0           0           0
4     1     3     6     0.45        0.03367     1     0           0           0           0           0
0
HO ------------------------------ Cooling/Dilution Holes (Type 20) ------------------------------
Elem  Class N1    N2    Area[m^2]   Num   Mix           Pres  Cond  Flag1
-     -     -     -     -Diam[m]    -     flag           Op   Node  1(Cd)         Cd          Length  PlunRad
J2 J3 J4
-     -     -     -     -           -     0-def        0-Tot  -     2(Lucas)    Flag2           [m]     [m]
-     -     -     -     -           -     1-ph         1-St   (0-def)           1     Length  PlunRad
Z/W     Alpha   MchCrss If PresOp=2,3:  J1 J2 J3 J4
-     -     -     -     -           -           prmix  2-RTot -                 2     Length  PlunRad
Z/W                     J1 J2 J3 J4
-     -     -     -     -           -     2-oh         3-RSt  -     3(NASA1)    Flag2
-     -     -     -     -           -     3-slot        -     -                 1     Length  PlunRad
MchCrss  If PresOp=2,3:  J1 J2 J3 J4
```

```
-     -     -     -     -           -           dpth     -     -               2     Length  PlunRad
J2 J3 J4
-     -     -     -     -           -     4-ptch         -     -     4(NASA2)  Flag2
-     -     -     -     -           -           lgth     -     -               1     Length  PlunRad
MchCrss  If PresOp=2,3:  J1 J2 J3 J4
-     -     -     -     -           -     5-rev          -     -               2     Length  PlunRad
J2 J3 J4
-     -     -     -     -           -     -              -     -     5(PrkSdII)  Flag2
-     -     -     -     -           -     -              -     -               1     Length  PlunRad
MchCrss  If PresOp=2,3:  J1 J2 J3 J4
-     -     -     -     -           -     -              -     -               2     Length  PlunRad
J2 J3 J4
-     -     -     -     -           -     -              -     -     6(B.E.D.)   Flag2
-     -     -     -     -           -     -              -     -               1     Length  PlunRad
MchCrss  If PresOp=2,3:  J1 J2 J3 J4
-     -     -     -     -           -     -              -     -               2     Length  PlunRad
J2 J3 J4
-     -     -     -     -           -     -              -     -     7(Tr1-4/1)
J2 J3 J4
-     -     -     -     -           -     -              -     -     8(Tr1-2/5)
J2 J3 J4
-     -     -     -     -           -     -              -     -     9(Tr1-2/6)
J2 J3 J4
-     -     -     -     -           -     -              -     -     10(AngEffu)   Length
J2 J3 J4
34    2     13    7     4.5743E-06  108   0              2     0     2           2     0.0018   0        0.0104
110 31
11    2     13    70    6.23193E-06 162   0              2     0     2           2     0.0018   0        0.00993
31 31
8     2     13    70    6.56092E-07 972   0              2     0     2           2     0.002682
0        0.00172                5  5 31 31
16    2     13    70    5.01939E-06 144   0              2     0     2           2     0.0018   0        0.01101
31 31
12    2     13    71    5.23861E-06 144   0              2     0     2           2     0.0018   0        0.0073
31 31
17    2     13    7     3.24566E-06 144   0              2     0     2           2     0.0018   0        0.011
110 31
33    2     13    71    2.35991E-06 162   0              2     0     2           2     0.0018   0        0.00696
31 31
13    2     13    7     0.000198098 18    0              2     0     1             1           0.02     0
110 31
28    2     13    7     3.11327E-05 40    0              2     0     1             0.75        0.00802  0
110 31
9     2     13    71    7.3846E-07  576   0              2     0     2           2     0.0018   0        0.00177
31 31
20    3     16    19    0.00027521  18    1    0.5       2     0     2           2     0.00312  0.00312  0.09827
70 26 65
38    3     24    39    7.60248E-05 36    2              2     0     2           2     0.0016   0        0.04956
50 66 67
```

```
43    5     30    19    0.000285088 18    1    0.5      2     0     2           2     0.003175
0.003175 0.0522               54 56 26 65

42    5     33    39    0.000110036 36    2            2     0     2           2     0.0016   0        0.02901
58 66 67

14    11    29    4     0.000490874 18    2            2     0     2           2     0.0025   0        0.0424
54 31 31

23    3     16    19    1.97524E-07 3200  4    0.008518 2     0     10            0.002485
70 26 65

48    5     30    19    1.96462E-07 1276  4    0.0085   2     0     10            0.0025
56 26 65

19    3     15    18    2.20527E-07 4850  3    0.005009 2     0     1             0.8          0.0025885
0                    21 22 31 26

29    3     23    38    2.19175E-07 4850  3    0.00536  2     0     1             0.8          0.002363
0                    70 49 65 66

30    3     25    40    2.20475E-07 4750  3    0.00594  2     0     1             0.8          0.002516
0                    50 51 67 68

37    3     26    41    2.20475E-07 4750  3    0.00583  2     0     1             0.8          0.00248
0                     51 52 68 69

3     3     27    42    1.06062E-06 600   3    0.0045   2     0     1             0.8          0.0048  0
69  6

44    5     29    18    2.21143E-07 3400  3    0.005417 2     0     1             0.8          0.002912
0                    53 54 31 26

45    5     32    38    2.20544E-07 3240  3    0.011011 2     0     1             0.8          0.004024
0                    56 57 65 66

46    5     34    40    2.22326E-07 3400  3    0.005819 2     0     1             0.8          0.002445
0                    58 59 67 68

47    5     36    41    2.20566E-07 3570  3    0.007343 2     0     1             0.8          0.002498
0                    60 61 68 69

0

FA ------------------------------ Fan (Type 2) and special pressure drop (Type 4) element data
-

Elem  Class  Type  N1    N2    Flow      P1  P2  P3  P4  P5  P6    Q1  Q2  Q3  Q4  Q5  Q6    Refdens    Heat

-     -      2/4   -     -     [m^3/s] <-----------Pa-----------><---------m^3/s-----------> [kg/m^3]   [kW]

0

CA ------------------------------ CAU data -------------------------------------------------------

CompElem  Class  N1  N2  TurbElem  N1  N2  LeakElem  N1  N2  CompFile  TurbFile  LeakFile  MatchFile  Speed[rev

0

HX ------------------------------ Heat exchanger element data (Type 7) -------------------------

Elem1  Class  N1  N2  Elem2  N1  N2  Effop  Flag

-      -      -   -   -      -   -   (1/2)  1  HeatFile

-      -      -   -   -      -   -          2  Eff   Ck  Beta  Alpha   Ck  Beta  Alpha

-      -      -   -   -      -   -          -        <--- Elem1 --->   <--- Elem2 --->

0

EV ------------------------------ Evapourator element data (Type 8) -----------------------------

Elem  Class  N1  N2  Flag

-     -      -   -   1    EvapFile
```

```
-     -      -   -   2    Eff  Ck  Beta  Alpha
0
PR ------------------------------ Pressure relief valve data (Type 13) ------------------------
Elem  Class  N1  N2  ReliefFile
0
PG ------------------------------ Pressure regulating valve data (Type 14) ---------------------
Elem  Class  N1  N2  InitMass[kg/s]  InitPres[Pa]  ReguFile
0
CM ------------------------------ Compressor (Type 5) -----------------------------------------
Elem  Class  N1  N2  Speed[revs/sec]  CompFile
0
TU ------------------------------ Turbine (Type 5) --------------------------------------------
Elem  Class  N1  N2  Speed[revs/sec]  TurbFile
0
US ------------------------------ User specified flow characteristic element (Type 19) ---------
Elem  Class N1    N2    Flag
-     -     -     -     1  UspesFile
-     -     -     -     2  Factor .  Next line :  Row of m*sqrt[T0]/P0 values followed by
                           row of corresponding P01/P02 values.  Row entries end with -1.
0
CP ------------------------------ Compressible Pipes (Type 21) --------------------------------
Elem  Class N1    N2    Length  Diam  Rough  NumInc  Flag
-     -     -     -     [m]     [m]   [m^-6] -       1 Heat[kW]
-     -     -     -     -       -     -      -       2 WallTh  InsTh  Ho         Kpipe    Kins     To
-     -     -     -     -       -     -      -         [m]     [m]    [W/m^2.K]  [W/m.K]  [W/m.K]  [K]
-     -     -     -     -       -     -      -       3 Texit[K]
-     -     -     -     -       -     -      -       4 -
0
SW ------------------------------ Swirler (Type 22) -------------------------------------------
Elem  Class N1    N2    vane    vane    swirler    flametube    Number
-     -     -     -     const   angle    area        area       -
-     -     -     -     -       [deg]    [m^2]       [m^2]      -
0
FHT ------------------------------ Flametube Heat Transfer (Type 23) ----------------------------
Elem  Class N1    N2    h/sl  Gas   Film  Out/  ConvectFlag          (** all dimensions metres
**)                                             RadFlag
-     -     -     -     elem  elem  elem  Inn   1     WallHeat[W]
-     -     -     -     0-Def 0-Def 0-Def 0-Def 2(L)                                              (*
No film cooling *)              2(L)
-     -     -     -     -     -     -     1-Out 3(Pt) xdist                                       (*
No film cooling *)              3(RR) -
```

```
-     -     -     -     -     -     -     2-Inn 4(L)  lgth              depth       skthk
Slot films *)              -     -

-     -     -     -     -     -     -     -     5(RR) etaflag
Film cooling *)              -     -

-     -     -     -     -     -     -     -     -     1     data        lgth        depth       beta        blo
hp    hd     ha  hta  -     -

-     -     -     -     -     -     -     -     -     2     lgth        depth       beta        hp          htd
sklgth          -     -

-     -     -     -     -     -     -     -     -     3     lgth        depth       beta        bloc        skl

-     -     -     -     -     -     -     -     -     4     lgth        depth       hradptch    hcirptch

-     -     -     -     -     -     -     -     6(RR) data  distance
Transply films *)              -     -

-     -     -     -     -     -     -     -     7(RR) data  distance
Effusion films *)              -     -

15    12    18    8     110   31    0     1     4     0.02798           0.005818    0.001723 3

25    12    19    11    19    26    0     1     5     4     0.01634     0.005009    0.009491    0.0011

35    12    38    43    23    65    25    1     7     125   0.014559

36    12    39    46    29    66    0     1     5     4     0.02043     0.00536     0.009865    0.00115

40    12    41    50    30    68    0     1     5     4     0.03607     0.00594     0.009925    0.00115

41    12    42    52    37    69    0     1     5     4     0.02734     0.00583     0.009926    0.00115

62    15    18    54    109   31    0     2     4     0.03757           0.005818    0.001723

64    15    19    56    44    26    0     2     5     4     0.02689     0.005417    0.0086      0.001

72    15    38    60    48    65    64    2     7     125   0.017345

73    15    39    62    45    66    72    2     5     4     0.02221     0.011011    0.00921     0.00105

75    15    41    66    46    68    0     2     5     4     0.02688     0.005819    0.00912     0.00105

76    15    42    68    47    69    0     2     5     4     0.02842     0.007343    0.0091      0.00105

0

-------------------- If the DT calculation is invoked the following data must be specified ----------------------

Num_reg    Num_theta    Num_phi    Maxpts1    Maxpts2    Maxpts3    Maxpts4    Maxpts5

   1          8            16         22          -          -          -          -

Tempin     Emissin      Tempout    Emissout      (specify for each flow region, starting with
1st, default = 0)

  0           0          1000         0

Geometry file:    TempFlag   Gas_temp file:    Abs/SootFlag    Abs_Coef/Soot file:    GasAbsProp   SourceCalc

geomtl71.dat        2       trtem7av.dat          4             trsoi7av.dat           0            0

CD ------------------------------ Conductive Heat Transfer (Type 24) ----------------------------

Elem  Class N1    N2    segment     material    geometry flag

-     -     -     -     thickness   flag        1   (use computed surface areas)

-     -     -     -       [m]        (1-7)      2           surf_area[m^2]

77    13    8     10    0.0015      1           1

78    13    11    12    0.00105     1           1

80    13    43    45    0.00105     1           1

81    13    46    47    0.00105     1           1
```

```
83    13    50    51    0.00105     1           1
84    13    52    53    0.00105     1           1
85    16    54    55    0.0015      1           1
86    16    56    57    0.00105     1           1
88    16    60    61    0.00105     1           1
89    16    62    63    0.00105     1           1
91    16    66    67    0.00105     1           1
92    16    68    69    0.00105     1           1
0
AHT ------------------------------ Annulus Heat Transfer (Type 25) ------------------------------
Elem  Class N1    N2    annulus Out/  ConvectFlag           RadFlag
-     -     -     -     elem    Inn 1     WallHeat[W]       1     RadHeat
-     -     -     -     0-Def 0-Def 2(L)                    2(L)    [W]
-     -     -     -     -     1-Out 3(RR) xdist[m] (def=0)  3(RR) -
-     -     -     -     -     2-Inn 4(Pt) xdist[m] (def=0)  -     -
93    14    10    15    21    2     3     0.01399     3
94    14    12    16    22    2     3     0.00817     3
96    14    45    23    70    2     3     0.004829    3
97    14    47    24    49    2     3     0.010215    3
99    14    51    26    51    2     3     0.018035    3
100   14    53    27    52    2     3     0.01367     3
101   17    55    29    53    1     3     0.018785    3
102   17    57    30    54    1     3     0.013445    3
104   17    61    32    56    1     3     0.00911     3
105   17    63    33    57    1     3     0.011105    3
107   17    67    36    60    1     3     0.02146     3
108   17    69    37    61    1     3     0.01421     3
0
PED ------------------------------ Pedestals and pin fins (Type 27) ------------------------------
Elem  Class N1    N2    x-area  perim  height  maxpwdth  width  stpitch  trpitch  radfil  theta  ns  nx  flag
-     -     -     -      [m^2]   [m]     [m]      [m]     [m]     [m]      [m]      [m]   [deg]  -   -   0
stag
-     -     -     -      -       -       -        -       -       -        -        -     -      -   -   1
in-line
0
============================== MOMENTUM ADDITION DATA ==========================================
Elem  Inject  E1  Theta1  E2  Theta2 ...  {Up to 20 pairs}
-     0/1     -   [rad]   -   [rad]      {End list with 0}
26    0     19    0       44    0       0
65    1     23    0       48    0       0
66    0     29    0       45    0       0
```

```
67    1     42    0       38    0       0
68    0     30    0       46    0       0
69    0     37    0       47    0       0
0
============================== RADIAL PRESSURE GRADIENT DATA ==================================
Node  EM1   EM2    R          H           E1 E2 E3 ...  {List up to 20 elements}
-     -     -     [m]        [m]          -  -  -      {End list with 0}
0
============================== FIXED ELEMENT FLOW DATA ========================================
Elem  Flow[kg/s]   FixFlow(y/n)
6     19.268      y
7     0.756       y
10    0.964       y
0
============================== CHECK VALVE DATA ===============================================
Elem  CheckValve(y/n)
0
============================== FIXED NODE PRESSURE DATA =======================================
Node  Pressure[Pa]
1     1.0185E+06
0
============================== FIXED NODE MASS SOURCE DATA ====================================
Node  Massource[kg/s] {inflow positive}
4     -0.658
7     0.503
0
============================== NODE HEIGHT DATA ===============================================
Node  Height[m]
0
============================== FIXED NODE TEMPERATURE DATA ====================================
Node  Temperature[K]
1     677.457
0
============================== SCALAR BOUNDARY CONDITIONS (for combustion model) ===============
Node  Scalar(1,2) Concentration
1     1           1
0
============================== EQUILIBRIUM CALCULATION DATA (for combustion model) =============
Fuel data file      Chemical data file   GP data directory
./data/fuel.dat     ./data/chem.dat      ./GPdata/gpandytest.dat
Class gp-   load-       spec_ineff[%] fuel_temp  pz-length   flairflow   stfar       combustor-volume
-     elem  flag(1-5)   (if flag = 1) -             [m]         [Kg/s]     -                 [m^3]
7     7     5                        288         0.0852      18.769      0.0682      0.02747
```

5.3 The result file of the "generic combustor 01"

```
RESULTS OF NETWORK FILE : networks/ network2.net

-------------------------------------

CLASS          **** NAME ****

   0           Generic

   1           Diffuser section

   2           Baseplate

   3           Outer flametube

   4           Outer annulus

   5           Inner flametube

   6           Inner annulus

   7           Combustion zone

   8           NGV

   9           Inner annulus liner cooling slots

  10           Baseplate cooling slots

  11           HP Turbine Bleed

  12           Outer FHT

  13           Outer CD

  14           Outer AHT

  15           Inner FHT

  16           Inner CD

  17           Inner AHT

  18           **** No Description ****

  19           **** No Description ****

  20           **** No Description ****

========================== ELEMENT_RESULTS ==========================================

LEGEND:

1     = Refers to side of Node N1

2     = Refers to side of Node N2

0     = Refers to total conditions

s     = Refers to static conditions

N     = Node

P     = Pressure

T     = Temperature

M     = Mach number

VEL   = Velocity

AREA  = Effective area i.e.  (area per element)x(number)x(discharge coefficient)

MFLOW = Mass flow

-------------------------------------------------------------------------------------
```

ELEM NO	CLASS NO	N1 NO	N2 NO	P01 [kPa]	P02 [kPa]	Ps1 [kPa]	Ps2 [kPa]	T01 [K]	T02 [K]	Ts1 [K]	Ts2 [K]
1	1	1	2	1018.50	1018.42	970.20	970.12	677.46	677.46	668.63	668.63
2	1	2	3	1018.42	1018.13	970.12	991.42	677.46	677.46	668.63	672.61
3	3	27	42	996.48	963.13	951.18	916.07	685.56	685.56	677.04	676.39
4	1	3	6	1018.13	996.70	1014.52	993.01	677.46	677.46	676.81	676.78
5	1	3	13	1018.13	1012.79	1017.14	1011.79	677.46	677.46	677.28	677.28
6	7	42	28	963.13	963.98	950.89	942.82	1469.03	1391.62	1464.68	1384.47
7	8	27	5	996.48	996.48	996.45	996.46	685.56	685.56	685.56	685.56
8	2	13	70	1012.79	966.36	966.36	917.43	677.46	677.46	668.94	668.04
9	2	13	71	1012.79	966.96	966.96	918.70	677.46	677.46	669.06	668.17
10	8	44	9	998.50	998.50	998.44	998.46	683.70	683.70	683.69	683.69
11	2	13	70	1012.79	966.36	966.36	917.43	677.46	677.46	668.94	668.04
12	2	13	71	1012.79	966.96	966.96	918.70	677.46	677.46	669.06	668.17
13	2	13	7	1012.79	962.30	926.14	869.95	677.46	677.46	661.33	659.29
14	11	29	4	998.57	988.96	988.96	979.25	677.79	677.79	676.02	675.99
15	12	18	8	961.97	961.97	961.97	961.97	1669.41	701.70	1669.41	701.70
16	2	13	70	1012.79	966.36	966.36	917.43	677.46	677.46	668.94	668.04
17	2	13	7	1012.79	962.30	926.14	869.95	677.46	677.46	661.33	659.29
18	1	3	14	1018.13	998.63	1012.10	992.48	677.46	677.46	676.37	676.33
19	3	15	18	996.66	961.97	961.51	925.45	677.68	677.68	671.16	670.65
20	3	16	19	996.60	962.40	961.83	926.29	678.73	678.73	672.27	671.78
21	4	6	15	996.70	996.66	993.01	993.26	677.46	677.46	676.78	676.83
22	4	15	16	996.66	996.60	991.08	989.94	677.68	677.68	676.66	676.46
23	3	16	19	996.60	962.40	961.83	926.29	678.73	678.73	672.27	671.78
25	12	19	11	962.40	962.40	962.40	962.40	1301.43	785.79	1301.43	785.79
26	7	18	19	961.97	962.40	961.43	961.83	1669.41	1754.89	1669.20	1754.66
28	2	13	7	1012.79	962.30	926.14	869.95	677.46	677.46	661.33	659.29
29	3	23	38	996.55	962.15	960.48	924.68	679.99	679.99	673.27	672.75
30	3	25	40	996.49	962.36	959.87	924.31	681.84	681.84	675.00	674.47
31	7	7	18	962.30	961.97	962.09	961.51	648.40	1761.58	648.37	1761.40
33	2	13	71	1012.79	966.96	966.96	918.70	677.46	677.46	669.06	668.17
34	2	13	7	1012.79	962.30	926.14	869.95	677.46	677.46	661.33	659.29
35	12	38	43	962.15	962.15	962.15	962.15	1549.45	720.74	1549.45	720.74
36	12	39	46	962.46	962.46	962.46	962.46	1529.04	812.64	1529.04	812.64
37	3	26	41	996.48	962.81	958.33	923.19	683.75	683.75	676.60	676.06
38	3	24	39	996.52	962.46	960.43	924.98	681.84	681.84	675.10	674.58
40	12	41	50	962.81	962.81	962.81	962.81	1459.77	803.87	1459.77	803.87
41	12	42	52	963.13	963.13	963.13	963.13	1469.03	808.51	1469.03	808.51
42	5	33	39	998.52	962.46	960.43	922.80	680.93	680.93	673.83	673.25

43	5	30	19	998.53	962.40	961.83	924.19	678.68	678.68	671.86	671.31
44	5	29	18	998.57	961.97	961.51	923.37	677.79	677.79	670.92	670.35
45	5	32	38	998.53	962.15	960.48	922.54	679.39	679.39	672.32	671.74
46	5	34	40	998.51	962.36	959.87	922.12	680.93	680.93	673.73	673.14
47	5	36	41	998.50	962.81	958.33	921.00	681.72	681.72	674.23	673.62
48	5	30	19	998.53	962.40	961.83	924.19	678.68	678.68	671.86	671.31
49	4	23	24	996.55	996.52	994.98	994.17	679.99	679.99	679.70	679.56
50	4	24	25	996.52	996.49	994.64	995.53	681.84	681.84	681.49	681.66
51	4	25	26	996.49	996.48	995.96	996.16	681.84	681.84	681.74	681.78
52	4	26	27	996.48	996.48	996.36	996.42	683.75	683.75	683.72	683.73
53	6	14	29	998.63	998.57	990.86	991.44	677.46	677.46	676.03	676.15
54	6	29	30	998.57	998.53	993.13	991.69	677.79	677.79	676.79	676.53
56	6	30	32	998.53	998.53	997.84	997.72	678.68	678.68	678.55	678.53
57	6	32	33	998.53	998.52	997.89	997.62	679.39	679.39	679.27	679.22
58	6	33	34	998.52	998.51	997.83	997.79	680.93	680.93	680.80	680.79
59	6	34	35	998.51	998.51	998.04	997.96	680.93	680.93	680.84	680.83
60	6	35	36	998.51	998.50	997.96	998.13	680.93	680.93	680.83	680.86
61	6	36	37	998.50	998.50	998.31	998.36	681.72	681.72	681.69	681.70
62	15	18	54	961.97	961.97	961.97	961.97	1669.41	715.28	1669.41	715.28
63	6	37	44	998.50	998.50	998.36	998.44	683.70	683.70	683.67	683.69
64	15	19	56	962.40	962.40	962.40	962.40	1301.43	785.22	1301.43	785.22
65	7	19	38	962.40	962.15	960.88	960.48	1301.43	1582.81	1300.97	1582.19
66	7	38	39	962.15	962.46	960.37	960.43	1549.45	1550.98	1548.80	1550.24
67	7	39	40	962.46	962.36	960.33	959.87	1529.04	1556.58	1528.27	1555.67
68	7	40	41	962.36	962.81	959.67	958.33	1519.50	1496.14	1518.54	1494.55
69	7	41	42	962.81	963.13	957.94	951.18	1459.77	1479.32	1458.08	1475.10
70	4	16	23	996.60	996.55	994.92	994.30	678.73	678.73	678.42	678.32
72	15	38	60	962.15	962.15	962.15	962.15	1549.45	723.46	1549.45	723.46
73	15	39	62	962.46	962.46	962.46	962.46	1529.04	803.66	1529.04	803.66
75	15	41	66	962.81	962.81	962.81	962.81	1459.77	812.20	1459.77	812.20
76	15	42	68	963.13	963.13	963.13	963.13	1469.03	813.49	1469.03	813.49
77	13	8	10	961.97	996.66	961.97	996.66	701.70	699.61	701.70	699.61
78	13	11	12	962.40	996.60	962.40	996.60	785.79	775.99	785.79	775.99
80	13	43	45	962.15	996.55	962.15	996.55	720.74	717.28	720.74	717.28
81	13	46	47	962.46	996.52	962.46	996.52	812.64	807.02	812.64	807.02
83	13	50	51	962.81	996.48	962.81	996.48	803.87	801.64	803.87	801.64
84	13	52	53	963.13	996.48	963.13	996.48	808.51	806.72	808.51	806.72
85	16	54	55	961.97	998.57	961.97	998.57	715.28	710.52	715.28	710.52
86	16	56	57	962.40	998.53	962.40	998.53	785.22	774.75	785.22	774.75
88	16	60	61	962.15	998.53	962.15	998.53	723.46	721.27	723.46	721.27

89	16	62	63	962.46	998.52	962.46	998.52	803.66	797.66	803.66	797.66
91	16	66	67	962.81	998.50	962.81	998.50	812.20	809.75	812.20	809.75
92	16	68	69	963.13	998.50	963.13	998.50	813.49	811.20	813.49	811.20
93	14	10	15	996.66	996.66	996.66	996.66	699.61	677.68	699.61	677.68
94	14	12	16	996.60	996.60	996.60	996.60	775.99	678.73	775.99	678.73
96	14	45	23	996.55	996.55	996.55	996.55	717.28	679.99	717.28	679.99
97	14	47	24	996.52	996.52	996.52	996.52	807.02	681.84	807.02	681.84
99	14	51	26	996.48	996.48	996.48	996.48	801.64	683.75	801.64	683.75
100	14	53	27	996.48	996.48	996.48	996.48	806.72	685.56	806.72	685.56
101	17	55	29	998.57	998.57	998.57	998.57	710.52	677.79	710.52	677.79
102	17	57	30	998.53	998.53	998.53	998.53	774.75	678.68	774.75	678.68
104	17	61	32	998.53	998.53	998.53	998.53	721.27	679.39	721.27	679.39
105	17	63	33	998.52	998.52	998.52	998.52	797.66	680.93	797.66	680.93
107	17	67	36	998.50	998.50	998.50	998.50	809.75	681.72	809.75	681.72
108	17	69	37	998.50	998.50	998.50	998.50	811.20	683.70	811.20	683.70
109	7	71	7	966.96	962.30	919.65	914.13	677.46	685.56	668.37	676.15
110	7	70	7	966.36	962.30	930.81	926.14	677.46	685.56	670.66	678.53

ELEM NO	CLASS NO	N1 NO	N2 NO	MFLOW [kg/s]	VEL1 [m/s]	VEL2 [m/s]	VELi [m/s]	M1 []	M2 []	AREA1 [m^2]	AREA2 [m^2]
1	1	1	2	21.14	137.28	137.30	137.29	0.2672	0.2672	0.0305900	0.0305900
2	1	2	3	21.14	137.30	101.70	116.77	0.2672	0.1974	0.0305900	0.0406500
3	3	27	42	0.29	135.18	140.22	135.18	0.2616	0.2715	0.0004427	0.0004427
4	1	3	6	6.52	37.21	38.01	0.00	0.0720	0.0736	0.0336700	0.0336700
5	1	3	13	7.01	19.49	19.59	0.00	0.0377	0.0379	0.0690400	0.0690400
6	7	42	28	19.27	104.00	133.24	116.39	0.1403	0.1848	0.0822500	0.0612000
7	8	27	5	0.76	3.68	3.22	3.43	0.0071	0.0062	0.0407300	0.0465900
8	2	13	70	0.34	134.94	141.94	134.94	0.2627	0.2765	0.0005016	0.0005016
9	2	13	71	0.19	134.05	140.91	134.05	0.2609	0.2745	0.0002891	0.0002891
10	8	44	9	0.96	4.75	3.95	4.31	0.0091	0.0076	0.0400800	0.0481100
11	2	13	70	0.40	134.94	141.94	134.94	0.2627	0.2765	0.0005880	0.0005880
12	2	13	71	0.66	134.05	140.91	134.05	0.2609	0.2745	0.0009797	0.0009797
13	2	13	7	3.20	185.76	197.15	185.76	0.3637	0.3866	0.0035492	0.0035492
14	11	29	4	0.66	61.43	62.04	61.43	0.1190	0.1201	0.0021101	0.0021101
15	12	18	8	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
16	2	13	70	0.64	134.94	141.94	134.94	0.2627	0.2765	0.0009467	0.0009467
17	2	13	7	0.25	185.76	197.15	185.76	0.3637	0.3866	0.0002717	0.0002717
18	1	3	14	7.61	48.12	49.07	0.00	0.0931	0.0950	0.0304800	0.0304800
19	3	15	18	0.55	118.14	122.65	118.14	0.2296	0.2384	0.0009442	0.0009442
20	3	16	19	2.85	117.58	122.01	117.58	0.2283	0.2370	0.0048794	0.0048794

21	4	6	15	6.52	38.01	36.48	37.23	0.0736	0.0706	0.0336700	0.0350800
22	4	15	16	5.96	46.81	51.14	48.88	0.0906	0.0990	0.0250600	0.0229600
23	3	16	19	0.12	117.58	122.01	117.58	0.2283	0.2370	0.0001980	0.0001980
25	12	19	11	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
26	7	18	19	8.37	23.36	24.45	23.91	0.0298	0.0304	0.1793000	0.1800000
28	2	13	7	0.80	185.76	197.15	185.76	0.3637	0.3866	0.0008913	0.0008913
29	3	23	38	0.49	119.91	124.46	119.91	0.2327	0.2416	0.0008299	0.0008299
30	3	25	40	0.57	121.01	125.56	121.01	0.2345	0.2435	0.0009601	0.0009601
31	7	7	18	7.52	9.11	22.13	15.98	0.0186	0.0275	0.1603000	0.1793000
33	2	13	71	0.19	134.05	140.91	134.05	0.2609	0.2745	0.0002785	0.0002785
34	2	13	7	0.34	185.76	197.15	185.76	0.3637	0.3866	0.0003781	0.0003781
35	12	38	43	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
36	12	39	46	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
37	3	26	41	0.62	123.72	128.33	123.72	0.2395	0.2485	0.0010141	0.0010141
38	3	24	39	0.27	120.11	124.61	120.11	0.2328	0.2416	0.0004512	0.0004512
40	12	41	50	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
41	12	42	52	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
42	5	33	39	0.24	123.24	128.15	123.24	0.2391	0.2487	0.0003889	0.0003889
43	5	30	19	4.48	120.74	125.55	120.74	0.2345	0.2440	0.0074760	0.0074760
44	5	29	18	0.30	121.25	126.15	121.25	0.2357	0.2453	0.0005003	0.0005003
45	5	32	38	0.23	123.01	127.96	123.01	0.2389	0.2486	0.0003778	0.0003778
46	5	34	40	0.33	124.13	129.10	124.13	0.2408	0.2506	0.0005334	0.0005334
47	5	36	41	0.37	126.68	131.70	126.68	0.2457	0.2555	0.0005921	0.0005921
48	5	30	19	0.04	120.74	125.55	120.74	0.2345	0.2440	0.0000722	0.0000722
49	4	23	24	2.51	24.91	30.37	27.37	0.0481	0.0587	0.0198000	0.0162500
50	4	24	25	2.24	27.19	19.52	22.73	0.0524	0.0377	0.0162500	0.0226200
51	4	25	26	1.66	14.52	11.26	12.68	0.0280	0.0217	0.0226200	0.0291600
52	4	26	27	1.05	7.11	5.09	5.93	0.0137	0.0098	0.0291600	0.0407300
53	6	14	29	7.61	55.17	52.87	54.00	0.1068	0.1024	0.0271400	0.0283100
54	6	29	30	6.66	46.17	51.80	48.83	0.0894	0.1003	0.0283100	0.0252600
56	6	30	32	2.13	16.52	17.76	17.12	0.0319	0.0343	0.0252600	0.0234900
57	6	32	33	1.90	15.85	18.75	17.18	0.0306	0.0362	0.0234900	0.0198700
58	6	33	34	1.66	16.44	16.90	16.67	0.0317	0.0326	0.0198700	0.0193300
59	6	34	35	1.33	13.57	14.58	14.06	0.0262	0.0281	0.0193300	0.0179900
60	6	35	36	1.33	14.58	12.01	13.17	0.0281	0.0232	0.0179900	0.0218400
61	6	36	37	0.96	8.69	7.27	7.91	0.0168	0.0140	0.0218400	0.0261000
62	15	18	54	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
63	6	37	44	0.96	7.29	4.75	5.75	0.0140	0.0091	0.0261000	0.0400800
64	15	19	56	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
65	7	19	38	15.86	34.39	39.80	37.16	0.0494	0.0519	0.1800000	0.1892000

66	7	38	39	16.58	40.74	43.49	42.07	0.0536	0.0572	0.1892000	0.1774000
67	7	39	40	17.09	44.18	48.14	46.10	0.0586	0.0633	0.1774000	0.1658000
68	7	40	41	17.99	49.48	63.42	55.53	0.0657	0.0849	0.1658000	0.1275000
69	7	41	42	18.98	65.29	103.12	80.05	0.0884	0.1389	0.1275000	0.0822500
70	4	16	23	3.00	25.66	29.77	27.56	0.0496	0.0575	0.0229600	0.0198000
72	15	38	60	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
73	15	39	62	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
75	15	41	66	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
76	15	42	68	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
77	13	8	10	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
78	13	11	12	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
80	13	43	45	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
81	13	46	47	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
83	13	50	51	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
84	13	52	53	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
85	16	54	55	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
86	16	56	57	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
88	16	60	61	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
89	16	62	63	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
91	16	66	67	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
92	16	68	69	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
93	14	10	15	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
94	14	12	16	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
96	14	45	23	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
97	14	47	24	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
99	14	51	26	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
100	14	53	27	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
101	17	55	29	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
102	17	57	30	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
104	17	61	32	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
105	17	63	33	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
107	17	67	36	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
108	17	69	37	0.00	0.00	0.00	0.00	0.0000	0.0000	0.0000000	0.0000000
109	7	71	7	1.04	139.48	141.96	140.72	0.2717	0.2749	0.0015620	0.0015620
110	7	70	7	1.38	120.67	122.70	121.69	0.2346	0.2372	0.0023701	0.0023701

========== NODE_RESULTS ==========

NODE	P0	T0	SCALAR[1]	MASSOURCE	MASSERR	Flametube_heat_loss
NO	[kPa]	[K]		[kg/s]	[kg/s]	[kW]
1	1018.500	677.46	1.00000	21.1430	0.000000e+00	0.0000

2	1018.417	677.46	1.00000	0.0000	-1.232483e-09	0.0000
3	1018.128	677.46	1.00000	0.0000	-8.252659e-08	0.0000
4	988.957	677.79	0.00000	-0.6580	3.896583e-10	0.0000
5	996.482	685.56	1.00000	-0.7560	0.000000e+00	0.0000
6	996.696	677.46	1.00000	-0.0000	0.000000e+00	0.0000
7	962.300	648.40	0.96612	0.5030	-8.562195e-08	0.0000
8	961.973	701.70	0.00000	0.0000	0.000000e+00	0.0000
9	998.497	683.70	1.00000	-0.9640	0.000000e+00	0.0000
10	996.656	699.61	0.00000	0.0000	0.000000e+00	0.0000
11	962.399	785.79	0.00000	0.0000	0.000000e+00	0.0000
12	996.597	775.99	0.00000	0.0000	0.000000e+00	0.0000
13	1012.789	677.46	1.00000	0.0000	0.000000e+00	0.0000
14	998.626	677.46	1.00000	-0.0000	0.000000e+00	0.0000
15	996.656	677.68	1.00000	0.0000	-1.964090e-09	0.0000
16	996.597	678.73	1.00000	0.0000	-1.068430e-09	0.0000
18	961.973	1669.41	0.96639	0.0000	-4.467246e-09	0.0000
19	962.399	1301.43	0.97296	0.0000	-8.674604e-08	0.0000
23	996.553	679.99	1.00000	0.0000	-5.857697e-10	0.0000
24	996.516	681.84	1.00000	0.0000	-5.514841e-10	0.0000
25	996.495	681.84	1.00000	0.0000	-1.708637e-10	0.0000
26	996.484	683.75	1.00000	0.0000	-9.376511e-10	0.0000
27	996.483	685.56	1.00000	0.0000	2.158945e-10	0.0000
28	963.976	1391.62	0.98034	-19.2680	0.000000e+00	0.0000
29	998.569	677.79	1.00000	0.0000	-3.344067e-09	0.0000
30	998.534	678.68	1.00000	0.0000	-1.901100e-09	0.0000
32	998.526	679.39	1.00000	0.0000	-2.419468e-10	0.0000
33	998.520	680.93	1.00000	0.0000	-3.718483e-10	0.0000
34	998.513	680.93	1.00000	0.0000	-6.505763e-11	0.0000
35	998.505	680.93	1.00000	0.0000	5.714140e-11	0.0000
36	998.501	681.72	1.00000	0.0000	-2.930554e-10	0.0000
37	998.499	683.70	1.00000	0.0000	-2.674073e-10	0.0000
38	962.154	1549.45	0.97503	0.0000	-1.611504e-08	0.0000
39	962.462	1529.04	0.97503	0.0000	-5.008132e-08	0.0000
40	962.356	1519.50	0.97695	0.0000	-1.186553e-08	0.0000
41	962.811	1459.77	0.97761	0.0000	3.242723e-08	0.0000
42	963.131	1469.03	0.98034	0.0000	2.579003e-07	0.0000
43	962.154	720.74	0.00000	0.0000	0.000000e+00	0.0000
44	998.498	683.70	1.00000	0.0000	7.803127e-10	0.0000
45	996.553	717.28	0.00000	0.0000	0.000000e+00	0.0000
46	962.462	812.64	0.00000	0.0000	0.000000e+00	0.0000

47	996.516	807.02	0.00000	0.0000	0.000000e+00	0.0000
50	962.811	803.87	0.00000	0.0000	0.000000e+00	0.0000
51	996.484	801.64	0.00000	0.0000	0.000000e+00	0.0000
52	963.131	808.51	0.00000	0.0000	0.000000e+00	0.0000
53	996.483	806.72	0.00000	0.0000	0.000000e+00	0.0000
54	961.973	715.28	0.00000	0.0000	0.000000e+00	0.0000
55	998.569	710.52	0.00000	0.0000	0.000000e+00	0.0000
56	962.399	785.22	0.00000	0.0000	0.000000e+00	0.0000
57	998.534	774.75	0.00000	0.0000	0.000000e+00	0.0000
60	962.154	723.46	0.00000	0.0000	0.000000e+00	0.0000
61	998.526	721.27	0.00000	0.0000	0.000000e+00	0.0000
62	962.462	803.66	0.00000	0.0000	0.000000e+00	0.0000
63	998.520	797.66	0.00000	0.0000	0.000000e+00	0.0000
66	962.811	812.20	0.00000	0.0000	0.000000e+00	0.0000
67	998.501	809.75	0.00000	0.0000	0.000000e+00	0.0000
68	963.131	813.49	0.00000	0.0000	0.000000e+00	0.0000
69	998.499	811.20	0.00000	0.0000	0.000000e+00	0.0000
70	966.362	677.46	1.00000	0.0000	-1.407131e-08	0.0000
71	966.959	677.46	1.00000	0.0000	-2.756186e-08	0.0000

========== SUMMARY_TABLE_1 ==========

ELEM NO	MFLOW [kg/s]	MIXEDFLOW [kg/s]	PRESS_ELEM [kPa]	DELTA_P0 [kPa]	PRESS_ERR [kPa]
Diffuser section					
1	21.14300	21.14300	9.70160e+02	8.31752e-02	1.91738e-10
2	21.14300	21.14300	9.83169e+02	2.88654e-01	6.93772e-10
5	7.01210	7.01210	1.01714e+03	5.33868e+00	-1.22064e-08
18	7.61499	7.61499	1.01210e+03	1.95024e+01	-4.68558e-08
4	6.51591	6.51591	1.01452e+03	2.14324e+01	-5.14253e-08
Baseplate					
34	0.34124	0.34124	1.01279e+03	5.04893e+01	-3.51469e-09
11	0.39772	0.39772	1.01279e+03	4.64278e+01	8.41916e-07
8	0.33925	0.33925	1.01279e+03	4.64278e+01	8.41916e-07
16	0.64032	0.64032	1.01279e+03	4.64278e+01	8.41916e-07
12	0.65860	0.65860	1.01279e+03	4.58302e+01	3.56330e-06
17	0.24525	0.24525	1.01279e+03	5.04893e+01	-3.51469e-09
33	0.18721	0.18721	1.01279e+03	4.58302e+01	3.56330e-06
13	3.20362	3.20362	1.01279e+03	5.04893e+01	-3.51469e-09
28	0.80453	0.80453	1.01279e+03	5.04893e+01	-3.51469e-09
9	0.19436	0.19436	1.01279e+03	4.58302e+01	3.56330e-06

Outer flametube					
20	2.84813	2.84813	9.96597e+02	3.41981e+01	-1.18717e-07
38	0.26751	0.26751	9.96516e+02	3.40541e+01	-1.03743e-07
23	0.11559	0.11559	9.96597e+02	3.41981e+01	-1.18717e-07
19	0.55449	0.55449	9.96656e+02	3.46835e+01	-1.27245e-07
29	0.49259	0.49259	9.96553e+02	3.43992e+01	-1.23151e-07
30	0.57325	0.57325	9.96495e+02	3.41386e+01	-9.28548e-08
37	0.61662	0.61662	9.96484e+02	3.36728e+01	-5.62696e-08
3	0.29174	0.29174	9.96483e+02	3.33520e+01	2.13040e-09
Outer annulus					
21	6.51591	6.51591	9.93143e+02	3.97037e-02	-2.15254e-07
22	5.96142	5.96142	9.90545e+02	5.92407e-02	-2.85093e-10
70	2.99770	2.99770	9.94642e+02	4.33568e-02	-1.97638e-10
49	2.50511	2.50511	9.94632e+02	3.70593e-02	-1.57085e-10
50	2.23760	2.23760	9.95197e+02	2.17283e-02	-8.69827e-11
51	1.66436	1.66436	9.96082e+02	1.05657e-02	-3.64686e-11
52	1.04774	1.04774	9.96394e+02	1.42431e-03	-2.45062e-12
Inner flametube					
43	4.48344	4.48344	9.98534e+02	3.61352e+01	-1.07691e-07
42	0.23699	0.23699	9.98520e+02	3.60580e+01	-9.28870e-08
48	0.04331	0.04331	9.98534e+02	3.61352e+01	-1.07691e-07
44	0.30162	0.30162	9.98569e+02	3.65965e+01	-1.16147e-07
45	0.23036	0.23036	9.98526e+02	3.63721e+01	-1.12212e-07
46	0.32733	0.32733	9.98513e+02	3.61566e+01	-8.20433e-08
47	0.36993	0.36993	9.98501e+02	3.56900e+01	-4.54639e-08
Inner annulus					
53	7.61499	7.61499	9.91162e+02	5.68293e-02	-1.96184e-07
54	6.65536	6.65536	9.92472e+02	3.50565e-02	-1.21425e-10
56	2.12861	2.12861	9.97783e+02	7.62668e-03	-2.10103e-11
57	1.89826	1.89826	9.97771e+02	6.01165e-03	-1.48505e-11
58	1.66127	1.66127	9.97810e+02	7.55705e-03	-1.58976e-11
59	1.33393	1.33393	9.98007e+02	7.46598e-03	-1.09614e-11
60	1.33393	1.33393	9.98062e+02	4.00092e-03	-5.98359e-12
61	0.96400	0.96400	9.98341e+02	2.50773e-03	1.09099e-12
63	0.96400	0.96400	9.98415e+02	9.11956e-04	4.50263e-13
Combustion zone					
110	1.37728	5.04307	9.28480e+02	4.06144e+00	1.32225e-06
109	1.04017	4.70595	9.16896e+02	4.65905e+00	6.01088e-06
31	7.51510	11.18088	9.61783e+02	3.27702e-01	-3.49513e-10
26	8.37121	11.27006	9.61628e+02	-4.26180e-01	-1.53529e-08

65	15.86169	14.00959	9.60679e+02	2.44516e-01	-1.13288e-09
66	16.58463	15.16890	9.60403e+02	-3.08089e-01	-2.55839e-08
67	17.08913	15.16968	9.60112e+02	1.06197e-01	-2.06009e-08
68	17.98971	16.43232	9.59173e+02	-4.55194e-01	-8.30695e-08
69	18.97626	16.91868	9.55708e+02	-3.19420e-01	-6.12337e-08
6	19.26800	19.26800	9.47820e+02	-8.44904e-01	-1.11876e-09
NGV					
7	0.75600	0.75600	9.96453e+02	3.44980e-04	3.96341e-14
10	0.96400	0.96400	9.98451e+02	3.57212e-04	-1.80657e-14
HP Turbine Bleed					
14	0.65800	0.65800	9.98569e+02	9.61245e+00	1.17149e-07
Outer FHT					
15	0.00000	0.00000	9.61973e+02	0.00000e+00	0.00000e+00
25	0.00000	0.00000	9.62399e+02	0.00000e+00	0.00000e+00
35	0.00000	0.00000	9.62154e+02	0.00000e+00	0.00000e+00
36	0.00000	0.00000	9.62462e+02	0.00000e+00	0.00000e+00
40	0.00000	0.00000	9.62811e+02	0.00000e+00	0.00000e+00
41	0.00000	0.00000	9.63131e+02	0.00000e+00	0.00000e+00
Outer CD					
77	0.00000	0.00000	9.61973e+02	-3.46835e+01	-3.46835e+01
78	0.00000	0.00000	9.62399e+02	-3.41981e+01	-3.41981e+01
80	0.00000	0.00000	9.62154e+02	-3.43992e+01	-3.43992e+01
81	0.00000	0.00000	9.62462e+02	-3.40541e+01	-3.40541e+01
83	0.00000	0.00000	9.62811e+02	-3.36728e+01	-3.36728e+01
84	0.00000	0.00000	9.63131e+02	-3.33520e+01	-3.33520e+01
Outer AHT					
93	0.00000	0.00000	9.96656e+02	0.00000e+00	0.00000e+00
94	0.00000	0.00000	9.96597e+02	0.00000e+00	0.00000e+00
96	0.00000	0.00000	9.96553e+02	0.00000e+00	0.00000e+00
97	0.00000	0.00000	9.96516e+02	0.00000e+00	0.00000e+00
99	0.00000	0.00000	9.96484e+02	0.00000e+00	0.00000e+00
100	0.00000	0.00000	9.96483e+02	0.00000e+00	0.00000e+00
Inner FHT					
62	0.00000	0.00000	9.61973e+02	0.00000e+00	0.00000e+00
64	0.00000	0.00000	9.62399e+02	0.00000e+00	0.00000e+00
72	0.00000	0.00000	9.62154e+02	0.00000e+00	0.00000e+00
73	0.00000	0.00000	9.62462e+02	0.00000e+00	0.00000e+00
75	0.00000	0.00000	9.62811e+02	0.00000e+00	0.00000e+00
76	0.00000	0.00000	9.63131e+02	0.00000e+00	0.00000e+00
Inner CD					

85	0.00000	0.00000	9.61973e+02	-3.65965e+01	-3.65965e+01
86	0.00000	0.00000	9.62399e+02	-3.61352e+01	-3.61352e+01
88	0.00000	0.00000	9.62154e+02	-3.63721e+01	-3.63721e+01
89	0.00000	0.00000	9.62462e+02	-3.60580e+01	-3.60580e+01
91	0.00000	0.00000	9.62811e+02	-3.56900e+01	-3.56900e+01
92	0.00000	0.00000	9.63131e+02	-3.53681e+01	-3.53681e+01
Inner AHT					
101	0.00000	0.00000	9.98569e+02	0.00000e+00	0.00000e+00
102	0.00000	0.00000	9.98534e+02	0.00000e+00	0.00000e+00
104	0.00000	0.00000	9.98526e+02	0.00000e+00	0.00000e+00
105	0.00000	0.00000	9.98520e+02	0.00000e+00	0.00000e+00
107	0.00000	0.00000	9.98501e+02	0.00000e+00	0.00000e+00
108	0.00000	0.00000	9.98499e+02	0.00000e+00	0.00000e+00

========== DUCT_WITH_AREA_CHANGE_AXIAL_PATH_RESULTS ==========

ELEM	PATH	MIDPT	MFLOW	MIXEDFLOW	MACH NO.	P TOTAL	P STATIC	T TOTAL	T STATIC	F.A.R.	EFFIC
na	[m]	[m]	[kg/s]	[kg/s]		[kPa]	[kPa]	[K]	[K]		
Diffuser section											
1	0.0000	0.0033	21.143	21.143	0.26720	1.01850e+06	9.70203e+05	677.457	668.629	0.00000	0.000
2	0.0066	0.0251	21.143	21.143	0.26725	1.01842e+06	9.70115e+05	677.457	668.632	0.00000	0.000
Outer annulus											
21	0.0000	0.0140	6.516	6.516	0.07356	9.96696e+05	9.93013e+05	677.457	676.781	0.00000	0.000
22	0.0280	0.0362	5.961	5.961	0.09060	9.96656e+05	9.91077e+05	677.681	676.657	0.00000	0.000
70	0.0443	0.0589	2.998	2.998	0.04960	9.96597e+05	9.94921e+05	678.732	678.424	0.00000	0.000
49	0.0734	0.0837	2.505	2.505	0.04811	9.96553e+05	9.94977e+05	679.987	679.697	0.00000	0.000
50	0.0939	0.1030	2.238	2.238	0.05245	9.96516e+05	9.94643e+05	681.836	681.491	0.00000	0.000
51	0.1122	0.1303	1.664	1.664	0.02799	9.96495e+05	9.95961e+05	681.836	681.738	0.00000	0.000
52	0.1483	0.1620	1.048	1.048	0.01369	9.96484e+05	9.96356e+05	683.747	683.723	0.00000	0.000
Inner annulus											
53	0.0000	0.0188	7.615	7.615	0.10683	9.98626e+05	9.90864e+05	677.457	676.034	0.00000	0.000
54	0.0376	0.0510	6.655	6.655	0.08935	9.98569e+05	9.93131e+05	677.787	676.790	0.00000	0.000
56	0.0645	0.0818	2.129	2.129	0.03192	9.98534e+05	9.97838e+05	678.675	678.548	0.00000	0.000
57	0.0992	0.1103	1.898	1.898	0.03063	9.98526e+05	9.97886e+05	679.387	679.270	0.00000	0.000
58	0.1214	0.1332	1.661	1.661	0.03173	9.98520e+05	9.97833e+05	680.925	680.799	0.00000	0.000
59	0.1450	0.1585	1.334	1.334	0.02618	9.98513e+05	9.98045e+05	680.925	680.839	0.00000	0.000
60	0.1719	0.1799	1.334	1.334	0.02813	9.98505e+05	9.97965e+05	680.925	680.826	0.00000	0.000
61	0.1880	0.2022	0.964	0.964	0.01675	9.98501e+05	9.98310e+05	681.723	681.688	0.00000	0.000
63	0.2164	0.2289	0.964	0.964	0.01404	9.98499e+05	9.98364e+05	683.698	683.673	0.00000	0.000
Combustion zone											
110	0.0000	0.0066	1.377	5.043	0.23465	9.66362e+05	9.30814e+05	685.559	678.678	0.00000	96.400

109	0.0132	0.0198	1.040	4.706	0.27169	9.66959e+05	9.19654e+05	685.559	676.366	0.00000	96.40
31	0.0264	0.0474	7.515	11.181	0.01862	9.62300e+05	9.62087e+05	1761.584	1761.499	0.03507	93.60
26	0.0685	0.0766	8.371	11.270	0.02977	9.61973e+05	9.61427e+05	1754.886	1754.668	0.03478	93.67
65	0.0846	0.1005	15.862	14.010	0.04945	9.62399e+05	9.60883e+05	1582.806	1582.246	0.02779	95.21
66	0.1164	0.1241	16.585	15.169	0.05364	9.62154e+05	9.60367e+05	1550.980	1550.328	0.02561	98.62
67	0.1318	0.1318	17.089	15.170	0.05857	9.62462e+05	9.60332e+05	1556.583	1555.804	0.02561	99.28
68	0.1318	0.1497	17.990	16.432	0.06571	9.62356e+05	9.59669e+05	1496.138	1495.184	0.02360	99.28
69	0.1675	0.1799	18.976	16.919	0.08845	9.62811e+05	9.57943e+05	1479.318	1477.603	0.02290	99.82
6	0.1923	0.1995	19.268	19.268	0.14027	9.63131e+05	9.50889e+05	1391.621	1387.496	0.02005	99.94
NGV											
7	0.0000	0.0117	0.756	0.756	0.00708	9.96483e+05	9.96449e+05	685.562	685.556	0.00000	0.00
10	0.0233	0.0341	0.964	0.964	0.00914	9.98498e+05	9.98441e+05	683.698	683.688	0.00000	0.00