AALTO UNIVERSITY
SCHOOL OF ENGINEERING
Department of Applied Mechanics

Viljami Maakala

Multi-Objective Optimization of Recovery Boiler Dimensions Using Computational Fluid Dynamics

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Helsinki, April 26, 2013

Supervisor: Professor Timo Siikonen

Instructor: Pasi Miikkulainen, D.Sc. (Tech.)

AALTO UNIVERSITY
SCHOOL OF ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author:	Viljami Maakala
Title of the Thesis:	Multi-Objective Optimization of Recovery Boiler Dimensions Using Computational Fluid Dynamics
Department:	Department of Applied Mechanics
Professorship:	Kul-34 Aeronautical Engineering
Supervisor:	Professor Timo Siikonen
Instructor:	Pasi Miikkulainen, D.Sc. (Tech.)

The purpose of this work was to develop a multi-objective optimization program based on computational fluid dynamics (CFD). The program combines a CFD model with a genetic algorithm optimizer and a radial basis function network learner.

The tool was applied to optimizing a furnace geometry of a recovery boiler using two approaches: an uncoupled method and a coupled method. Before solving the optimization task, a study was done on the CFD model errors and the CFD-optimization program was verified.

Error analyses revealed that the simulations have substantial iteration errors. Discretization errors were studied using grid convergence index values, but iteration errors made their interpretation difficult. The program was verified in test problems and the proposed methodology was concluded to work as intended. Both optimization approaches found several geometries that deliver better performance than the original boiler design. The coupled method is more reliable, because it performs numerous CFD evaluations near the final solutions.

Future research should be done to reduce iteration errors when modeling recovery boilers. It would also be useful to develop the CFD-optimization methodology further and to use it in different applications. Areas for development include improved integration of preferences into the optimization and usage of hybrid optimization methods.

Date:	April 26, 2013	**Number of pages:**	92 + 6
Keywords:	Computational fluid dynamics, Error estimation, Genetic algorithm, Optimization, Radial basis function, Recovery boiler		

AALTO-YLIOPISTO
INSINÖÖRITIETEIDEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Viljami Maakala
Työn otsikko:	Soodakattilan dimensioiden monitavoiteoptimointi laskennallisen virtausmekaniikan avulla
Laitos:	Sovelletun mekaniikan laitos
Professuuri:	Kul-34 Lentotekniikka
Työn valvoja:	Professori Timo Siikonen
Työn ohjaaja:	Tekniikan tohtori Pasi Miikkulainen

Työn tarkoituksena oli kehittää laskennallista virtausmekaniikkaa (CFD) käyttävä monitavoiteoptimointiohjelma. Ohjelma yhdistää CFD-mallin geneettisen algorimiin pohjautuvaan optimoijaan sekä radiaalikantafunktioverkkoa käyttävään oppijaan.

Työkalua käytettiin soodakattilan tulipesägeometrian optimointiin kahdella lähestymistavalla: kytkemättömällä sekä kytketyllä menetelmällä. Ennen tehtävän ratkaisemista CFD-mallille suoritettiin virhetarkastelu ja CFD-optimointiohjelma verifioitiin.

Virhetarkastelussa havaittiin simuloinneissa merkittäviä iteraatiovirheitä. Diskretointivirheitä tutkittiin hilakonvergenssi-indeksin avulla, mutta iteraatiovirheet hankaloittivat tulosten tulkintaa. Ohjelma verifioitiin testiongelmissa, ja menetelmän havaittiin toimivan halutusti. Molemmat optimointilähestymistavat löysivät useita geometrioita, joilla kattilan suorituskyky paranee alkuperäiseen geometriaan verrattuna. Kytketty menetelmä on luotettavampi, koska se tekee useita CFD-laskentoja lopullisten ratkaisujen läheisyydessä.

Tutkimusta tulisi tehdä iteraatiovirheiden pienentämiseksi soodakattiloiden mallinnuksessa. Lisäksi olisi hyödyllistä kehittää CFD-optimointia eteenpäin ja käyttää sitä muissakin sovelluksissa. Kehittämisalueita ovat esimerkiksi preferenssien parempi integrointi optimointiin sekä hybridimenetelmien käyttö.

Päiväys:	26.4.2013	**Sivumäärä:**	92 + 6
Avainsanat:	Geneettinen algoritmi, Laskennallinen virtausmekaniikka, Optimointi, Radiaalikantafunktio, Soodakattila, Virhetarkastelu		

Acknowledgements

The work for this thesis was carried out between September 2012 and April 2013. I wish to thank my employer ANDRITZ for giving me the chance to do this project and for providing a good working environment.

I am grateful to my supervisor Professor Timo Siikonen and instructor D.Sc. Pasi Miikkulainen for all the support and guidance they gave me throughout my work. I especially wish to thank them for having the time to answer my questions and to review my work.

My gratitude also goes to D.Sc. Tor-Martin Tveit for lending me his expertise on multi-objective optimization, for reviewing my work and and for giving me valuable advice.

I would also like to thank M.Sc. Jukka Savolainen, M.Sc. Hu Kexin, Ph.D. Keijo Salmenoja, M.Sc. Jukka Röppänen, M.Sc. Miro Loschkin, M.Sc. Niko Metsämuuronen, M.Sc. Juha Kyttälä, B.Eng. (Mech. Eng.) Veli Riikonen, M.Sc. Lauri Pakarinen and M.Sc. Petri Pynnönen for all the help and professional advice I received from them. Finally, I would like to give special thanks to all the friendly people working at ANDRITZ Helsinki and Kotka offices.

Helsinki, April 26, 2013

Viljami Maakala

Contents

Nomenclature

Abbreviations

ANN	Artificial neural network
CFD	Computational fluid dynamics
CNCG	Concentrated non-condensible gas
DM	Decision maker
DNCG	Dilute non-condensible gas
DOE	Design of experiments
DPM	Discrete phase model
GA	Genetic algorithm
GCI	Grid convergence index
HDG	High density grid
IDG	Intermediate density grid
LDG	Low density grid
LES	Large eddy simulation
MLP	Multilayer perceptron
NSGA-II	Elitist non-dominated sorting genetic algorithm
POF	Pareto optimal front
RA	Running average
RANS	Reynolds-averaged Navier-Stokes
RBF	Radial basis function

Symbols

ϵ	Attenuation coefficient
ϵ	Relative error
$\mathcal{D}$	Design space
$\mathcal{S}$	Feasible space
$\mathcal{Z}$	Objective space
ϕ	Arbitrary variable
τ	Minimum level of dominance

$\vec{c}$	RBF neuron center
$\vec{o}$	Ordering vector
$\vec{x}$	Design point vector, solution vector, individual
A	Actual fractional error
a	Influence coefficient
b	Decision variable string length
b	Source and boundary condition term
c	Constraint function on the geometry
D	Dimensionality of a flow domain
d	Boiler depth
d	Crowding distance
d	Local database size
E	Estimated fractional error
E	Total input signal to a neuron
F	Fitness front
f	Activation function of a neuron
f	Discrete solution
f	Objective function
F_s	Factor of safety
g	Derivative related term
g	Inequality constraint function
h	Boiler nose height
h	Equality constraint function
h	Grid spacing
N	Neuron
N	Number of cells in a grid
n	Population size
P	Population
p	Order of a discretization method
p_c	Crossover probability
p_m	Mutation probability
Q	Offspring population
R	Combined parent and offspring populations
R	Residual
r	Fitness rank
r	Grid refinement ratio
S	Asymptotic range indicator
S	Asymptotic range indicator
S	Output signal of a neuron

s	Binary string
w	Boiler width
w	Weight of a connection between two neurons
w	Weight of an objective function
x	Design variable
y	Solution in the local database

Subscripts and Superscripts

$carry$	Carryover
CO	CO content
$COspi$	CO spikes
dv	Decision variable
eff	Effective
$exact$	Exact
ind	Individual
$lowT$	Lower furnace temperature
max	Maximum value, maximum bound
min	Minimum value, minimum bound
nb	Neighbor cell
$O2$	O_2 content
P	Inspected cell
$pWall$	Particles landing to walls
RBF	Radial basis function
T	Temperature
$T\sigma$	Temperature standard deviation
$yVel\sigma$	Upward velocity standard deviation

List of Figures

List of Tables

Chapter 1

Introduction

Recovery boilers are used for burning black liquor, which is a byproduct of the pulp making process, to produce steam for energy generation and to recover inorganic chemicals in the liquor. The capacities of the largest boilers currently built are approximately 7 000 tons of dry solids per day (tds/d) and even larger boilers have been planned. The furnace designs of these large boilers have not been optimized and are mostly based on heuristic rules or experience with smaller boilers. Because experimental research is often infeasible to use in the design process for its high cost, recovery boiler modeling based on computational fluid dynamics (CFD) has been extensively utilized in recent years.

The large capacity boilers are physically large (even 20 m wide, 20 m deep and 80 m high) and this causes difficulties in computational modeling. Simulations with these models are expensive because they require large amounts of resources, including, processor memory, computing power and computing time. It has also been observed that when large models are involved, the computations do not always converge within a reasonable period of time. This has raised interest in researching errors related to iterative convergence and also the density requirements of the grids used.

Because it is computationally expensive to model large boilers, studying different design choices using CFD modeling is challenging. The computational cost makes it unfeasible to do trial and error comparisons of different designs by running a large amount of simulations. Furthermore, an improved design is not guaranteed this way. Because of these reasons, there is a growing interest in CFD driven optimization, where optimization methods are used to intelligently search for an optimal design.

The basic idea of the CFD-optimization methodology is that a CFD model is com-

bined with an optimization method. The optimization method finds the optimum using information about some defined objective functions in different points of a design space and the CFD model is used to provide this information by simulations. Approximate evaluation schemes, also called meta-model schemes, using machine learning methods are often employed in addition to decrease the amount of needed CFD evaluations.

A multi-objective optimization program based on CFD is developed in this work. The program combines a recovery boiler CFD model with an optimization method based on a genetic algorithm (GA) and a meta-model based on a radial basis function (RBF) network. The developed program is used in a chosen application of optimizing a furnace geometry of a large capacity (7 000 tds/d) recovery boiler.

1.1 Recovery Boiler Process

In a kraft pulping process wood is converted into pulp by treatment with a solution called white liquor. In this process, organic matter is separated from the wood and the resulting solution is called weak black liquor, which can be concentrated to black liquor by rising its dry solids content. The purpose of a recovery boiler is to burn black liquor in such a way that its chemicals can be recovered and high pressure steam for energy generation can be produced efficiently. Vakkilainen (2005) notes that a modern recovery boiler has to fulfill these goals also in an environmentally friendly way. The general theory of recovery boilers is presented below according to Adams (1997).

Black liquor is sprayed into the boiler as coarse droplets and it is wanted that they land on the char bed, located on the furnace floor. Combustion of black liquor occurs in three main stages: drying, pyrolysis and char combustion. In the drying stage the moisture of in the black liquor evaporates and in the pyrolysis stage organic materials in the black liquor release gas-phase volatiles. The combustion of the solid char occurs mainly on the char bed. When the carbon is combusted the inorganic chemicals of the char form a mixture called smelt, which is recovered from the furnace through smelt spouts.

A modern recovery boiler with its most important features named is shown in Figure 1.1. The boiler consists of two main sections: a furnace section and a heat transfer section, which are separated by the nose. The purpose of the nose is to turn the flow of the combustion gas in such a way as to produce a uniform flow over the heat transfer surfaces. The mixing and combustion of fuel and air should be com-

pleted in the furnace section. The heat transfer surfaces usually present are boiler waterwalls, screen tubes, superheaters, boiler bank and economizers. The wall on which the nose is located is called the rear wall and opposite to it is the front wall. When looking from the front wall to the rear wall the right wall is on the right hand side and the left wall is on the left hand side. Boiler depth is the distance between the front and rear walls and boiler width is the distance between the side walls.

The combustion air system commonly consists of three levels: primary and secondary levels located below the liquor guns and a tertiary level located above the liquor guns. Typically, from 20% to 30% of the combustion air is introduced on the primary level, from 35% to 40% on the tertiary level and the rest on the secondary level.

The primary air is located approximately one meter above the furnace floor. The

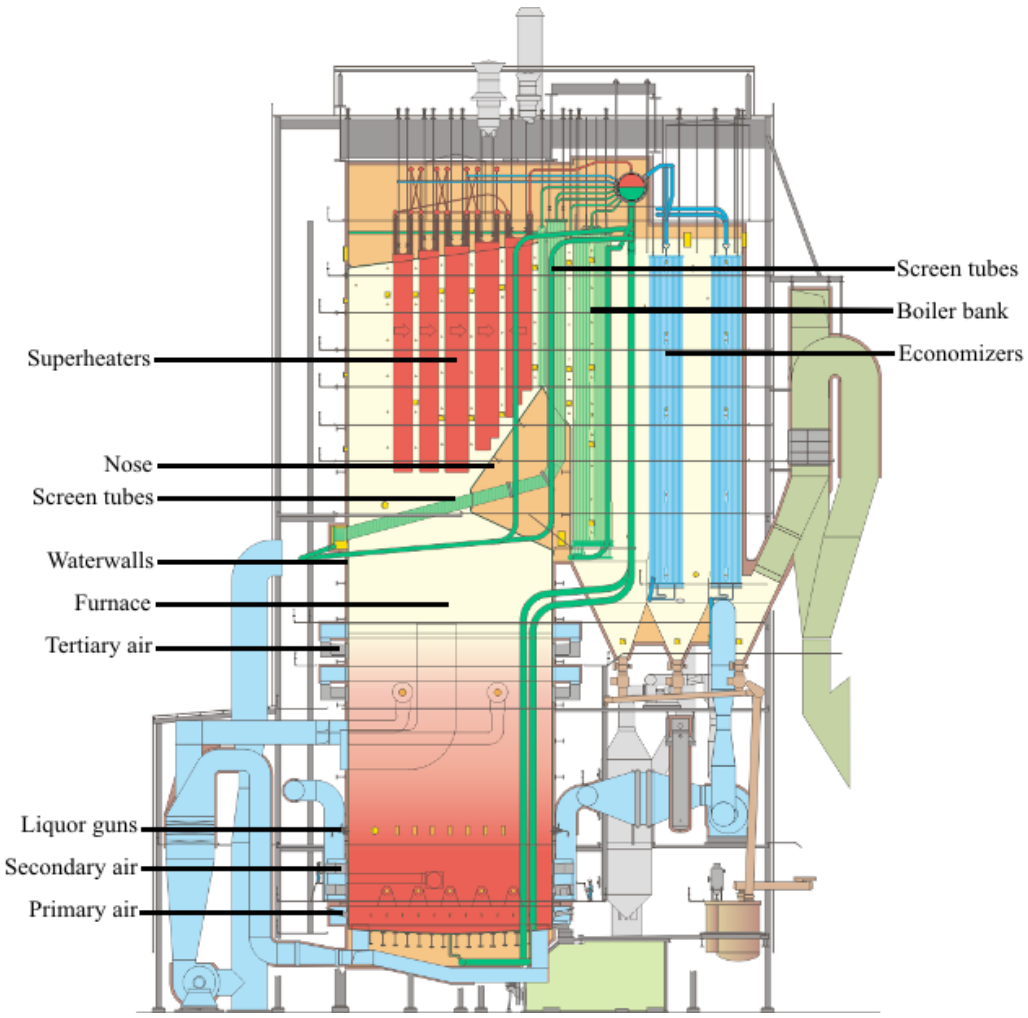


Figure 1.1: A modern recovery boiler (6 000 tds/d) with its most important features named. Courtesy of ANDRITZ Oy.

ports are the smallest of the air ports in the boiler and they are located on all of the walls. They are used to control the shape and position of the char bed and to provide air for char combustion. The ports on the secondary air level are larger than the primary air ports and there are fewer of them. They are located above the primary air level on the front and rear walls. The purpose of the secondary air is to burn the gases and CO released from the char bed while providing under-stoichiometric conditions for reactions involving O_2. This means that aggressive mixing is not wanted on the secondary air level. The tertiary air ports are the largest of the air ports and they are located on the front and rear walls. The purpose of the tertiary air level is to finalize the mixing of the combustion gases and to provide air to complete the combustion process. The location of the tertiary air depends on the manufacturer and design of the boiler.

1.2 Computational Fluid Dynamics in Recovery Boiler Modeling

Computational fluid dynamics (CFD) is a branch of fluid mechanics that uses computational methods for solving problems involving fluid flow and heat transfer. Using computational methods for solving problems is particularly attractive when experimentation is expensive or otherwise unfeasible. It is also invaluable when the problem involves comparing a large number of designs against each other. The general description of CFD is presented below according to Ferziger and Perić (2002) and Fletcher (1991).

Mathematical representation of a CFD problem involves a set of continuous governing equations along with their respective boundary and initial conditions. These can be, e.g., equations of continuity, motion, energy, species conservation or radiation. For some phenomena, such as turbulence and combustion, exact equations are not known or their numerical solution is not feasible. This means that usage of models is a necessity, which introduces modeling error in the solutions.

The continuous equations need to be approximated by algebraic equations at a set of locations in space and time. The locations at which the variables are to be calculated are defined by a computational grid, which is a discrete representation of the geometric domain. It divides the continuous domain into a finite number of subdomains, called cells. The governing equations are allowed to have values only at certain points of these cells and are in this way discretized and algebraically presented. The algebraic equations can then be solved using a computer.

Equations involved are often non-linear and are usually solved by successive linearization. This involves iteration on two levels: On the inner level a linear equation system is solved iteratively for a particular governing equation and the iterations on the outer level are needed because of the non-linearity and coupling of the equations. The outer level usually involves advancing the solution in time, even when solving for a time independent, steady state, solution.

Using CFD for simulating recovery boilers is challenging because of several unique characteristics. The essential property of these simulations is the combination of a large computational domain with important processes happening in very small areas of this large domain. One challenge is posed by the injection of black liquor, which is sprayed into the boiler as droplets. The droplets need to be tracked using a discrete phase model, which is computationally expensive and slows down convergence considerably. There is also a large number of chemical reactions taking place in the boiler and equations for all of the participating species need to be solved. In addition, boiler operation is very sensitive to the mixing provided by the air injections. These need to be modeled correctly to simulate the real operation with a given air injection model.

It has also been observed that there is no real steady state in the boiler and that there exist both large and small scale time dependencies in the flow. According to Grace et al. (1998), experimental results show that the real flows in the boiler are only quasi-stable and that there can be time dependent phenomena that cannot be predicted using a steady state CFD model. The same issue has been observed in time dependent large eddy simulations (LES) of recovery boilers. LES is very time consuming in boiler modeling and can not be used in practice except in individual simulations done for research purposes. Because of this, one must instead result to time averaged steady state modeling. Reynolds-averaged Navier-Stokes (RANS) turbulence models have been traditionally used to solve for these time averaged fields. It is often observed that even the RANS solutions exhibit time dependency and the quantities of interest need to be taken as iteration averages of the RANS solutions.

In practice, the iteration averaging approach works well for small boilers but when modeling large boilers the RANS solutions exhibit larger time dependencies. Because of this, it is often noticed that it might be hard to find a reliable solution even by iteration averaging the RANS solutions. The large dependencies might result from physical characteristics of the flow in the large boiler or they might be related to the properties of the models involved. It might also be that the grid densities traditionally used with large boilers are not sufficient because there are some small

scale phenomena happening in the boiler which require a certain minimum grid spacing. Even with these difficulties present, with considerable care and effort, reasonably accurate solutions are usually obtained for large boilers.

CFD based recovery boiler models have been used by a number of authors to study different phenomena in the boiler. For example, Engblom et al. (2012) have used CFD modeling to study asymmetric furnace temperatures in a recovery boiler and compared the results against validation measurements. Char bed modeling and processes have been studied by Bergroth et al. (2010) and Engblom et al. (2010). Vakkilainen et al. (1998) have analyzed high solids black liquor firing processes using CFD and compared the predictions of the model to measured data. Mueller et al. (2004) have used CFD modeling to study the influence of liquor-to-liquor differences on the overall furnace process.

Extensive furnace model validation work has been done by Grace et al. (1998). More recently, Miikkulainen et al. (2010) have found that validation of furnace models is challenging because the fluctuating furnace conditions in the real boiler make it very hard to get the modeled steady state and the measured situation to correspond to each other. Saviharju et al. (2004) have studied some of the weak points of the recovery boiler models using both CFD experiments and feedback data. Grace (1995) has reviewed the usage of furnace modeling critically and identified the most notable issues and uncertainties involved. Verification and validation in CFD in general have been extensively studied by Oberkampf and Trucano (2002) and Roache (1998). Quantifying errors involved in CFD computations has been discussed by Ferziger and Perić (1996).

Even though the furnace CFD models have not been thoroughly validated and there are known to be uncertainties involved in the simulations, the models have been found to be valuable tools in boiler design. The models predict the overall combustion process in the boiler well despite inaccuracies related to small scale phenomena. It has been seen in practice that changes in features or operation mode of a modeled boiler are reflected realistically in the overall boiler performance. This makes it possible to make comparisons of different designs based on their modeled performance. Because of these reasons the CFD furnace models can be used in the CFD-optimization framework.

1.3 Multi-Objective Optimization

As defined by Deb (2001), optimization means finding one or more feasible solutions which correspond to extreme values of one or more objectives. What it in practice means is that one is interested in finding a solution which makes the inspected system as good as possible in some defined sense.

When an optimization problem involves more than one objective, it is called a multi-objective optimization problem. Often these objectives are conflicting and optimal decisions will involve trade-offs between them. In presence of conflicting objectives, it is not possible to find a single optimal point, but rather a set of points that are optimal in the sense of Pareto. The points on the Pareto optimal front are the non-dominated group of points in the feasible set of points. A solution is said to dominate another solution when it is better than it in terms of all the objectives. This means that the solutions on the Pareto front have the quality that none of their objective functions can be improved in value without worsening the other objective functions in value. Mathematically, all the solutions on the Pareto front are equally good. The concept of Pareto optimality, and the Pareto optimal front, is visualized in Figure 1.2.

Normally, the objective of a multi-objective optimization problem is to find a diverse set of solutions close to the Pareto front. Choosing the most suitable solution on the front for some particular situation is natural to do after the optimization pro-

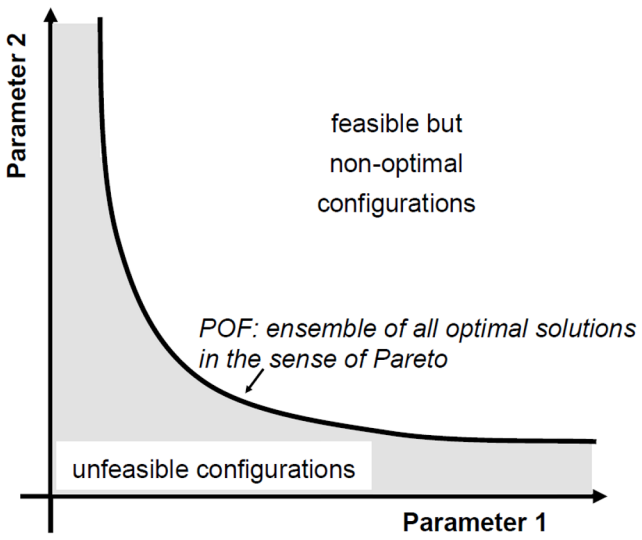


Figure 1.2: The concept of Pareto optimality. Regions of feasible and infeasible solutions are indicated on both sides of the Pareto optimal front (POF). Reprinted from Thévenin (2008).

cess. A human decision maker (DM), who is usually a professional working in the application domain, has an important role at this point. The decision making process involves making trade-offs and requires detailed knowledge about the relative importances of the objective functions. Professional information which is unfeasible to formulate mathematically can also be employed at this stage. A good example of such information in engineering applications is cost, which often needs to computed on an ad hoc basis.

A mathematical description of a multi-objective optimization problem has a number of objective functions which are to be minimized or maximized along with a number of constraints which the feasible solutions need to satisfy. According to Deb (2001), the general problem can be stated as

$$\begin{aligned} \text{min/max} \quad & f_i(\vec{x}), && i = 1, 2, ..., I; \\ \text{subject to} \quad & g_j(\vec{x}) \geq 0, && j = 1, 2, ..., J; \\ & h_k(\vec{x}) = 0, && k = 1, 2, ..., K; \\ & x_l^{min} \leq x_l \leq x_l^{max}, && l = 1, 2, ..., L; \end{aligned} \tag{1.1}$$

where f_i are objective functions, g_j are inequality constraints, h_k are equality constraints and x_l are design variables comprised in the design point vector $\vec{x}$. x_l^{min} and x_l^{max} are the lower and upper bounds of each design variable, respectively.

The design point vectors that respect the x_l^{min} and x_l^{max} bounds constitute a design space $\mathcal{D}$. The points in this space which also satisfy all the constraints g_j and h_k constitute a feasible space $\mathcal{S}$. Additionally, for each design point $\vec{x}$ in $\mathcal{D}$ there exists a mapping $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_i(\vec{x}))^T$ to the so called objective space $\mathcal{Z}$.

Different multi-objective optimization methods and their suitabilities for various engineering applications have been reviewed by authors such as Marler et al. (2004) and Belegundu and Chandrupatla (2011). Zhou et al. (2011) have focused on studying the applicability of several multi-objective evolutionary algorithms for different problems. Chinchuluun and Pardalos (2007) have reviewed developments in the field and formulated the mathematical features of some of the widely used optimization methods. Performance assessment of different algorithms has been discussed by Zitzler et al. (2003).

1.4 Combining Optimization with Computational Fluid Dynamics

The fundamental challenge in CFD-Optimization is that the values of the objective and constraint functions at a given design point $\vec{x}$ can only be obtained by performing a CFD computation and that even then it is not possible to obtain exact evaluations of the functions. As Thévenin (2008) has noted, evaluations done by CFD are results of approximate numerical simulations and include modeling, numerical, discretization, iteration and other errors. This results in a so called stochastic evaluation uncertainty which produces a large number of non-physical local maxima and minima. Furthermore, the functions are multivariable and most likely non-convex and non-linear.

For the aforementioned reasons, it is required from the optimization algorithm that it is robust in the sense that it does not get stuck in local maxima or minima. This means that a so called global optimization method should be used. The numerous available local methods can only guarantee convergence to a local optimum. The local methods can be used, more practically, by combining them with global methods. This can improve convergence speed of the optimization and also the accuracy of the final Pareto front obtained. When local and global methods are combined the resulting algorithms are called hybrid methods. Finally, the used method should not need any gradient information because it is not directly available and approximation of gradients might be inaccurate and would introduce additional uncertainties.

The global optimization methods can be classified into two categories: deterministic and stochastic methods. According to Rangaiah (2010), the deterministic methods utilize analytical properties of the optimization problem to find the global optimum and they require certain assumptions on some properties of the functions, for example, continuity or convexity. Convergence to the global optimum is guaranteed if and only if the assumptions are satisfied. The stochastic methods involve probabilistic elements in the search for the global optimum and require few or no assumptions on the characteristics of the optimization problem. These methods do not in theory guarantee convergence to an optimum in a finite amount of iterations. However, in practice, they often converge quickly to an acceptable global optimal solution. Because little information about the objective and constraint functions is available, the stochastic optimization methods are more suitable for the present application.

Genetic algorithms (GAs) are a particular class of stochastic global optimization methods which satisfy the aforementioned requirements. Furthermore, they require

no assumptions and need minimal information about the optimization problem. Because of these reasons they are suitable to be combined with CFD models. They also have an unique feature of finding and maintaining multiple solutions in a single optimization run. This corresponds to finding the whole Pareto front after the optimization. This makes them particularly useful in multi-objective optimization. GAs are well researched and have been successfully used in CFD-optimization by a number of authors. Foli et al. (2006) have used them for optimizing a micro heat exchanger design, Brizzolara et al. (2012) for optimizing a waterplane underwater hull shape and Beliganur et al. (2007) in a flow control optimization problem. GAs have also been used in combination with local methods, for example by Poloni et al. (2000) for optimizing a design of a sailing yacht fin keel. Because of these reasons, the optimization program developed in this work is based on a GA.

A single CFD simulation of a recovery boiler design commonly takes at least a few days of computer time, when supercomputers are not available, and thousands of different design points need to be evaluated when a GA is used. It is quickly realized that CFD based optimization is not feasible in this application without accelerating the optimization somehow. When a large number of evaluations is required, the largest accelerations can be obtained by reducing their total number or by reducing the computational cost of a single evaluation.

The cost of a single evaluation can be reduced by not using an excessively dense grid and by simplifying used models. When these methods are used their effects on the results should be analyzed. They must be employed without compromising the accuracy of the results. Both of these methods are utilized in this work and their effects on the simulations are discussed in the error estimation studies.

The number of evaluations needed can be reduced by utilizing machine learning approaches. The idea behind these methods is that knowledge already acquired from the problem is used to perform approximate evaluations and to reduce future computational costs. These approximate evaluation schemes, also called meta-model schemes, have been extensively studied by, for example, Simpson et al. (2001). The methods have been successfully combined with optimization algorithms in a number of studies. Both Georgopoulou and Giannakoglou (2009) and Duvigneau and Visonneau (2004) have studied combining hybrid methods with approximate evaluation schemes in complex design problems and proposed methodologies for using them concurrently with CFD. Gaspar-Cunha and Vieira (2005) have done comparable studies, proposed their own methodologies and tested them in polymer extrusion problems. Mengistu and Ghaly (2008) have combined a GA with an artificial neural network (ANN) meta-model in aerodynamic shape optimization, but in their ap-

proach the CFD model and the optimization algorithm are essentially uncoupled. An approximate evaluation scheme utilizing a machine learner based on a radial basis function (RBF) network is used in this work.

The CFD-optimization methodology has been utilized in different fields by a number of researchers. Thévenin and Janiga (2008) provide a compilation of different approaches to CFD-optimization and present how it has been used in a number of different applications. Van den Braembussche (2008) has used a combination of a GA optimization method and an ANN meta-model for turbine blade optimization. Hämäläinen et al. (2008) have studied using gradient-based optimization methods in papermaking applications. In the field of industrial boiler modeling, Saario (2008) has examined both the GA and the Powell's method for minimizing boiler emissions. A different CFD-optimization approach, where the so called adjoint equations are used instead of the flow equations, has been used in aerodynamic shape optimization by Soto et al. (2004) and Fazzolari et al. (2007). A large amount of work has also been done with response surface methods, where objective functions are approximated according to a number of CFD evaluations and optimum is then found according to these approximations. Madsen et al. (2000) have used response surface methods for a diffuser shape optimization and Keane and Scanlan (2007) have used them for a wing shape optimization. There have been few, if any, studies where CFD-optimization has been used in the field of recovery boilers. In this thesis, the CFD-optimization methodology is introduced to the recovery boiler design process.

1.5 Genetic Algorithms

Genetic algorithms (GAs) are optimization methods that are motivated by the principles of genetics and natural selection. They differ from other optimization algorithms fundamentally because they maintain a population of solutions during the optimization and because they require minimal information about the optimization problem. They have been extensively studied, among with other similar algorithms, by authors such as Bäck (1996), Deb (2001) and Srinivas and Patnaik (1994).

In a classical GA, a population of individuals is evolved towards the optimum by the processes of reproduction, crossover and mutation applied based on the fitness values of the individuals. Each individual represents a particular design point in optimization and is in essence a binary string that has the information about the design point encoded in it.

The algorithm essentially works by repeatedly allowing the fittest solutions to mate

together in the hope that the offspring solutions would inherit the best features of both parents. Random mutations are sometimes applied to the solutions to possibly add new features to the population not yet present in the parents. The working principle of a basic GA is schematically presented in Figure 1.3 and described below mostly according to Deb (2001).

After the initial population has been created, usually by inserting a desired number of random points into the design space, the main cycle of the GA is started. The individuals are first evaluated, which means calculating their objective function values and constraint violations. Based on the results of these evaluations, a fitness value is assigned for each solution. Fitness is a metric based on which the solutions can be ranked against each other. After the assignment fitness values, a convergence condition is checked. If it is not satisfied, genetic operators of selection, crossover and mutation are applied to the population.

The selection operator is used to make duplicates of good solutions and to eliminate

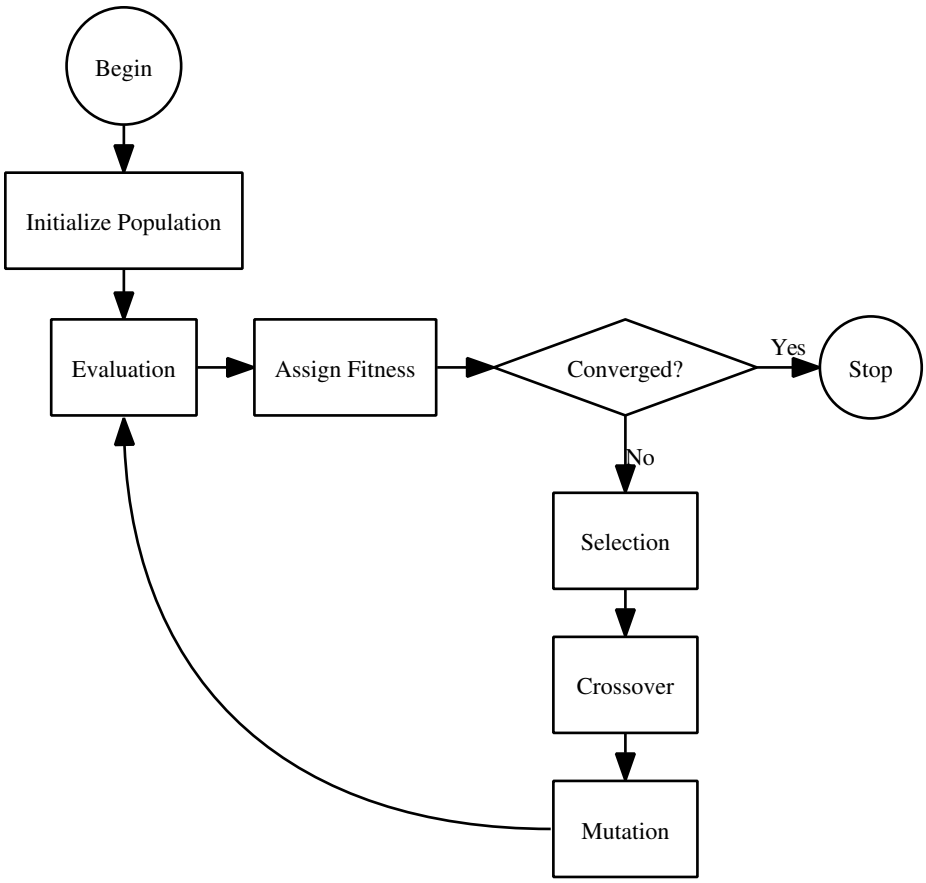


Figure 1.3: The working principle of a basic GA. Adapted from Deb (2001).

bad solutions in a population. The solutions remaining after using the selection operator is called the mating pool. The crossover operator is applied for generation of new solutions and it is responsible for the search portion of the algorithm. Pairs of strings are repeatedly picked from the mating pool and portions of them are exchanged with a crossover probability of p_c. The resulting strings are called offspring. Finally, the mutation operator is used on the population to expand the search space and to maintain diversity. It goes through every bit of every string in the population and changes a zero to one, and vice versa, with a mutation probability of p_m. After the operators have been used, the cycle is continued from the evaluation phase.

There exists a large number of different implementations of GAs, each with their own advantages and shortcomings. Konak et al. (2006) have conducted a thorough comparison of different GAs. They have found that the elitist non-dominated sorting genetic algorithm (NSGA-II) is efficient and has advantageous features. The elitism preservation feature of the algorithm is particularly useful in the present application. It means that the quality of the solutions found can only improve as the algorithm advances in generations. This is wanted in the present CFD-optimization application because the number of generations that can be run is limited by the computational time available. The NSGA-II algorithm is widely used and, according to Konak et al. (2006), also well tested. Because of these reasons, the NSGA-II algorithm is used in this work.

1.6 Machine Learning

Machine learning algorithms are designed to quantify relationships in a given data and to identify patterns for making predictions. The core objective of a learner is to generalize from its experience, which means the ability to perform accurately on new, unseen examples after having trained on a training data set. In CFD-optimization, machine learners can be used used to retain information already acquired in the course of the optimization and to make predictions based on this information to reduce future work.

Algorithms belonging to a class of machine learners called supervised learning methods are designed for inferring functions from training data of input-output pairs. As Haykin (1999) has noted, supervised learning can then be described as a curve fitting problem in a high-dimensional space. According to this viewpoint, learning is equivalent of finding a surface in a multidimensional space that provides the best fit to the training data according to some criterion. Correspondingly, generalization is

then equivalent to using this surface to interpolate function values in unseen data points.

Optimization using CFD poses particular requirements for the learning algorithm that is used for approximate evaluations. It needs to build a non-linear function using a low number of known points and, when the CFD model is coupled with the optimization process, also to be able to continuously learn from new observed input-output pairs.

Supervised learning methods, including kriging, polynomial regression, multilayer perceptron (MLP) networks and radial basis function (RBF) networks, can be used in optimization applications for approximate evaluations. Thorough comparisons of different approximate evaluation schemes have been conducted by Zhou et al. (2007) and Regis and Shoemaker (2004). The MLP and RBF networks are types of artificial neural networks (ANNs) and they are well suited for the present optimization problem because they are easily able to learn from new observed input-output pairs. MLP networks have been widely used in many applications and also in CFD-optimization. Zhou et al. (2004) have used them for interpolating from experimental data to optimize NOx emissions of a tangentially fired boiler. Gaspar-Cunha and Vieira (2005) have utilized MLP networks for approximate evaluations in a hybrid CFD-optimization scheme and Han and Maeng (2003) have combined MLP networks with CFD to optimize a fan design.

MLP networks need to be trained using the available database entries by using optimization methods for minimizing the evaluation errors of the network for the known points and this stage involves approximation of a number of parameters by the user. The results are highly dependent on the parameters given for the network and weak training is a common problem associated with bad choices of parameters.

RBF networks have a low number of parameters that need to be determined by the user and because of this they rarely have problems with weak training. They are also conceptually simpler than other neural networks. RBF networks have been used in CFD-optimization applications, for example, in problems related to aerodynamics by Georgopoulou and Giannakoglou (2009) and Duvigneau and Visonneau (2004). Furthermore, Jin et al. (2001) have compared different supervised learning methods and shown that RBF networks work well in a wide range of applications in generalizations from observed data. Because of the aforementioned reasons, an RBF network is used in this work.

1.7 Outline of the Thesis

The main goal of this thesis is to develop a CFD-optimization program and to use it in a multi-objective furnace geometry optimization task of a large capacity (7 000 tds/d) recovery boiler. The optimization problem is solved using two methods: an uncoupled method utilizing mainly approximate evaluations and a coupled method concurrently using both approximate and CFD evaluations. Before this task is approached, the developed program is verified and an error study is done on the used CFD model. These are the secondary goals of this work. The program verification consists of evaluating the performance of the implemented optimization and learning algorithms in test problems that have been used in literature. In the error study, magnitudes of iteration and discretization errors of the CFD solutions are assessed. This is important because the used optimization methods rely on the accuracy of the provided CFD evaluations.

The CFD-optimization program is constructed by integrating an existing CFD model with optimizer, learner and CFD preprocessor subprograms that are developed in this thesis. The program construction is schematically presented in Figure 1.4. It uses the CFD model to run simulations with given design variables to find the objective function values in these points. The model is based on ANSYS Fluent 14.0 CFD solver and it is combined with a postprocessing code developed in this work (ANSYS, 2011b). The CFD preprocessor subprogram is used to automate the simulation case setup and grid building processes. The CFD preprocessor subprogram uses ANSYS DesignModeler for geometry generation and ANSYS Meshing for grid generation (ANSYS, 2011c, 2011a). The commercial programs are used in combination with a preprocessing code built in this work.

In the CFD-optimization program, the objective function values with different design variables are used by the optimizer subprogram based on a particular optimization algorithm, the elitist non-dominated sorting genetic algorithm (NSGA-II), to move the design variables towards geometries with better objective function values. The boiler simulations done with the CFD model are computationally expensive and because of this a learner subprogram based on a meta-model, radial basis function (RBF) network, is developed to perform fast approximate objective function evaluations.

Theory and methods relevant for this work are presented in Chapter 2. The recovery boiler computational model is introduced. This includes the base design that is used in the optimization problem along with the detailed construction of the CFD model. Computational model error estimation methods are also introduced. Next,

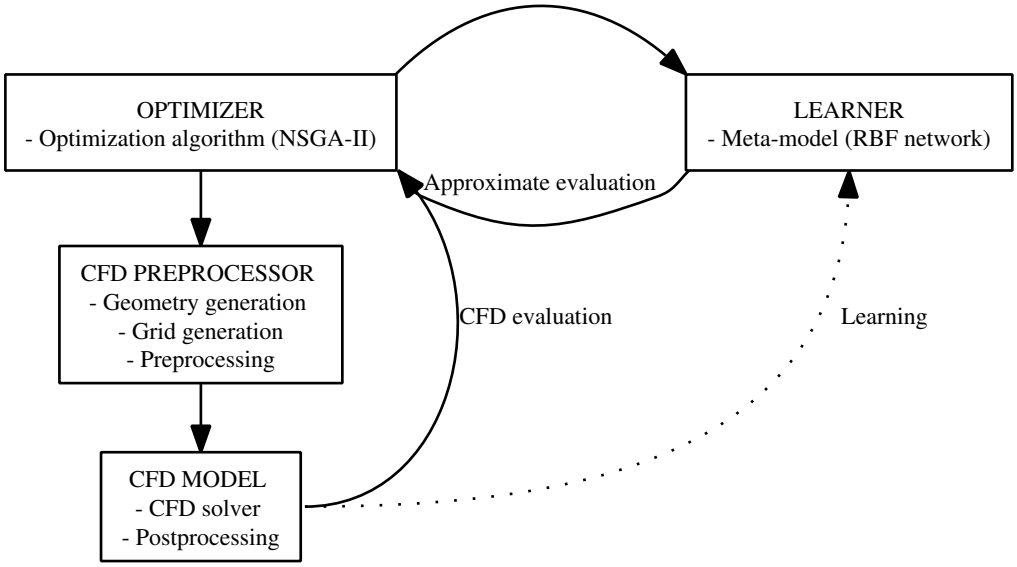


Figure 1.4: The developed CFD-optimization program. The boxes represent subprograms and the arrows indicate the directions of the information flow. Contents of each subprogram are listed below their titles.

the recovery boiler furnace geometry optimization problem is defined mathematically. The elitist non-dominated sorting genetic algorithm and radial basis function network that are used in the CFD-optimization program are presented. The chapter is concluded by combining the aforementioned ideas and by presenting the developed CFD-optimization program.

In Chapter 3, the results of the work are presented and discussed. The results of the computational model iteration and discretization error studies are presented first. Next, the results for verifying the developed optimization program are shown. Finally, the recovery boiler furnace geometry optimization results are presented and it is discussed how they can be utilized in practice.

Chapter 2

Theory and Methods

This chapter introduces the methods used in this work and the theory on which those methods are based on. Recovery boiler modeling in the context of this work is discussed and the furnace geometry optimization problem that is solved is formulated mathematically. The used elitist non-dominated sorting genetic algorithm (NSGA-II) is presented and the radial basis function (RBF) network machine learning method that is used for approximate evaluations is introduced. The chapter is concluded by presenting the CFD-optimization program developed in this work.

2.1 Recovery Boiler Computational Model

This section presents the recovery boiler computational model that is used in this work. First, the base design that is used in the optimization problem is presented and after this the computational model and the base grid are introduced. The behavior of the model during the geometry optimization task is also explained. Finally, the error estimation methods that are used to study errors in the simulations are presented.

2.1.1 Recovery Boiler Base Design in the Optimization Problem

The base design of the recovery boiler used in this work conforms to the basic design principles of boilers. The design has been done in collaboration with professionals working for a major boiler manufacturer and the design includes all the features of a modern high-efficiency recovery boiler. The furnace geometry of the base design, that is optimized in this work, is an arbitrary starting point and does not correspond to any existing recovery boiler. The design is presented schematically in Figure 2.1,

with the most important design features named. The figure is of the computational model but the modeling related features will not be focused on in this section.

Primary air ports are located on all four walls of the boiler. There are 34 ports with constant spacing on both the front and rear walls, and 14 ports with constant spacing on both the left and right walls. Secondary air ports are located on two levels and the lower level contains nine ports on both the front wall and the rear wall with constant spacing. The upper level contains five ports on the front wall and four ports on the rear wall in an interlaced arrangement. Dilute non-condensible gas (DNCG) is mixed to the input air on the rear wall on the both secondary air levels. DNCG gases are more humid than the normal combustion air and they consist mainly of various process emissions and exhaust gases. Tertiary air is also located on two levels and the arrangement of openings on these levels is identical. The levels have five ports on the front wall and four ports on the rear wall in an interlaced layout.

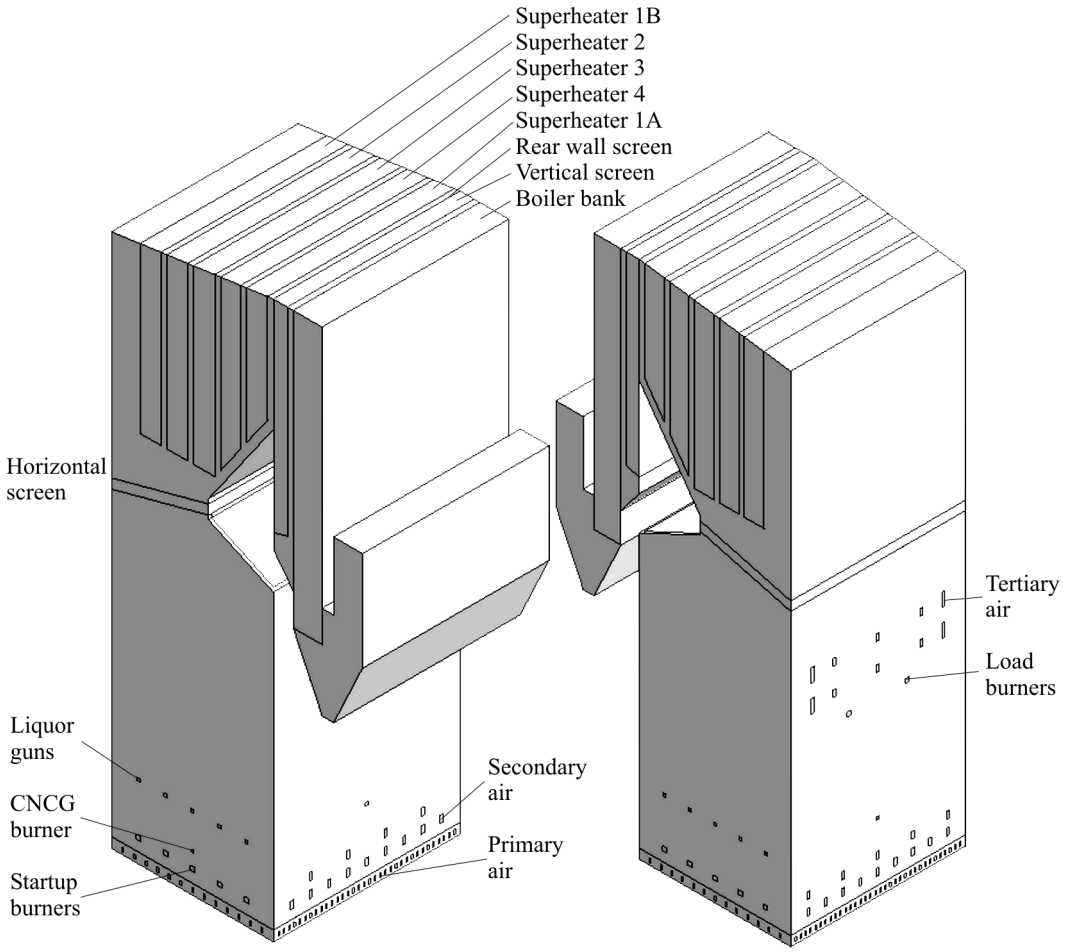


Figure 2.1: Isometric views of the boiler design. Locations of openings and heat transfer surfaces are indicated in the figure.

There are three different types of burners in the designed boiler: startup, concentrated non-condensible gas (CNCG) and load burners. There is only cooling air flowing into the boiler through the openings. There are five startup burners located on the lower secondary air level on both of the side walls. A single CNCG burner is located on the right wall on the upper secondary air level and two load burners are located below the lower tertiary air level on both the front and rear walls. The liquor gun level is located above the secondary air levels. There are five liquor guns on the both side walls and one gun on both front and rear walls. Leakage air is also flowing into the boiler through the gun openings.

There is a horizontal screen on the nose level of the boiler and superheaters are five in total: 1A, 1B, 2, 3 and 4. They are numbered with rising numbers and letters in the direction of the steam flow. Rear wall screen is located directly above the nose, after the superheaters. Vertical screen and boiler bank are located after the rear wall screen.

Combustion air is injected into the boiler in such a way that approximately 25% is injected on the primary level, 40% on the secondary level and 35% on the tertiary level. The air jets on the lower secondary level are interlaced so that there is always a strong jet opposing a weak jet. The both tertiary air levels are also interlaced in a similar way. On all the other levels the jets are equally strong.

The width of the boiler base design is 16.8 m, depth is 19.4 m and the elevation of the nose from the model bottom (nose height) is 31.5 m. These three parameters are changed during the optimization and are called the design variables in that context. When the design variables are changed, adjustments to the boiler design are done in such a way that the operation of the boiler is affected as little as possible to keep the optimization task meaningful.

Locations of the air ports and other openings change but they are placed on their respective walls using the same rules as in the base design. This means that their relative distances from the walls and each other stay constant. The elevations of the openings from the model bottom do not change.

Volumes of the superheaters are kept constant so that the amount of the heat transfer to them can be kept fixed. This means that when the boiler width is changed the depths of the heat transfer surfaces need to change in a way which keeps their volumes constant. The distances between all the heat transfer surfaces are kept fixed. Depth of the nose is always 40% of the boiler depth and its shape is constant.

Volumes of the screens and the boiler bank are allowed to vary when the boiler

dimensions are changed but the total heat transfer to them is kept constant. This represents a situation where spacing of the tubes is varied in different boiler geometries so that the heat transfer surface area stays approximately constant. The boiler geometry behind the rear wall screen is kept as similar as possible between different designs. This is to minimize its impact on furnace conditions and to ensure good outlet conditions for the flue gas.

2.1.2 Construction of the Computational Model

The computational fluid dynamics (CFD) model used in this work is based on ANSYS Fluent 14.0 customized with numerous in-house sub-models (ANSYS, 2011b). The performance of the model has been validated for certain actual boilers by comparing measured data to simulated results. The performance has been found robust and the model has produced accurate results for a wide range of boiler designs.

In the model, combustion air is injected into boiler using an in-house code. Air ports and other openings are modeled as patches on the boiler walls whose opening areas can be changed by code. The opening areas are adjusted by the code automatically to get the mass flow and velocity that are given as input by the user. This corresponds well to the real operation of the air ports where they can be closed partially or fully with velocity dampers. Boundary conditions for turbulence, flow temperature and species concentrations are given for the air ports individually.

The heat transfer surfaces are modeled as momentum and heat sinks in the equations in the volumes they occupy. The heat sink values are provided by heat balance calculations of the boiler based on design parameters. The momentum sink values are approximated for all heat transfer surfaces by taking into account their individual physical constructions.

Liquor spraying to the boiler is done by an in-house code which forms several sheets of liquor in both vertical and horizontal directions. The particle quantity and size distributions have realistic values in both directions. Black liquor particle trajectories and reactions in the boiler are modeled with an in-house code, which is based on the ANSYS Fluent 14.0 discrete phase model (DPM) (ANSYS, 2011b). These codes are largely based on the work by Kankkunen and Miikkulainen (2003), Miikkulainen et al. (2009) and Kankkunen et al. (2011). The DPM sources are updated after every 100 iterations. When the black liquor particles hit the walls below the nose, the particles are combusted there. If the particles move above the nose, they are classified as carryover and do not participate in the reactions anymore.

The flow field is solved by the pressure based solver with the SIMPLE scheme. Turbulence is solved with the standard k-ϵ model and wall functions are used. Energy equation is included in the model and the P-1 model is used as the radiation heat transfer model (ANSYS, 2011e). Gas phase species transportation equations in the boiler that are solved are equations for H_2O, O_2, CO, CO_2, H_2, CH_4, H_2S, SO_2, Na, NaCl, NaOH and Na_2SO_4. Mass fraction of N_2 is solved by subtracting mass fractions of all the other species from 1.0. The gas phase reaction rates are solved by the Finite Rate/Eddy-Dissipation model.

In this work, the grids used are built using ANSYS DesignModeler and ANSYS Meshing programs (ANSYS, 2011c, 2011a). The employed meshing method makes predominantly hexahedral meshes with tetrahedral cells in areas where hexahedral cells are hard to place. Local mesh refinements are handled in the structured areas by dividing the hexahedral cells into eight parts and by using hanging nodes. With the model used in this work, the method generates meshes in which more than 95% of the cells are hexahedral.

The base grid has 530 000 cells that are mainly hexahedral and it has been locally refined near the air ports, liquor guns and burners. It is also finer on the air port levels and near the walls than in the other parts of the boiler. The total number of cells used in the grid is dictated by computational resources available. The whole mesh for the base boiler is shown in Figure 2.2. The mesh in the superheater area of the boiler is also finer than in the upper part of the furnace. This is to ensure that enough cells are placed in the volumes that represent the heat transfer surfaces to model them correctly. The mesh extends over the boiler bank area in order to get an accurate solution on the nose level and also in the heat transfer area. The outlet is located behind the boiler bank.

The lower furnace mesh with one corner of the model cut off is presented in Figure 2.3. The mesh refinements near the air ports can be seen well in the figure along with the refinement of the mesh on the air port levels. The mesh is finer near the primary air ports, the CNCG burner and liquor gun openings than near the secondary air ports and the startup burners. This is because physically smaller ports require a finer mesh near them so that it is ensured that there are enough cells in the patches that represent them. The char bed can also be seen in the figure. Its shape is fixed throughout the simulation and it is modeled as a pyramid with its top cut off.

The automated mesh generation for new designs is built in such a way that certain requirements on the grid are always taken into account. The grid is not stretched

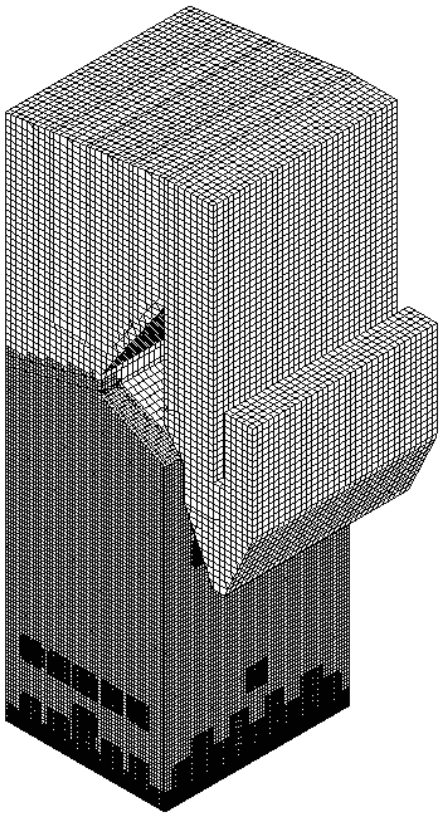


Figure 2.2: The mesh for the recovery boiler base design. The mesh is finer on the air levels and near the walls than in the other areas of the boiler. It has also been locally refined near the air ports, liquor guns and burners.

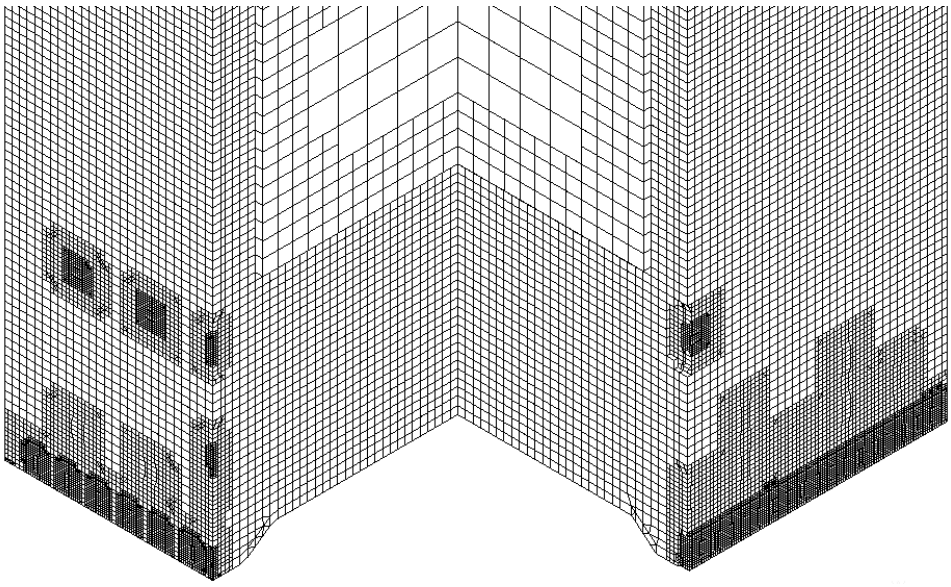


Figure 2.3: The mesh for the recovery boiler base design in the lower furnace with one corner of the model cut off. Refinements near the openings and the shape of the char bed can be seen.

when the geometry of the boiler is changed, because this would quickly cause problems with the quality of the grid. Rather, the base size of the cells is kept constant and cells are added or removed as needed when the geometry is changed.

A manual method for establishing convergence commonly used in recovery boiler simulations is based on the criterions that a sufficiently high number of iterations have been done and that the values of some monitored quantities on the nose level, such as average temperature, CO content or O_2 content, settle to oscillate around some values as functions of iterations. It is known from experience that even with a low quality initialization file for the fields this usually happens between 10 000 and 20 000 iterations.

A convergence monitoring code is developed in this work which is based on experience from using the model and on test simulations. The code works by monitoring values of the residuals of the solved quantities. Using the residuals for monitoring convergence is more reliable than monitoring the solved variables or their functionals. This is because the sizes of the residuals are related to the magnitudes of errors present in the solution. The solved variables can be monitored in addition to the residuals for confirmation.

Residual R^ϕ of a variable ϕ is obtained by summing the absolute values of the cell imbalances over all cells in the domain,

$$R^\phi = \sum_{cells} |\sum_{nb} a_{nb}\phi_{nb} + b - a_P\phi_P|, \tag{2.1}$$

where P refers to the cell, nb refers to neighbor cells of P, a is an influence coefficient and b is the contribution of the source terms and boundary conditions (ANSYS, 2011d).

Test simulations run with the present model have shown that residuals behave similarly in simulations with different geometries and that they settle to move around some values after approximately 5 000 iterations or even earlier. The tests were run by always initializing the fields with an already computed solution with the most similar geometry. Plots of the residuals in an example simulation as functions of iterations on a logarithmic scale are shown in Figure 2.4. The figure shows that the magnitudes of the residuals settle around their respective levels around 2 500 iterations and that the values do not visibly decrease in magnitude after that even if the iterations are continued to high counts.

Based on the aforementioned considerations, a convergence criterion is set for simulations done with the model. It is required that the residual of each variable

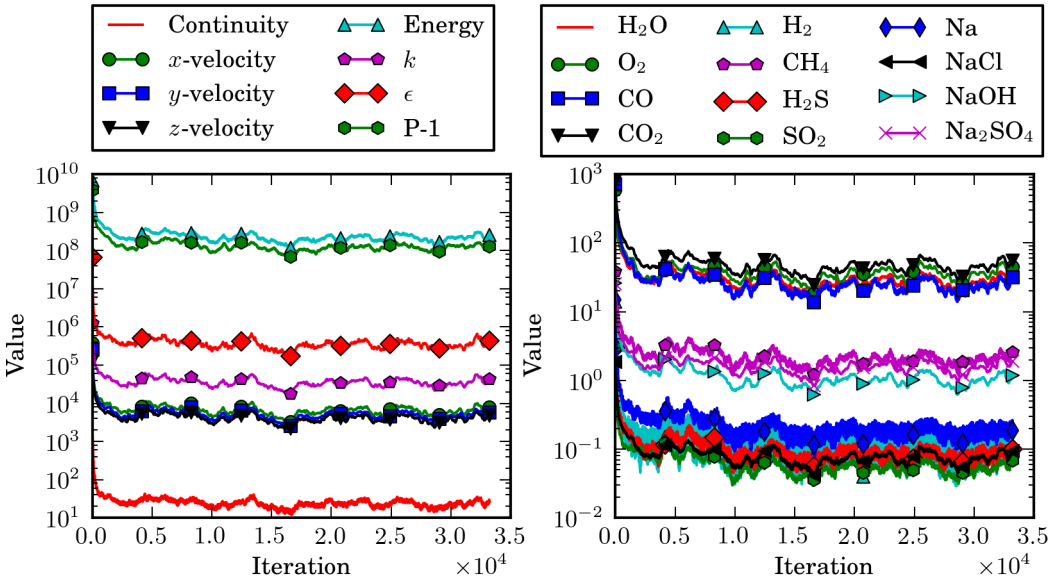


Figure 2.4: Sample plots of residuals as functions of iterations. Magnitudes of the residuals settle around their respective levels around 2 500 iterations and the values do not visibly decrease in magnitude after that if iterations are continued.

decreases to a certain magnitude and that the residual values are not substantially changing anymore as functions of iterations. The allowed maximum magnitudes for each residual are summarized in Table 2.1 and the residual values are concluded unchanging if their relative percentage changes over the last 500 iterations are all less than 50%. The magnitudes given in the table are high in general, but from experience with the model it is known they are what are commonly observed in converged simulations. The magnitudes of the residuals stay high even when more iterations are done or the model settings or grid changed.

Monitoring for convergence is always started at 5 000 iterations and the code checks after every 250 iterations whether the convergence criterions are satisfied. If the criterions are met, 5 000 iterations more are run in the converged state and after this the simulation is stopped. This is because values of average temperature, CO content, O_2 content and carryover on the nose level are taken from the simulations as iteration averages and the values included in the averages need to be from a converged simulation. If the code has not established convergence at 15 000 iterations then the simulation is stopped and it is concluded that with that geometry the simulation does not converge in a reasonable time. With these settings the iterations done in a single simulation are limited to between 10 000 and 20 000 in total. This means that, with the computers used in this work, one simulation approximately requires a computing time of between 24 and 48 hours.

Table 2.1: The allowed maximum magnitudes of the residuals of the different variables for convergence.

Variable	Allowed magnitude
Energy, P-1	10^8
ϵ	10^5
k	10^4
x, y and z -velocities	10^3
Continuity	10^1
H_2O, O_2, CO, CO_2	10^1
CH_4, NaOH, Na_2SO_4	10^0
H_2, H_2S, SO_2, Na, NaCl	10^{-1}

2.1.3 Computational Model Error Estimation

This work includes a computational model error study, where the objective is to assess the magnitudes of errors involved in the CFD solutions. Because the solutions are used in the geometry optimization, it is essential to know how much they should be trusted. Iteration and discretization errors are studied and the methods used are presented in this section

The iteration error study is done using a simple method developed in this work. In the approach, a CFD computation is run with the base grid until it can be manually detected from the iteration history that the solved variables (average temperature, CO content, O_2 content and carryover on the nose level) are not changing or are oscillating around some values as functions of iterations. After this, iterations are continued in this converged state and values of the solved variables are tracked. The magnitudes of iteration errors in these quantities are assessed by calculating their minimum and maximum values and minimum and maximum values of their running iteration averages in this iteration range. Profiles of temperature, y-velocity, CO content and O_2 content as functions of boiler height are also drawn periodically in the converged iteration range at suitable intervals.

The aforementioned approach shows how much simulation results, represented by the monitored quantities, can vary when the iterations are stopped at an arbitrary point when the iteration history seems to indicate that the simulation is converged. The magnitudes of the variations in the running averages and drawn profiles can also be compared against the results of the discretization error study. By this comparison it can be concluded whether the changes in the results are significant enough to be

attributed to the differences in the grids or can they be resulting from iteration error.

Studying the grid is important because the obtained computational solution should not be affected by the grid used. This is called grid independence. Strictly speaking, grid independence is achieved at the limit when grid spacing approaches zero, because only then does the discretization error go to zero. In practice, grid independence is seldom sufficiently achieved. Regardless, it should be made sure that grid quality and density are adequate for the grid to be in the so called asymptotic range and assessed how much the solution is changing when the grid is refined.

The grid study is done by following a method and reporting conventions proposed by Roache (1994). This method involves calculating a grid convergence index (GCI) for the grid used, which is based on discretization error estimators derived from the theory of Richardson extrapolation. The basic idea is to approximately relate the results from any grid refinement test to the expected results from a grid doubling when a second-order method is used. The GCI is a conservative estimator for the error and tends to rather overestimate than underestimate the errors. The method is used as described by Roache (1994) and presented in Appendix A.

In general, estimating the discretization error involves running the same simulation on two or more grids and comparing the results. The GCI is an estimate of the error band of the solution on the examined grid and it can be calculated for all the grids used in the comparison. In the case of comparing solutions on a coarse and fine grid, it can be calculated for the fine grid as

$$\mathrm{GCI}_{fine} = \frac{F_s|\epsilon|}{r^p - 1}, \tag{2.2}$$

and for the coarse grid as

$$\mathrm{GCI}_{coarse} = \frac{F_s|\epsilon|r^p}{r^p - 1}, \tag{2.3}$$

where F_s is a factor of safety, ϵ is the relative error, p is the order of the discretization methods used and $r = h_2/h_1$ is the grid refinement ratio computed using the grid spacings h_1 (fine grid) and h_2 (coarse grid). The factor of safety is recommended by Roache (1997) to be $F_s = 3.0$ for comparisons of two grids and $F_s = 1.25$ for comparisons over three or more grids. The relative error is defined as

$$\epsilon = \frac{f_2 - f_1}{f_1}, \tag{2.4}$$

where f_1 is the solution on the fine grid and f_2 is the solution on the coarse grid.

The method of using GCIs depends on the assumption that the grids are in the asymptotic range, where the error decreases with a speed relative to the order of the discretization method used as the grid is refined. If the asymptotic range has not been achieved the error estimates obtained using the GCIs are not valid. According to Roache (1994), it can be checked whether a grid is in the asymptotic range by computing the same case using two progressively refined grids and by examining their GCI values. All of the grids are in the asymptotic range if $\mathrm{GCI}_{23} \approx r^p \mathrm{GCI}_{12}$, where GCI_{23} is computed from the intermediate to the coarse grid and GCI_{12} from the fine grid to the intermediate grid. This idea is developed further in this work and the relative difference of GCI_{23} and $r^p \mathrm{GCI}_{12}$ is formulated into an asymptotic range indicator S

$$S = \frac{|\mathrm{GCI}_{23} - r^p \mathrm{GCI}_{12}|}{\mathrm{MAX}(|\mathrm{GCI}_{23}|, |r^p \mathrm{GCI}_{12}|)}, \tag{2.5}$$

where MAX is a function that returns the larger of its arguments. The absolute difference $|\mathrm{GCI}_{23} - r^p \mathrm{GCI}_{12}|$ is scaled with a reference value that is always the larger of the two quantities. This makes sense because the quantities should be equal to each other and neither one of them can be considered to be a definite reference value. The quantity S obtains values that are between 0 and 1 and signifies how much smaller the smaller of the two values is relative to the larger one. If the value of S is close to 0, it is an indication that the grids are in the asymptotic range.

Based on this theory, uniform reporting of the results of grid-refinement studies with arbitrary r and p are possible. This is particularly useful when using $r = 2$ would make the computational cost of the study too high, as in the present situation. Regardless, when using small values for r the leading truncation error term might become too small and be masked by errors from other sources. Roache (1994) suggests that $r = 1.1$ is a suitable practical minimum value for the grid refinement ratio. A value of $r = 1.2$ is used in this work.

In the present case, the grids are generated using an automated method which places unstructured cells in areas where hexahedral cells are hard to place. Because of this, the final refinement ratios need to be computed using a formula for effective grid refinement ratios

$$r_{eff} = (\frac{N_1}{N_2})^{1/D}, \tag{2.6}$$

where N is the total number of cells in the grid and D is the dimensionality of the flow domain.

2.2 Recovery Boiler Optimization Problem

In this work, a large capacity recovery boiler furnace geometry is optimized according to multiple performance criterions and constraints. This multi-objective optimization problem is mathematically formulated as

$$\begin{aligned} \min \quad & f_{carry}(\vec{x}),\ f_{pWall}(\vec{x}),\ f_{CO}(\vec{x}),\ -f_{lowT}(\vec{x}), \\ & f_{COspi}(\vec{x}),\ f_{yVel\sigma}(\vec{x}),\ f_{T\sigma}(\vec{x}),\ f_{O2}(\vec{x}), \\ \text{subject to} \quad & g_T^{min} \leq g_T(\vec{x}) \leq g_T^{max}, \\ & g_{CO}(\vec{x}) \leq g_{CO}^{max}, \\ & w^{min} \leq w \leq w^{max}, \\ & d^{min} \leq d \leq d^{max}, \\ & h^{min} \leq h \leq h^{max}, \end{aligned} \tag{2.7}$$

where the variables superscripted with max or min refer to the upper or lower bounds of the corresponding variables, respectively. The function $f_{lowT}(\vec{x})$ is wanted to be maximized and so it is multiplied by -1 in the problem formulation. The design point vector $\vec{x} = (w, d, h)$ is comprised of boiler width, boiler depth and nose height variables, in this order. The meanings of functions are as given in Table 2.2. Standard deviation is shortened as σ in the table and the decided relative importances of the functions are indicated in the last column. An importance of one is the highest and three the lowest.

Preferences about the objective function importances need to be taken into account in the optimization because there exist many essential criterions for a good boiler design but the different criterions are not equally important. It is known from experience with actual boilers and the computational model that a boiler design that satisfies the CO content and nose temperature constraints and has good values in the carryover and the particles landing on the walls objectives is most likely a good design. These two objectives are significantly more important than the others.

Another reason for including the preferences in the optimization is that the high dimensionality of the present problem makes it mathematically difficult and slows down convergence of the optimization considerably. As demonstrated by Branke et al. (2001), in high dimensional cases it becomes essential to implement preference based guidance methods, also called weighting schemes, in the algorithms. They

Table 2.2: The meanings, units and decided importances of the objective and constraint functions. The importances of the objective functions are inspected only if the constraints are satisfied.

Function	Meaning	Unit	Importance
$g_{CO}(\vec{x})$	Mean CO content at nose level	ppmw	Essential
$g_T(\vec{x})$	Mean temperature at nose level	°C	Essential
$f_{carry}(\vec{x})$	Mean carryover at nose level	kg/s	1
$f_{pWall}(\vec{x})$	Liquor particles landing on walls	% of wet flow	1
$f_{CO}(\vec{x})$	Mean CO content at nose level	ppmw	2
$f_{lowT}(\vec{x})$	Mean temperature below liquor guns	°C	2
$f_{COspi}(\vec{x})$	CO spikes at nose level	% of area	3
$f_{yVel\sigma}(\vec{x})$	y-velocity σ at nose level	m/s	3
$f_{T\sigma}(\vec{x})$	Temperature σ at nose level	°C	3
$f_{O2}(\vec{x})$	Mean O_2 content at nose level	wt%	3

write that using these methods accelerates convergence and also leads to a better solution, with given computational resources, in the interesting parts of the unweighted Pareto front. The weighting methods try to guide the algorithm to the regions of the objective space that are the most favorable based on the defined preferences. The downside of these methods is that the whole unweighted Pareto front is not found after the optimization since some of the unweighted Pareto solutions are found unfavorable because of the weighting scheme. This means that some information is lost regarding the relationships of the objectives. In practice, this is not an issue if the weights are set correctly. The technical implementation of the weighting scheme in the code is described in Section 2.3.

The CO content and mean nose temperature are constrained by functions g_{CO} and g_T. A low CO content is wanted at nose level because it is a good indication of that the combustion processes in the furnace have completed. Little combustion occurs after the nose and material left uncombusted there will foul the heat transfer surfaces and raise the emission levels of the boiler. The content is constrained below 200 ppmw to exclude very bad designs early in the optimization process. The mean temperature on the nose level is constrained to 998 ± 10 °C, because the heat balance calculated for the design is based on a nose temperature of 998 °C. If temperature on the nose moves too far away from this value, the heat balance becomes invalid and the simulated boilers do not accurately represent the designed boiler any more.

The carryover and amount of particles landing on the walls are the two most im-

portant objectives. A low carryover is wanted because it indicates how well the fuel injected into the boiler combusts. Carryover fouls the heat transfer surfaces and raises boiler emissions. The amount of liquor particles landing on the walls is desired to be small because for good boiler operation the particles should mainly land on the char bed. It is known from experience with actual boilers that if a high amount of particles is landing on the walls the boiler does not operate well.

The second most important objectives are the CO content on the nose level and the average temperature in the lower furnace. The importance of the low CO content was discussed above. A high average temperature in the lower furnace, below liquor guns, is demanded since in a good boiler most of the combustion occurs in the lower furnace and a high temperature there is a good indication of this.

The least important objectives are the amount of CO spikes, upward velocity (y-velocity) standard deviation and O_2 content on the nose level. The amount of CO spikes on the nose level should be low because it is known from experience that very large deviations from the mean CO value in some areas on the nose level should be avoided. These spikes sometimes account for the most of the total CO emission and they might also indicate imbalances in the boiler operation. The CO spikes are measured by calculating the relative area on the nose level in which the CO content gets values that are larger than 300% of the mean value on the level. A low upward velocity standard deviation is wanted because it is an indication of an even upward velocity horizontal profile on the nose level. It signals that there are no spikes of high velocity present which might carry uncombusted particles out from the furnace. A low O_2 content is an indication that the combustion processes have completed in the furnace. It is less important than the CO content but needs to be used together with it because the two objectives are known to be conflicting. When CO content is on a low enough level then it becomes important to minimize the O_2 content also.

The bounds on the constraint functions and design variables in Equation (2.7) are summarized in Table 2.3. The MAX and MIN in the table are functions, which return the largest or smallest of their arguments, respectively. Functions c_1, c_2 and c_3 are ad hoc constraints that were developed in this work. The design space in the presence of the constraints on the geometry given in the table is visualized in Figure 2.5.

The c_1 function gives a minimum depth for a boiler of a given width that can fit the superheaters. It was derived by observing how the geometries of the constant volume superheaters must change when the boiler geometry is changed. The functions c_2 and c_3 take into account the required residence time for the gas between the upper

Table 2.3: A summary of the bounds on the functions and variables. MAX and MIN are functions which return the largest and smallest of their arguments, respectively.

Function/Variable	Value	Unit
g_T^{min}	988	°C
g_T^{max}	1008	°C
g_{CO}^{max}	200	ppmw
w^{min}	13.0	m
w^{max}	27.0	m
d^{min}	MAX(13.0, $c_1(w)$)	m
d^{max}	27.0	m
h^{min}	MAX(26.0, $c_2(w,d)$)	m
h^{max}	MIN(45.0, $c_3(w,d)$)	m

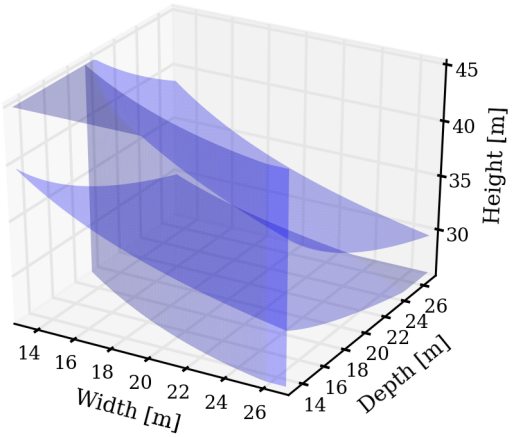


Figure 2.5: The design space in the presence of the geometric constraints. The feasible space is the region of the design space where all the constraint functions g are also satisfied.

tertiary air and nose levels according to boiler design practices. They constrain the nose height from below and above.

2.3 Elitist Non-Dominated Sorting Genetic Algorithm

In the elitist non-dominated sorting genetic algorithm (termed NSGA-II) proposed by Deb et al. (2002), in addition to the usual features of genetic algorithms (GAs), there is an elitism-preserving operator and a diversity-preserving mechanism. Fur-

thermore, the algorithm is modified in this work to employ a weighting scheme proposed by Cvetkovic and Parmee (2002) to enable guidance of the algorithm based on information about the relative importances of the different objective functions. In this subsection, the algorithm is presented, according to Deb et al. (2002), and it is shown how it is used in the developed optimization program.

The algorithm is elitist, which means that it is made sure that the fitness of the best solutions in the population does not deteriorate as a function of generations. This way a good solution found early on is never lost unless a better solution is found. Elitism preservation gives the algorithm favorable convergence properties and the presence of elites in the population enhances the probability of creating better offspring. Diversity preservation, which is also provided in the algorithm, is important to ensure the secondary objective of multi-objective optimization of finding a good spread of solutions on the Pareto front.

Initializing the algorithm involves first encoding the design points into binary strings. A binary substring s_i is formed from each design variable and the substrings are concatenated together as $s = \cup s_i$ to form the binary string representing a particular design point. The resolution available for each of the design variables is dictated by the chosen substring lengths. If b_i is the chosen substring length for the i:th design variable, then the number of intervals available between the minimum and maximum values of that variable is 2^{b_i}. Each individual in the population is then a binary string of zeroes and ones.

The basic principles of GAs given in Figure 1.3 still apply in the NSGA-II algorithm, but some of the operations are partially modified. The complete NSGA-II procedure is shown in Figure 2.6. The algorithm starts with the creation of the initial population and the assignment of a fitness rank r and a crowding distance value d to each individual. The assignments are done using the non-dominated-sort and crowding-sort procedures on lines from three to six of the algorithm.

Determination of the fitness ranks is based on the definition of dominance. In the presence of constraints, a feasible solution always dominates an infeasible solution, and if two solutions are both infeasible the one with the smaller constraint violation dominates the other. If both solutions are feasible, then relative dominance is based on their objective function values. The weighting scheme proposed by Cvetkovic and Parmee (2002) is used here to enable guidance of the algorithm based on preferences of the designer about the different objectives. Weighted dominance, that is inspected in the case of comparing two feasible solutions, is defined for a minimization problem

```
1: procedure NSGA-II
2:     Create initial population P_t
3:     NON-DOMINATED-SORT(P_t) to identify fronts F_i, i = 1, 2, ...
4:     for each F_i do
5:         CROWDING-SORT(F_i) to get crowding distance values of individuals
6:     end for
7:     while Not converged do
8:         Create Q_t from P_t using selection, crossover and mutation operators
9:         R_t ← P_t ∪ Q_t
10:        NON-DOMINATED-SORT(R_t) to identify fronts F_i, i = 1, 2, ...
11:        P_{t+1} ← ∅
12:        i ← 0
13:        while |P_{t+1}| + |F_i| ≤ n do
14:            CROWDING-SORT(F_i) to get crowding distance values of individuals
15:            P_{t+1} ← P_{t+1} ∪ F_i
16:            i ← i + 1
17:        end while
18:        CROWDING-SORT(F_i) to get crowding distance values of individuals
19:        Add the most widely spread (n − |P_{t+1}|) solutions in F_i to P_{t+1}
20:        t ← t + 1
21:    end while
22: end procedure
```

Figure 2.6: The NSGA-II algorithm. Adapted from Deb (2001).

as

$$\vec{x_i} \text{ dominates } \vec{x_j} \iff \sum_{k:f_k(\vec{x_i})\leq f_k(\vec{x_j})} w_k \geq \tau, \tag{2.8}$$

where $\vec{x_i}$ and $\vec{x_j}$ are two individuals, $f_k(\vec{x})$ are the objective functions, w_k are the weights associated with each $f_k(\vec{x})$ and τ is a parameter for the minimum level of dominance. The sum of all weights w_k is 1.0 and if $\tau = 1.0$, the standard definition of dominance without the weighting scheme is obtained. Weights that are used in this thesis are given in Table 2.4. They are based on the considerations presented in Section 2.2.

The fitness values r are assigned by a non-dominated-sorting procedure. It classifies a given population P into fitness fronts F_i where each individual has the fitness rank $r = i$. In the sorting, the best fitness front F_1 is the group of non-dominated solutions in P and the other fronts F_i are analogously the non-dominated solutions in $R_t \setminus \cup_{k=1}^{i-1} F_k$.

The crowding-sort procedure, presented in Figure 2.7, assigns good crowding distance values d to the individuals which are far away from other solutions with the same fitness rank. This process preserves diversity. To get an estimate of the density

Table 2.4: The weights of the objective functions used in this thesis. The minimum level of dominance τ determines in how many objectives a solution $\vec{x_i}$ needs to be better than a solution $\vec{x_j}$ for $\vec{x_i}$ to dominate $\vec{x_j}$.

Function/Variable	Meaning	Weight
$f_{carry}(\vec{x})$	Mean carryover at nose level	0.3
$f_{pWalls}(\vec{x})$	Liquor particles landing on walls	0.3
$f_{CO}(\vec{x})$	Mean CO content at nose level	0.1
$f_{lowT}(\vec{x})$	Mean temperature below liquor guns	0.1
$f_{COspikes}(\vec{x})$	CO spikes at nose level	0.05
$f_{yVel\sigma}(\vec{x})$	y-velocity σ at nose level	0.05
$f_{T\sigma}(\vec{x})$	Temperature σ at nose level	0.05
$f_{O2}(\vec{x})$	Mean O_2 content at nose level	0.05
τ	Minimum level of dominance	0.85

of solutions surrounding a particular solution $\vec{x}_j$ in the objective space $\mathcal{Z}$, the distance between the closest solutions on each side of solution $\vec{x}_j$ is taken with respect to each of the objectives. Each distance is normalized with the distance between front-maximum and front-minimum values of that particular objective function. This results in the expression $(f_m(F[o_{j+1}]) - f_m(F[o_{j-1}]))/(f_m^{max} - f_m^{min})$ for each function f_m. The distances corresponding to each f_m are added together to get the crowding distance d_j.

After the fitness and crowding distance values for the initial population have been evaluated, the main cycle of the NSGA-II is started. This is shown on lines from seven to twenty-one of the algorithm in Figure 2.6. The cycle begins with applying the genetic operators of selection, crossover and mutation to the population.

The algorithm uses a crowded tournament selection operator. It first doubles the population of size n to size $2n$. Then, two random individuals are repeatedly picked from it and the better of them is placed in the mating pool and the other discarded. The better solution is the one with the better fitness rank. If the individuals have the same rank, the individual that has the better crowding distance value is the better one. The picking process is repeated until a mating pool of size n has been obtained. Single-point crossover is used as the crossover operator. It chooses two individuals from the mating pool at random and a single crossover site on the binary strings at random. Then it exchanges the contents on the right hand side of the crossover site between the strings with a crossover probability of p_c. Two new offspring individuals are created this way. The two chosen individuals are removed from the mating pool

```
1: procedure CROWDING-SORT(F)
2:     for i ← 1, 2, ..., |F| do
3:         d_i ← 0
4:     end for
5:     for m ← 1, 2, ..., M do
6:         Sort the set F in worsening order of objective function f_m values to vector o⃗
7:         d_{o_1} ← ∞
8:         d_{o_|F|} ← ∞
9:         f_m^max ← f_m(F[o_|F|])
10:        f_m^min ← f_m(F[o_1])
11:        for j ← 2, 3, ..., |F − 1| do
12:            d_{o_j} ← d_{o_j} + (f_m(F[o_{j+1}]) − f_m(F[o_{j−1}]))/(f_m^max − f_m^min)
13:        end for
14:    end for
15: end procedure
```

Figure 2.7: The crowding-sort procedure. The procedure takes a set F as an input and assigns crowding distance values to its individuals. Adapted from Deb (2001).

and the crossover process is repeated until the mating pool is empty. The algorithm uses a bit-wise mutation operator in which every bit in the strings of the offspring population is gone through and is changed with a mutation probability of p_m.

After the genetic operators have been applied, a modified way of assembling the new population is employed to introduce elitism and diversity preservation. The process is done on lines from nine to nineteen of the algorithm in Figure 2.6 and illustrated in Figure 2.8.

In the process in Figure 2.8, after the offspring population Q_t of n individuals has

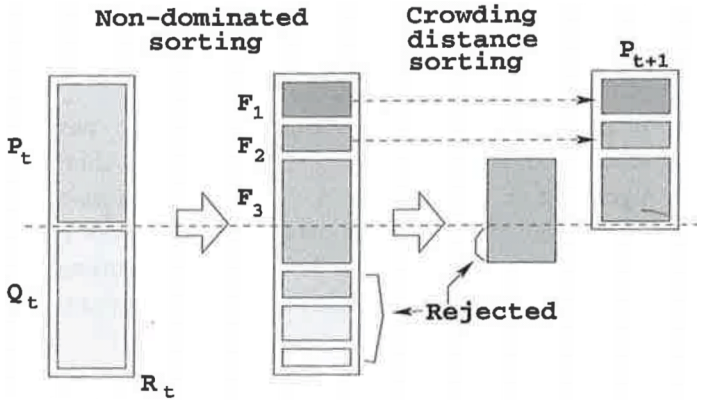


Figure 2.8: Assembling the new population after using the genetic operators. In the figure, non-dominated-sorting is used to find the best fitness fronts F_1, F_2 and F_3 and crowding-sort is utilized to exclude from population P_{t+1} the solutions in F_3 that have the worst crowding distance values. Reprinted from Deb (2001).

been created using the parent population P_t of n individuals with the genetic operators, the populations are combined together to form R_t of size $2n$. Combining the parent population with the offspring population at this point preserves elitism in R_t. The non-dominated-sorting procedure is called to classify the population R_t into fitness fronts F_i. After this, the new population P_{t+1} of size n is filled by the solutions of different non-dominated fronts (that have a total of $2n$ individuals), starting from the best front. When the the last front allowed in the population is considered, crowding-sort is employed to identify the solutions on that front which are in the least crowded region. They are added into the population and the others discarded.

2.4 Radial Basis Function Network

Radial basis function (RBF) networks belong to the class of artificial neural networks (ANNs) and, according to Kriesel (2007), are based on the idea that any function can be approximated with an arbitrary accuracy by summing together spatially shifted, stretched and compressed multidimensional Gaussian functions. In this section, the general theory of RBF networks is presented according to Kriesel (2007) and Haykin (1999).

RBF networks are built of layers of neurons and weighted connections between them like other artificial neural networks. A neuron is a data processing unit which takes as input data signals and outputs different data signals. It is comprised of the propagation, activation and output functions. The propagation function is a function that collects the inputs to the neuron and transfers them to the activation function. The activation function determines the activation of the neuron depending on the strength of the signal received from the propagation function. The output function transforms the activation signal to output. A schematic presentation of an artificial neuron is shown in Figure 2.9.

An RBF network has exactly three layers of neurons in the following order: The input layer consisting of input neurons, the hidden layer or the RBF layer consisting of RBF neurons and the output layer consisting of output neurons. Each layer is completely linked with the following one. The number of the input neurons equals the number of inputs and the number of the output neurons the number of outputs. The number of the RBF neurons is defined by the user. A schematic presentation of an RBF network and its layers is shown in Figure 2.10.

The input neurons forward the information they receive and they have the identity

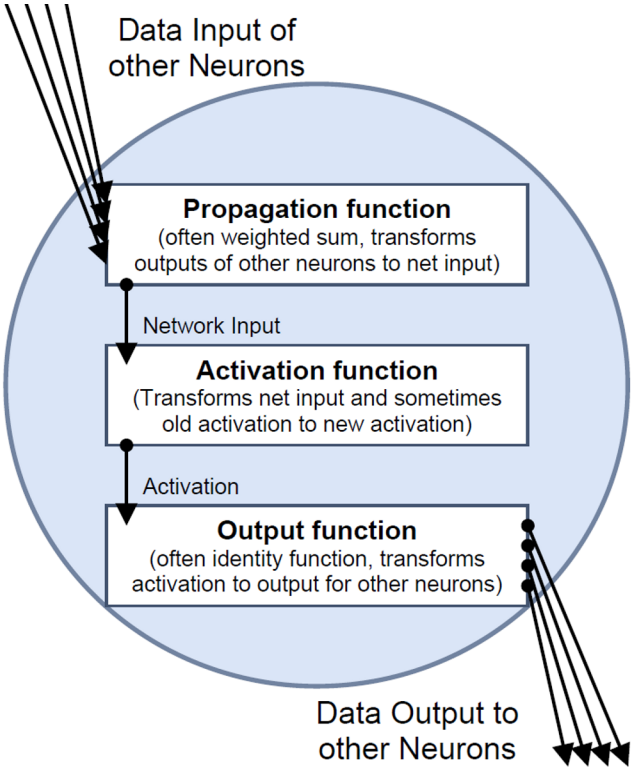


Figure 2.9: An artificial neuron. Propagation, activation and output functions are used to process the input to the neuron and transform it to output. Reprinted from Kriesel (2007).

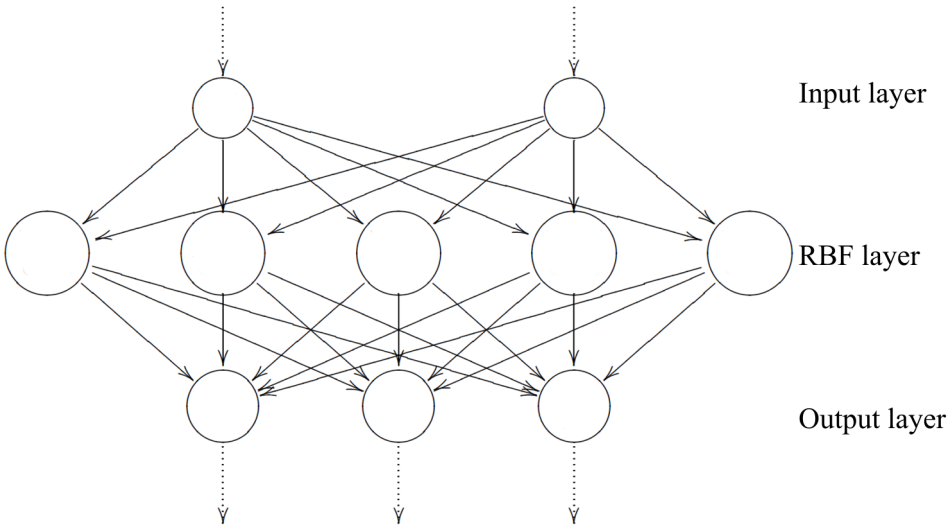


Figure 2.10: An RBF network. The number of input neurons equals the number of inputs and number of output neurons the number of outputs. Adapted from Kriesel (2007).

function as the propagation, activation and output functions. There are no user defined parameters associated with the input neurons. The RBF neurons calculate as propagation function a distance based on some norm that represents the distance between the input to the network and the center of the neuron. This distance is inserted into a radial activation function which calculates the activation of the neuron. The output function is the identity function and thus the closer the input vector is to the center vector of an RBF neuron, the higher is its output. When the network is built the user needs to specify the norm and the radial activation function used. The output neurons use a weighted sum as the propagation function and the identity is used as the activation and output functions. The weights are defined in the training phase of the network so that the error of predictions on the test data is minimized.

A methodology of using RBF networks proposed by Duvigneau and Visonneau (2004) is used in the learner subprogram that is developed in this work. The novelty of the approach is that RBF networks are trained locally using only a number of the nearest database entries every time an approximate evaluation is called for. In the method, the hidden layer consists of as many neurons as there are local database entries solved by CFD. The approach is implemented as described in Duvigneau and Visonneau (2004) and presented in Appendix B.

The idea of local training is based on the realization that farthest points in the global database may have a bad influence when evaluating an individual locally. Using a local database also helps against the possible problem of overfitting. The local database size d determines how many entries from the global database are used during the training process. If d is too large, points too far away may be included and if it is too small, the approximation may be poor.

2.5 CFD-Optimization Program

The CFD-optimization program developed in this work is presented in this section, which ties together all the concepts presented in the previous sections of this chapter. The general program construction is outlined and the subprograms and their interfaces are introduced. The operation of the program is explained along with the uncoupled and coupled methods for using it. Results that can be expected with both methods are also discussed.

2.5.1 Construction of the CFD-Optimization Program

The optimization program consists of the optimizer, learner and CFD preprocessor subprograms and it is integrated with an existing recovery boiler CFD model. The program is developed in this work to allow for a good integration between the CFD model and the subprograms and to provide possibilities for customization. Another reason why the program is developed is that it can be expanded in future work to handle different optimization tasks. The program construction is presented in Figure 2.11. The variables and functions mentioned in the figure are all vectors.

The optimizer subprogram takes as input values of objective and constraint functions at requested design points. It gives as output new design points, which should move gradually closer to the Pareto front as the generations advance. The optimizer subprogram uses the elitist non-dominated sorting genetic algorithm (NSGA-II) to search for the front in the objective space. It is initialized with an initial population of individuals each representing a different design point and evolves the population towards the Pareto front using evaluation information from the learner and CFD model. At the beginning of each new generation of individuals, it asks the learner for evaluations and then assigns fitness values to each of the individuals. The fittest individual, according to crowded selection among the individuals, is sent first to the CFD preprocessor and then to the CFD model. The CFD model provides a solution for the individual, which is used to update its evaluation. After this, the NSGA-II

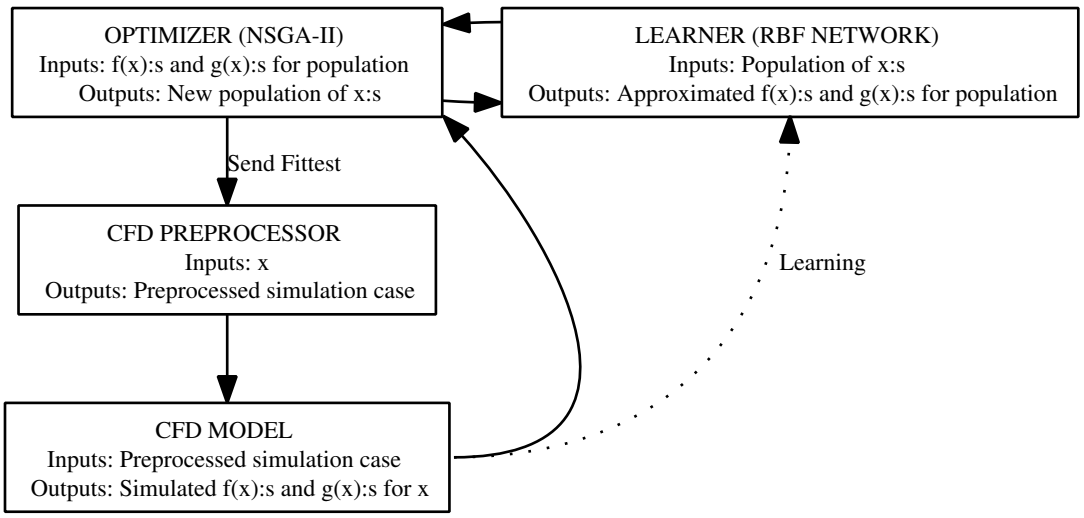


Figure 2.11: The CFD-optimization program construction and interaction between the subprograms. Inputs and outputs of each subprogram are listed below their names. The design point vector $\vec{x}$ is indicated as x, vector of objective functions $\vec{f}(\vec{x})$ as f(x) and vector of constraint functions $\vec{g}(\vec{x})$ as g(x).

algorithm checks for convergence and then, if needed, continues and uses the genetic operators to advance to a new generation of individuals.

The learner takes design points as input and returns values of the functions at these design points. It can also take as input a design point and values of the functions at this point. The learner adds this new information to its database and learns from it. The learner subprogram is constructed to use the radial basis function (RBF) network to provide a fit for each function of the design variables. It is trained with initial training data and then learns from each CFD evaluation performed during the program run. The objective of the learner is to generalize from the data and to interpolate good approximations for the functions in previously unseen design points.

The CFD preprocessor takes a design point as input and sets up a CFD simulation for it. It gives as output a CFD simulation case that is ready for solving in parallel computation. The CFD preprocessor builds a recovery boiler geometry according to what programmed design rules dictate for the given design variables. After this, the preprocessor builds a computational grid for this geometry using an automated method utilizing ANSYS DesignModeler and Meshing programs (ANSYS, 2011c, 2011a). Simulation settings and boundary conditions are then found for the case and the computation is set up in the flow solver. The fields of all the solved variables are initialized with a converged simulation of the closest design point by an Euclidean distance metric in the design space $\mathcal{D}$. This provides a good initial guess for the simulation and speeds up convergence considerably. After the setup is complete, the case is sent to the CFD model.

The CFD model takes as input a set up simulation case and outputs values for the functions. The CFD model solves the simulation case using ANSYS Fluent 14.0 (ANSYS, 2011b). The simulation is run until convergence or until an upper limit of iterations is reached. If the case does not converge, it is labeled as an unfeasible design point. This means that objective and constraint functions can not be evaluated at this design point and that it is removed from the feasible space. If the case converges then function values are extracted from it using postprocessing routines. The obtained information is sent to learner and optimizer subprograms in both cases.

2.5.2 Operation of the CFD-Optimization Program

The first step in using the CFD-optimization program is building a training database for the learner by evaluating a number of individuals by CFD. After the training database has been built, the optimization process can be started. During the optimization process no more CFD evaluations are done (in the uncoupled method) or one evaluation is done at the most suitable design point each generation of the genetic algorithm (in the coupled method).

In this work, the training database is built using an approach motivated by the theory of design of experiments (DOE). Design points are taken from the design space $\mathcal{D}$ by random, but in such a way that their Euclidean distances from each other have some minimum value. When no more points can be inserted into $\mathcal{D}$ with a given minimum distance, the distance is reduced. This is repeated until the wanted number of points has been inserted into the database. This procedure ensures that there is a good global coverage of the whole design space in the database. However, it is not guaranteed that local coverage (i.e., the density of the coverage) is good enough. Local coverage can be made better only by inserting more points into the training database. The suitable total number is dependent on the dimensionality of the design space and on the properties of the constraint and objective functions.

Another similar approach for training, that is suggested by Georgopoulou and Giannakoglou (2009), is to start with an empty training database and to run the GA the first couple of generations using only CFD evaluations, and only after that use the learner for the majority of evaluations. This method tries to focus the CFD evaluations to the most interesting areas from early on, but it does not ensure a good global coverage of the whole design space.

Recommendations for the parameters in the algorithms are shown in Table 2.5. The recommendations for the values are given according to literature, if such information is available. In general, parameters of GAs are somewhat problem specific and are often selected using heuristic rules. GAs are robust and work well with most sensible parameter settings.

The population size n_{ind} of the GA needs to be adequate for covering the design space in the beginning, for maintaining diversity in the population and for presenting the Pareto optimal front. A very large population size increases computational resources needed and might inhibit convergence. Alander (1992), among others, has studied choosing the population size in an optimal way and has found that the optimal population size depends on the features of the problem. In the present optimization

Table 2.5: The program parameters for the optimizer (NSGA-II) and learner (RBF network) subprograms and their recommended values.

Parameter	Symbol	Value
NSGA-II		
Population size	n_{ind}	50 to 250
Generations	-	at least 50
Decision variable string length	b	at least 7
Crossover probability	p_c	0.9 to 1.0
Mutation probability	p_m	$1.0/(bn_{dv})$
RBF network		
Initial database size	-	at least 50
Attenuation coefficient	e	10 to 100
Local database size	d	10 to 30

problem, where information about the objective and constraint functions is scarce, the population size needs to be chosen heuristically. The number of generations run is usually limited by computer time available. In this work, it is hypothesized that after 50 generations the Pareto front should already be visible. Decision variable string length b is chosen in general so that the wanted resolution is obtained for each design variable. Choosing an excessively large string length will slow down convergence. Crossover and mutation probabilities are chosen similarly as in Deb et al. (2002). A high crossover probability p_c is recommended because the algorithm is elitist and the best parent solutions are always included in the population. A mutation probability $p_m = 1.0/(bn_{dv})$, where n_{dv} is the number of decision variables, is used because then on average one mutation happens in each offspring per generation.

The initial database size of the learner should be as large as possible so that the algorithm has enough information to make good approximations, but it is usually limited by computational resources available. A size of 50 entries is estimated to be a practical minimum value in the three dimensional design space of the present problem. Duvigneau and Visonneau (2004) have studied the effects of different attenuation coefficients and local database sizes when using RBF networks for approximate evaluations in wing profile optimization. According to them, local database sizes from 10 to 30 and values of e from 10 to 100 are suitable for a wide range of situations. Georgopoulou and Giannakoglou (2009) are also advising to use similarly sized local databases.

In this work, the furnace geometry optimization problem is solved using two methods: an uncoupled method utilizing mainly approximate evaluations and a coupled method concurrently using both approximate and CFD evaluations. The methods can be thought of as two different ways of operating the developed CFD-optimization program.

Methods similar to the the uncoupled method are often included in optimization toolboxes available for CFD models. In this approach, there exists no connection between the optimization algorithm and the flow solver. The optimization is done based on global approximations built according to the evaluations in the training phase. This means that after training the program is run to convergence without performing any more CFD evaluations. In many situations, the global approximations might be locally inaccurate and this can easily result in the algorithm converging to a false optimum. This is especially true when the dimensionality of the design space is high, because then a large number of training points is needed to adequately cover it. The method is included in this study because it is widely used and computationally inexpensive.

In the coupled method, the learner is trained similarly as in the uncoupled method and in addition CFD evaluations are performed during each optimization cycle as requested by the optimization algorithm. In this work, the program is set to request the CFD evaluation of the fittest individual after each generation of the genetic algorithm. The knowledge of the learner is updated based on these performed evaluations. In this approach, a coupling is maintained and evaluations can be focused on the most promising areas of the design space. This process should lead to a continuously updated and more accurate solution.

Chapter 3

Results and Discussion

In this chapter, the results of the work done are presented and discussed. Before the developed CFD-optimization program is used for the recovery boiler furnace geometry optimization task it is essential to study errors involved in the simulations and to verify the developed program. The iteration and discretization errors involved in the simulations are discussed, because the used optimization methods rely on the accuracy of the provided CFD evaluations. The CFD-optimization program is verified in test problems and the obtained results are compared to exact solutions. It is made sure that the algorithms work as intended and that their performance is similar to the performance of the implementations found in literature. The furnace geometry multi-objective optimization task was solved using the uncoupled method and the coupled method. The results of the methods are compared against each other and to the performance of the base design used as the starting point. It is also discussed how the results can be utilized in practice.

3.1 Computational Model Error Estimation

Before using the computational model for the optimization task, its reliability is assessed by studying iteration and discretization errors. Furthermore, knowing the magnitudes of errors involved is important because the optimization methods employed in the furnace geometry optimization assume that accurate CFD evaluations are available. The main interest is to study discretization error by doing a grid refinement test, but the effect of iteration error can not be separated from the results of the discretization error study. It is essential to know the scale of the iteration error so that the results of the grid refinement study can be interpreted. Assessing

the magnitudes of errors is also important in general so that the trustworthiness of the simulation results can be analyzed.

3.1.1 Iteration Error Estimation

The iteration error study was done by running a CFD simulation of the recovery boiler base design described in Subsection 2.1.1 using a grid of 530 000 cells. The simulation was run until convergence was manually detected at 19 000 iterations and after this the iterations were continued for 21 000 iterations more. Mean values of temperature, CO content, O_2 content and carryover were tracked on the nose level in this converged state. Iteration errors are assessed by examining the minimum and maximum values of these quantities along with the minimum and maximum values of their running averages in the converged iteration range. Behavior of the mean profiles of temperature, y-velocity, CO content and O_2 content in the converged range is also examined.

The mean profiles of the monitored quantities as functions of boiler height are plotted between iterations 24 000 and 40 000 at intervals of 4 000 iterations in Figure 3.1. It can be seen in the figure that there exists significant variation in the profiles of the plotted quantities as functions of iterations. The mean temperature profile seems to stay relatively constant during the iterations and differences between the profiles shown are less than approximately 25 °C everywhere. It should be noted that there is a clear difference in the profiles at the nose level. In the velocity profiles it can be seen that the mean velocity is even more insensitive to the iteration error than the mean temperature. There is only slight variation in the profile as a function of iterations.

The mean O_2 profile in Figure 3.1 exhibits considerable variation when the iterations advance, especially in the area above the second tertiary air level. At the nose, the largest differences in values given by the profiles are approximately 0.5 wt%. Differences this high signify that the computed O_2 contents on the nose are substantially affected by the iteration error. In the mean CO profiles it can be seen that the iteration error affects these profiles significantly and that the largest differences in these profiles are more than 10 000 ppmw. The height coordinate value where the CO content goes close to zero varies considerably between these profiles. This coordinate value represents the point where all CO has combusted and it should be below the nose level. The point moves between below and above the nose level as a function of iterations. This affects the computed CO content on the nose level significantly.

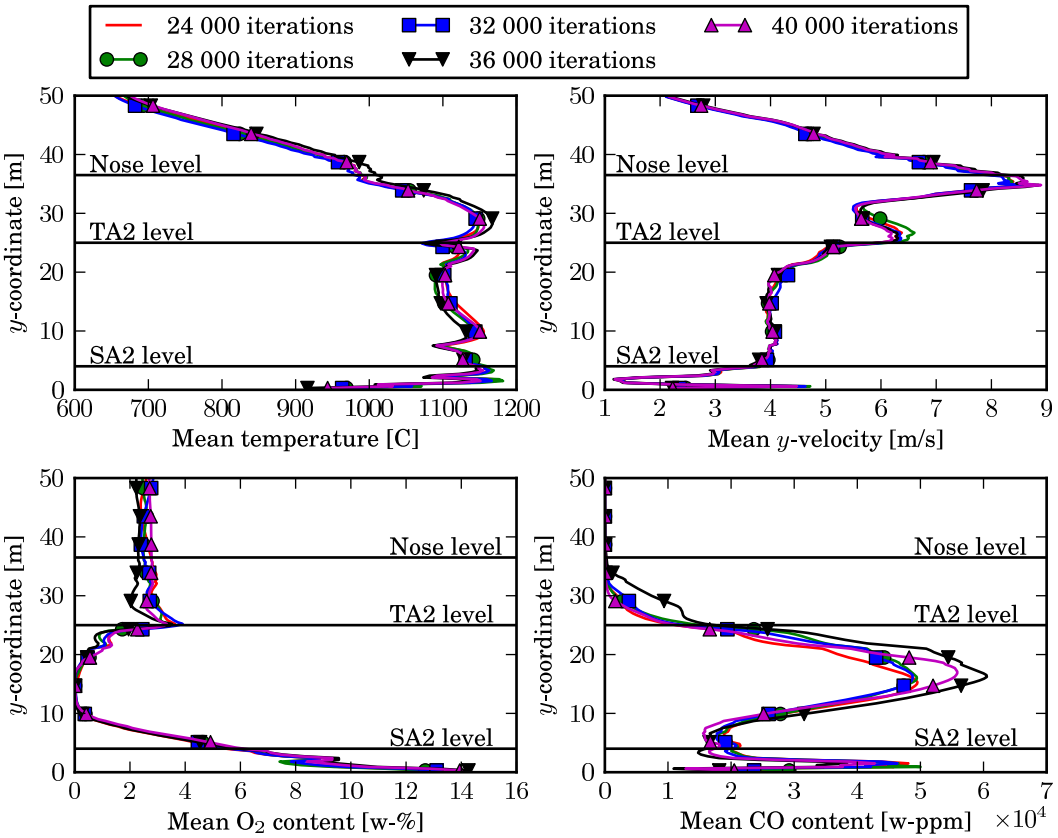


Figure 3.1: Profiles of the monitored quantities after different numbers of iterations. The nose level and the most important air injection levels are marked in the figure. The abbreviation SA2 refers to the upper secondary air level and TA2 to the upper tertiary air level.

Values of the monitored variables from the same iteration counts as the profiles in Figure 3.1 as averages over the last 5 000 iterations are shown in Table 3.1. It can be seen that there exists clear variation in the values depending on the iteration count at which they are taken out from the simulation. The values of the nose temperature, carryover and O_2 content show some variation but are relatively similar between the counts. On the other hand, values of the CO content show large differences. No definite conclusions can be drawn from the table because it shows the values of the quantities only at a small number of different iteration counts, but it clearly shows that the iteration error in the CO value is substantial.

Minimum and maximum values of temperature, CO content, O_2 content and carryover along with minimum and maximum values of their running averages (RA) over 5 000 iterations at nose level between iterations 24 000 and 40 000 are shown in Table 3.2. The simple minimum and maximum values show that the iteration error dominates the values taken from a single iteration and that it is hard to use them

Table 3.1: Values of the monitored quantities after different numbers of iterations as iteration averages over the last 5 000 iterations. The abbreviation i. refers to the number of iterations.

Quantity	24 000 i.	28 000 i.	32 000 i.	36 000 i.	40 000 i.
Temperature at nose [°C]	998	992	1001	990	998
CO content at nose [ppmw]	327	404	564	605	1111
Carryover at nose [kg/s]	0.10	0.11	0.11	0.10	0.11
O_2 content at nose [wt%]	2.4	2.5	2.4	2.6	2.4

for any meaningful analysis. The minimum and maximum values of the running averages exhibit less variation than the simple minimum and maximum values. The averaging has succeeded in filtering out a part of the iteration error and presumably the largest spikes in the values in the iteration history. The RA values of the nose temperature, carryover and O_2 have only slight variation but the CO content still fluctuates considerably.

To conclude the observations of the iteration error study it can be said that, even when convergence of a simulation is observed, depending on the iteration count where the computation is stopped there exists considerable variation in the values taken out and sometimes even large spikes. Iteration averaging of the values seems to help in this problem and filters out some of the iteration error. The values obtained in this way can not be said to be accurate values obtained by a CFD simulation but they are considerably better estimates than values of a single iteration and can be used for comparing results against each other. In the conducted simulations, it was seen that the residuals of the solved variables settled around some levels quite early in the iterations and did not substantially change in magnitude even if the iterations were continued to high counts. Also, at this point the values of the quantities solved on the nose level did not move to any direction but settled to oscillate around some

Table 3.2: Minimum and maximum values of the monitored quantities and their running averages over 5 000 iterations between iterations 24 000 and 40 000.

Quantity	Min.	Max.	Min. RA	Max. RA
Temperature at nose [°C]	962	1042	990	1003
CO content at nose [ppmw]	1	7554	183	1210
Carryover at nose [kg/s]	0.04	0.21	0.10	0.11
O_2 content at nose[wt%]	1.3	3.3	2.4	2.6

values. It can be deducted from these observations that the iteration errors are not reduced if iterations are continued after convergence has been observed.

Ferziger and Perić (2002) mention that iteration, discretization and modeling errors should each differ from each other by an order of magnitude, the iteration error having the smallest value. When the magnitudes of the errors are related to each other in this way, they can be attempted to be separated from each other. It is observable from the results that this is not possible in the context of this model because the iteration error is so large. If discretization errors were a magnitude larger than the iteration error the solutions would be very poor. It is clear that with an iteration error this substantial, it is not possible to accurately assess the magnitude of the discretization error.

3.1.2 Discretization Error Estimation

The discretization error study was done by computing CFD simulations of the recovery boiler base design described in Subsection 2.1.1 using three progressively denser grids. Mean profiles of temperature, y-velocity, CO content and O_2 content obtained using the different grids are compared along with the solved values of temperature, CO content, carryover and O_2 content on the nose. Results obtained with the grids are analyzed using the reporting conventions based on the grid convergence index (GCI) and S values, as described in Subsection 2.1.3.

The variables examined in the grid refinement study were chosen to be temperature, CO content, O_2 content and carryover on the nose level, taken as averages over the last 5 000 iterations to filter out some of the iteration error. GCI values were computed for each of these variables on all of the grids to examine their convergences individually, because it is known from experience that different variables might converge after different amounts of iterations. In theory, the GCI values of the base grid of 530 000 cells can be used as estimates for the discretization error bands of the solutions obtained when the grid is used in the geometry optimization. Values of S were also calculated for the variables to examine whether the solutions are in the asymptotic range.

In the discretization error study, a low density grid (LDG) with 530 000 cells was used as a starting point. An intermediate density grid (IDG) was obtained by refining the base cells of the low density grid using a refinement ratio of $r = 1.2$. This means that approximately 20% more cells were used in each direction. A high density grid (HDG) was obtained by refining the intermediate density grid in the

same way, again using $r = 1.2$. Effective grid refinement ratios r_{eff} were computed using Equation (2.6). The information of the grids used is summarized in Table 3.3.

The computations for the discretization error study with the different density grids were run until the convergences were observed manually. Mean profiles of temperature, y-velocity, CO content and O_2 content at convergence plotted as functions of boiler height with different grids are shown in Figure 3.2.

It can be seen in Figure 3.2 that there is considerable variation in the profiles obtained with the different grids. When comparing these profiles with the profiles in Figure 3.1, obtained in the iteration error study, it can be noted that the size of the variation in the temperature and velocity profiles is of comparable magnitude in both studies. If there is variation in these profiles because of the grid it might be masked by the iteration error. The O_2 profiles obtained with different grids are markedly different. Their differences are larger than the variation in the profiles obtained from the iteration error study below the second tertiary air level. The CO profiles also have differences that are larger than in the iteration error study. It is interesting that the CO profiles go to zero near the same y-coordinate value and thus give similar values at the nose level.

Values of the monitored quantities obtained with the different grids at convergence as averages over the last 5 000 iterations are shown in Table 3.4. The values in the table with different grids are clearly different. The value of the nose temperature obtained with the high density grid is somewhat higher than with the other grids. It is also outside the iteration error band obtained for the nose temperature in the iteration error study in Table 3.2. This might indicate that the accurate value of the nose temperature is higher than the value obtained with the low density grid. Also, the value of the CO content obtained with the highest density grid is smaller than the values obtained with the other grids and it is also clearly outside the iteration error band obtained in the iteration error study. This can be an indication that the

Table 3.3: A summary of the features of the grids used. The low density grid is abbreviated as LDG, the intermediate density grid as IDG and the high density grid as HDG.

Quantity	LDG	IDG	HDG
Total number of cells	530 000	770 000	1 140 000
r_{eff}	-	1.135	1.139
Largest cell [m]	1.200	1.000	0.833
Smallest cell [m]	0.075	0.063	0.052

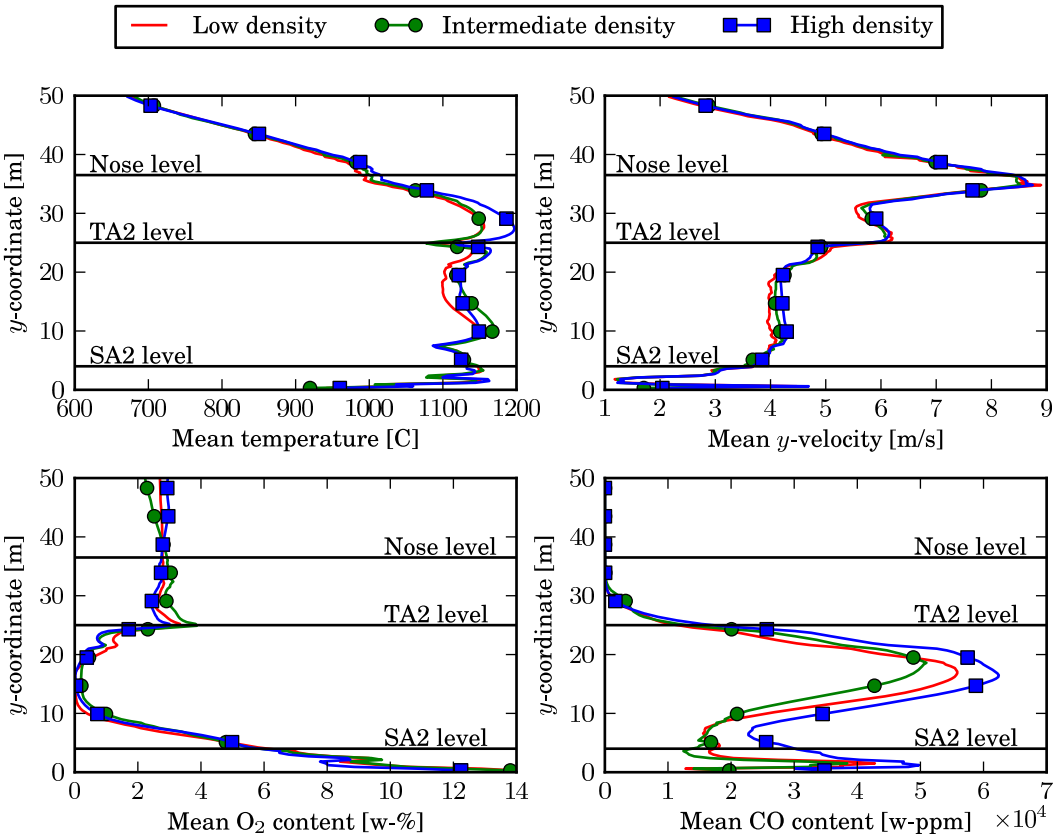


Figure 3.2: Profiles of the monitored quantities on different grids. The nose level and the most important air injection levels are marked in the figure. The abbreviation SA2 refers to the upper secondary air level and TA2 to the upper tertiary air level.

accurate CO value is lower than the value obtained with the low density grid. The carryover and O_2 values shown in the table are inside the iteration error bands or very close to them.

Using the results obtained with the different grids, GCI and S values were computed for each solved variable. The intermediate density and high density grids were inspected first and GCIs values for the quantities were calculated using Equation (2.2) with $F_s = 1.25$. A value of $p = 2$ was used because second-order discretization methods are employed in the model. After this, GCI values for the low density grid were calculated from Equation (2.3). Using the intermediate and fine grid GCIs and an averaged value of r_{eff} in Equation (2.5), S values were obtained for assessing whether the solutions are in the asymptotic range. The values were converted to percentages for clarity and they are summarized in Table 3.5.

According to the theory of Richardson extrapolation, the GCI values of the grids, in Table 3.5, should go down as the grid is refined in the asymptotic range. Also, the

Table 3.4: Values of the monitored quantities on the grids of different densities. The low density grid is abbreviated as LDG, the intermediate density grid as IDG and the high density grid as HDG.

Quantity	LDG	IDG	HDG
Temperature at nose [°C]	998	997	1012
CO content at nose [ppmw]	1111	538	53
Carryover at nose [kg/s]	0.11	0.12	0.11
O_2 content at nose [wt%]	2.4	2.5	2.5

Table 3.5: The GCI and S values of the monitored variables on the different grids. The GCI is an estimate for the error band of a solved variable on the grid used. Values of S close to 0% indicate that the solutions are in the asymptotic range. The low density grid is abbreviated as LDG, the intermediate density grid as IDG and the high density grid as HDG.

Quantity	LDG GCI [%]	IDG GCI [%]	HDG GCI [%]	S [%]
Temperature at nose	0.6	0.4	6.2	94.6
CO content at nose	595.0	461.9	3848.7	90.7
Carryover at nose	34.0	26.4	60.0	66.0
O_2 content at nose	23.3	18.1	0.2	98.8

values of the quantities given in Table 3.4 should be moving towards some values when the grid is refined. Even if this behavior takes place, it can not be detected from the results because the values obtained for the quantities (Table 3.4) are inside the iteration error bands obtained in the iteration error study (Table 3.2). Furthermore, the reporting conventions based on the GCI and S values assume a zero iteration error. Because of this, the iteration errors in the results obtained with the different grids are in the conducted analysis incorrectly labeled as discretization errors in the computed GCI and S values.

The depicted problems are evident in the GCI and S values obtained in Table 3.5. The computed GCIs are in general very high and the S values are far away from 0%. Because the S values are not close to 0%, the GCI values can not be considered as valid error bands. The high S values signify that either the grids are not in the asymptotic range or, which is more likely, the results are affected by the iteration error. Additionally, the assumption of $p = 2$ for the convergence order of the code might not be valid. The formal convergence order of the used discretization methods and the observed convergence order of the code might not always be equal. This can result from, for example, a numerical implementation of boundary conditions

with a lower formal accuracy or programming errors. Further examination of this is outside the scope of this work. Because of the aforementioned reasons, no conclusions regarding the magnitudes of discretization errors can be drawn from the obtained GCI and S values.

As a final note concerning the study, it can be said that no definite conclusions can be made concerning the sizes of the discretization errors based on the conducted analyses. According to the results of the iteration error study, the iteration errors are so large that discretization errors can not be reliably separated from the simulation results when the grid is refined. Furthermore, in Figures 3.1 and 3.2 it can be seen that even though some variables are relatively insensitive to the iteration error, their variations during the grid refinement is too small for adequately separating the errors from each other. The usage of the iteration averages for the quantities helps to mitigate this concern, but also using larger grid refinement ratios than in this study might show the discretization errors more clearly. On the contrary, it might also be possible that without considerably reducing the iteration errors first, the evaluation of the discretization errors is not practically achievable. No indication was found in the study that using a higher density grid in the geometry optimization than was originally planned would be necessary.

3.2 CFD-Optimization Program Verification

Before the CFD-optimization program can be used for optimizing the recovery boiler furnace geometry it has to be made sure that it works as intended. The optimizer and learner subprograms are used in test problems and their performance is compared to exact solutions of the problems. If the performance of the subprograms is found accurate and robust it can be said that the implementations of the algorithms are verified.

3.2.1 Optimization Algorithm Verification

The optimization algorithm was tested by running the optimization program using only exact objective and constraint function evaluations. This means that the learner subprogram was not used at all. The parameter settings of the algorithm were set to similar values that were later used when the geometry optimization was run. This was to ensure that the parameter settings work and that they are also verified to a degree. Two test problems taken from earlier studies on genetic algorithms (GAs) were used. The behavior of the population in the test problems is analyzed

and it is inspected how well and how fast the population converges to the Pareto front. Furthermore, diversity of the front is analyzed and it is checked that elitism is preserved correctly.

The first test problem, called CONSTR, is a case that was used by Deb et al. (2002) in their original paper on the elitist non-dominated sorting genetic algorithm (NSGA-II). In this problem, a part of the unconstrained Pareto optimal front is not feasible and the constrained Pareto front is partly on the first constraint boundary. The CONSTR problem is given as

$$\begin{aligned} \min \quad & f_1(\vec{x}) = x_1, \\ \min \quad & f_2(\vec{x}) = (1 + x_2)/x_1, \\ \text{subject to} \quad & g_1(\vec{x}) = x_2 + 9x_1 \geq 6, \\ & g_2(\vec{x}) = -x_2 + 9x_1 \geq 1, \\ & 0.1 \leq x_1 \leq 1.0, \\ & 0 \leq x_2 \leq 5. \end{aligned} \tag{3.1}$$

The second test problem is the TNK problem, that was first suggested by Tanaka et al. (1995) and later used by Deb et al. (2002). This problem features a discontinuous Pareto front which is completely on the first constraint boundary. The TNK problem is stated as

$$\begin{aligned} \min \quad & f_1(\vec{x}) = x_1, \\ \min \quad & f_2(\vec{x}) = x_2, \\ \text{subject to} \quad & g_1(\vec{x}) = -x_1^2 - x_2^2 + 1 + 0.1\cos(16\arctan(x_1/x_2)) \leq 0, \\ & g_2(\vec{x}) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5, \\ & 0 \leq x_1 \leq \pi, \\ & 0 \leq x_2 \leq \pi. \end{aligned} \tag{3.2}$$

The parameter settings used are given in Table 3.6. A value of $b = 10$ was used in the TNK problem because the feasible region lies in a small part of the design space. This raises the computational cost but produces a better spread of solutions on the Pareto front. Other parameter settings were chosen according to the recommendations

presented in Subsection 2.5.2.

The spread of the population in the CONSTR problem after different numbers of generations is shown in Figure 3.3, where the area of the feasible space is colored gray. The figure clearly shows how the population is moving towards the Pareto front when the generations advance. After five generations, all of the individuals are inside the constraints and some of them are already on the optimal front. Furthermore, the whole population is relatively close to the Pareto front but there are clearly areas on the front where there are no individuals. After 50 generations, the whole Pareto front is well presented. There exist small gaps in the representation of the front and some individuals have small deviations from the front. It can be seen that after 100 generations there are only few individuals left in the population which are not near the front and after 500 generations all of them are very close to it. It is interesting to note that even after 500 generations the individual closest to the upper constraint of f_2 is clearly away from it. This is caused by the random processes inherent in the search algorithm. The user might have to wait arbitrarily long until a search process happens which takes the wanted point in the desired direction.

In Figure 3.4, the population after different numbers of generations in the TNK problem is shown with the feasible space colored gray. When the performance of the program is looked at, it can be seen that after five generations the whole population is inside the constraints and clearly moving towards the Pareto optimal front. The population is very similar in the plots after 50 and 100 generations. The front is clearly visible and presented by the population but there are apparent gaps in the presentation. After 100 generations, diversity on the front is somewhat better than after 50 generations. At 500 generations, the spread of the population on the front is very good and the whole front is well presented. The discontinuities on the front are accurately described by the population at this point.

Table 3.6: The GA parameters used in the optimization algorithm verification.

Parameter	Symbol	Value
NSGA-II		
Population size	n_{ind}	100
Generations	-	500
Decision variable string length	b	7 or 10
Crossover probability	p_c	1.0
Mutation probability	p_m	$1.0/(bn_{dv})$

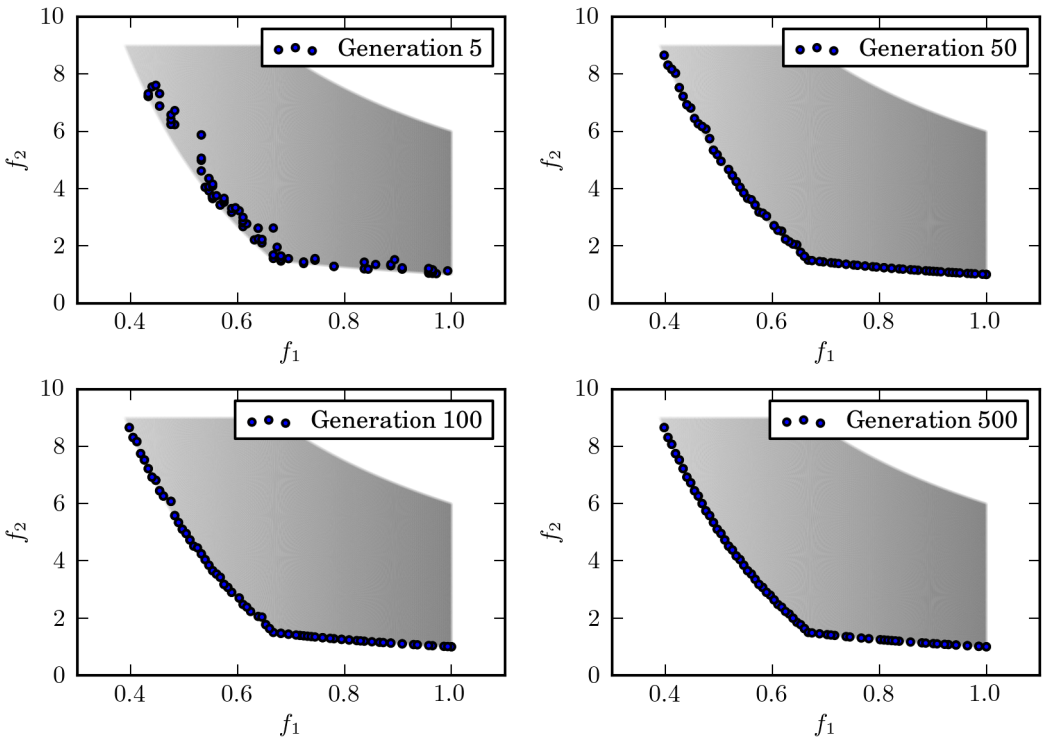


Figure 3.3: The population after different numbers of generations in the CONSTR problem when the learner is not used. The feasible space is colored gray.

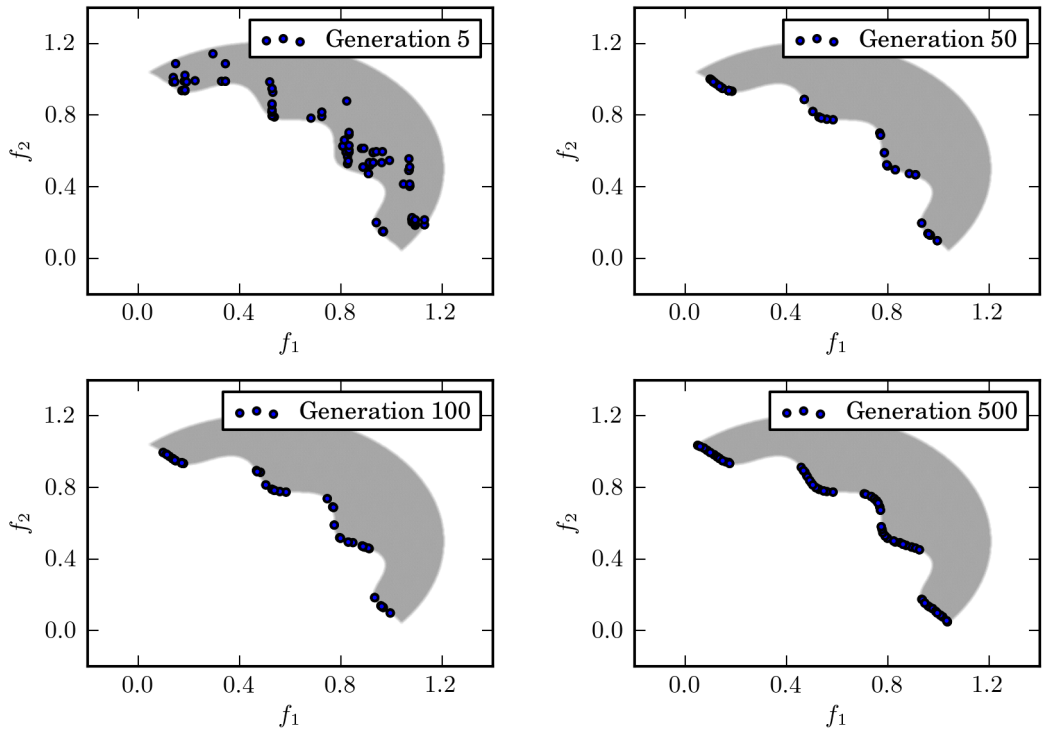


Figure 3.4: The population after different numbers of generations in the TNK problem when the learner is not used. The feasible space is indicated as gray.

As a conclusion, it can be said that the optimization algorithm clearly works as intended with the used parameters. Convergence towards the Pareto front is fast, constraints are respected correctly and diversity along with elitism is preserved. Deb et al. (2002) obtained very similar results in the test problems used with a real-coded NSGA-II algorithm.

3.2.2 Learner Algorithm Verification

The learner algorithm was tested by running the optimization program using the learner for the majority of evaluations. The CONSTR test problem given in Equation (3.1) was solved and the population was monitored as generations advanced. The population information is used to inspect convergence speed, convergence quality, diversity and elitism preservation. These are also compared to the results of the optimization algorithm verification presented in Subsection 3.2.1. Average and maximum errors of learner predictions are inspected as functions of generations to analyze how the learner performs on average and how bad the worst predictions are each generation. Contours of the objective and constraint functions as predicted at different generations by the learner are studied and by using them it is illustrated how the learner obtains and preserves information. The predicted contours are also compared to exact contours.

An initial population of 100 individuals was generated and 50 individuals chosen from it by random were evaluated exactly to train the learner. Then, after each generation an individual chosen from the best fitness group was evaluated exactly and added to the learner database. This approach corresponds well to the use of the optimization program in the coupled method geometry optimization. The parameter settings for both the GA and the radial basis function (RBF) network were chosen to be similar to the ones that were later used in the furnace geometry optimization. This was to make sure that the parameter settings for the learner are also verified to a degree. The used parameters are summarized in Table 3.7. They were set according to the recommendations given in Subsection 2.5.2.

Figure 3.5 shows the population after different numbers of generations, with the area of the feasible space colored gray. Clearly, after five generations the population is already near the Pareto front but a large number of the individuals is outside the feasible region. This is an indication that the learner might not yet accurately predict the objective or constraint functions. The spread of the solutions on the front is also bad, especially near the upper boundary of f_1. After 50 generations, the algorithm has correctly learned the functions near the Pareto front as all the

Table 3.7: The program parameters used in the learner algorithm verification.

Parameter	Symbol	Value
NSGA-II		
Population size	n_{ind}	100
Generations	-	500
Decision variable string length	b	7
Crossover probability	p_c	1.0
Mutation probability	p_m	$1.0/(bn_{dv})$
RBF network		
Initial database size	-	50
Attenuation coefficient	e	100
Local database size	d	10

individuals are inside the constraints and the majority of them is already on the front. The front is clearly visible even though its representation is in some areas inaccurate. When the generation number has advanced to 100, the population has moved closer to the optimal front and the diversity of the solutions has become better. After 500 generations, all the large gaps in the front have disappeared and the front is well presented.

Performance of the program in the CONSTR problem when using the learner, shown in Figure 3.5, can be compared to its performance without using the learner, shown in Figure 3.3. When the learner is used, the speed and quality of convergence with respect to the number of generations run are worse than when it is not used. This is evident when the populations are compared at generation five. When the learner is used, a large number of the individuals is outside the constraints whereas when the learner is not used, all the individuals are inside the constraints at this point. This behavior can result from inaccurate approximations of the functions near the constraints. At generations 50 and 100, the population on the Pareto front has a good diversity when the learner is not used. When the learner is used, there are clear gaps in the presentation. The gaps can signal that the approximations of constraint or objective functions are not yet accurate on the whole front. For example, a bad value for an objective function might be inaccurately predicted at a gap like this. An accurate evaluation is needed near the gap to correct the evaluation and to close it. After 500 generations, the program presents the Pareto front equally well when using the learner as when not using it.

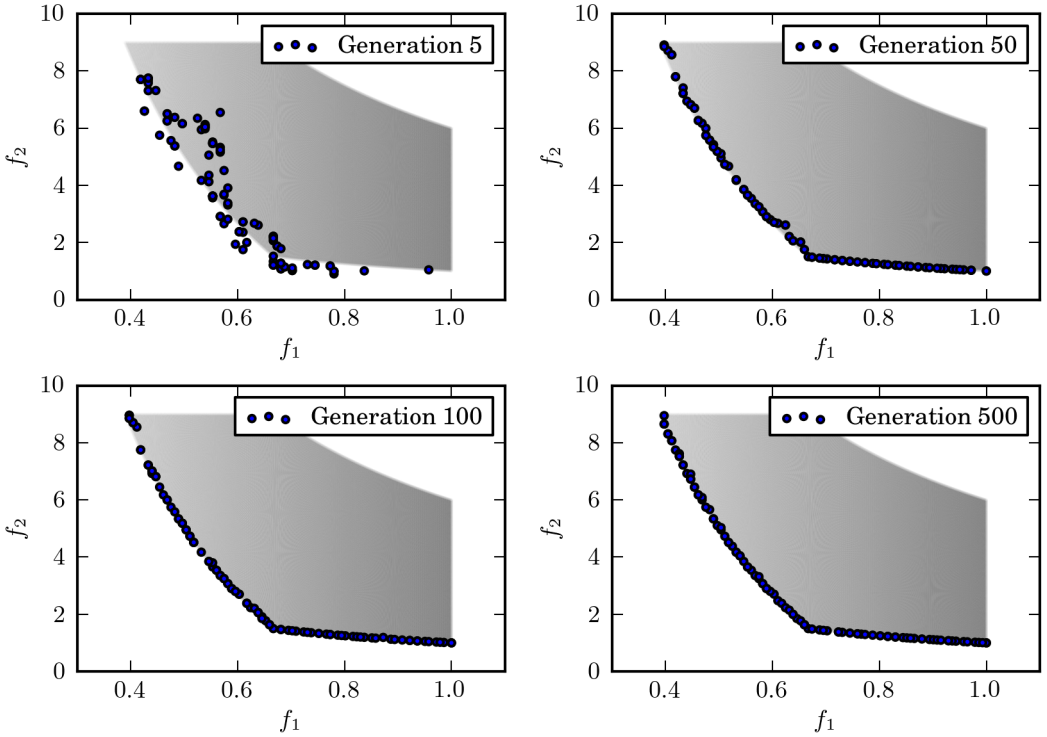


Figure 3.5: The population after different numbers of generations in the CONSTR problem when the learner is used. The feasible space is colored gray.

Average and maximum relative errors of learner predictions between generations zero and 200 are presented in Figure 3.6. After 200 generations, both errors stay close to zero. It can be seen in the figure that for the first few generations the maximum error is above 25%, but that the average error still has moderate values, ranging between 10% and 20%. At generation 20, the average error has fallen below 5% and it stays below it until the end of the optimization run. The maximum error falls below 5% after 45 generations and after that it still has spikes which sometimes take it to larger values. The spikes are caused by the random search processes of mutation and selection. These processes sometimes take the individuals to areas of the design space where the exact evaluations are scarce and the predictions inaccurate. This process expands the space searched and the spikes indicate that the program is working as intended.

Contours of the objective and constraint functions as predicted by the learner after training, at generation zero, along with the exact contours of the functions are shown in Figure 3.7. In the figure, feasible space is shown as a gray area and the design points in the database are visualized and colored by predicted function values using the same colors as in the contours. The objective function f_1 and constraint functions g_1 and g_2 are predicted by the learner very accurately already after training. The

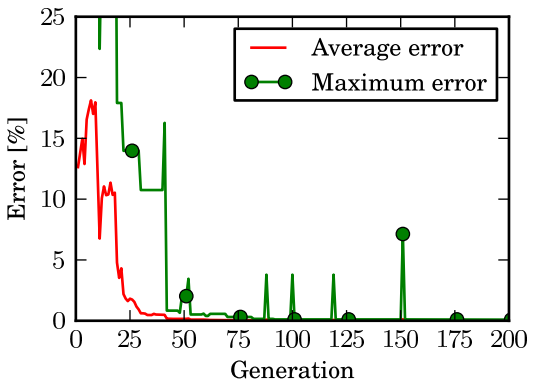


Figure 3.6: The average and maximum relative errors of learner predictions as functions of generations.

functions are linear so this is expected. The more difficult nonlinear function f_2 has quite large inaccuracies in its contours after training and the inaccuracies are clearly largest in the areas where there are few exact evaluations nearby.

Contours of function f_2 are shown at generations zero and 100 in Figure 3.8. After 100 generations, there are 100 more exact evaluations in the database and the contours are clearly more accurate near the areas where the evaluations have been done. This can be seen by comparing the contours of $f_2 = 3.5$ and $f_2 = 5.9$ at generations zero and 100. After 100 generations, the predicted contours are close to the exact contours in the most regions of the feasible space. The majority of the evaluations have been done near the Pareto front and the contours are very accurate there. The evaluations have centered to this area because the optimization has started to move to the right direction early on in the generation count.

The features of the RBF network learner are highlighted in Figure 3.8. The predictions get better in the areas where more database entries are inserted, but do not change in the other regions. Also, no signs of overtraining are present in the figures, since relatively small local databases are used. The figures furthermore illustrate the importance of good initial training. The exact evaluations after training are focused to the areas where the fitness values of the individuals are good. If bad values are predicted for the objective functions near the real Pareto front because of bad training, the program might not do any evaluations there when generations are advanced. This phenomenon results in converging to a false optimum front.

It can be concluded that the learner algorithm works as intended with the parameters used. In the test problem discussed, the optimization program converged to the real Pareto front quite fast when the learner was used and it started to respect constraints when enough evaluations were done near the boundaries of the feasible

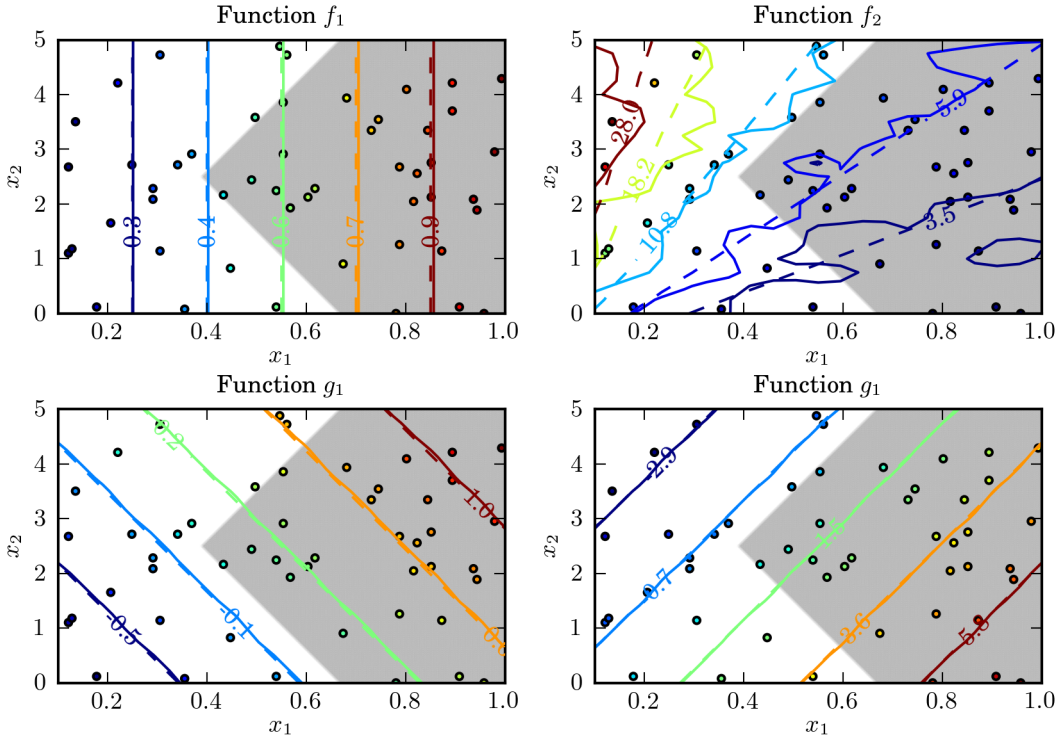


Figure 3.7: The contours of the functions as predicted by the learner (solid) and exact contours (dashed) after training (generation zero). Points in the training database are colored based on predicted function values. The feasible space is shown as gray.

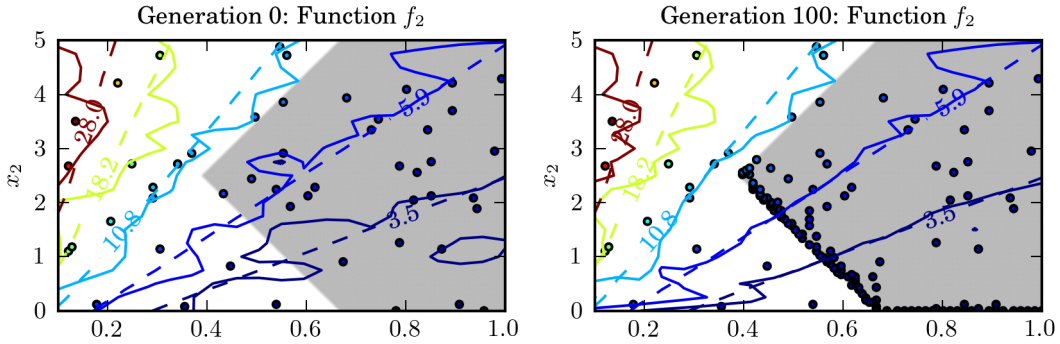


Figure 3.8: The contours of f_2 as predicted by the learner (solid) and exact contours (dashed) after training (generation zero) and generation 100. Points in the training database are colored based on predicted function values. The feasible space is indicated as gray.

space. The average error of learner predictions remained quite low as a function of generations and the maximum error obtained tolerable values. It is concluded that the optimization program can be effectively used with the learner when a good training is provided. All the fundamental parts of the complete optimization program were used in the tests discussed in this subsection and thus it can be said that the performance of the complete CFD-optimization program is essentially verified.

3.3 Recovery Boiler Furnace Geometry Optimization

The recovery boiler furnace geometry is optimized using two different methods. The results are discussed and it is illustrated how they can be utilized in practice. The first method is called the uncoupled method and in it global approximations of the functions are built using the CFD evaluations done in the training phase. After this, the optimization algorithm is run until convergence without performing any more CFD simulations. The second method is called the coupled method. When it is used, the learner is trained similarly as in the uncoupled method. In addition, CFD evaluations are performed during each optimization cycle as requested by the optimization algorithm.

3.3.1 Uncoupled Method Optimization

The furnace geometry optimization problem that was solved is defined in Section 2.2. The uncoupled method, presented in Subsection 2.5.2, was used. To examine the performance of the method, the population was monitored as a function of generations. The final population is inspected in the design space and it is discussed how different the optimum geometries are from each other. The ranges of the design variable and objective function values that the final geometries obtain are analyzed and the performance of the base design presented in Subsection 2.1.1 is compared to the performance of the solutions obtained using the uncoupled method. Additionally, four solutions from the final population are chosen for further inspection. The performance of these geometries in the objectives is discussed and the solutions are compared to each other and to the boiler base design. Finally, trade-off and concurrency relationships that can be found in the objectives on the Pareto front are discussed.

The program parameters that were used are shown in Table 3.8. The program was run for 500 generations to ensure convergence, since running the program is computationally inexpensive when no CFD simulations are done after training. The

initial database size, or training database size, was planned to be at least 50 points. All the computing time allocated for training was used and a total of 63 points was evaluated. A larger initial database might have been beneficial in the problem, but its size was limited by computational resources available. Convergence was observed manually in the simulations done for training. Other parameters in the table were set as recommended in Subsection 2.5.2 and verified in Section 3.2.

The training database and initial population are shown in Figure 3.9 along with the geometric constraints. Some points in the training database are outside the geometric constraints because they were simulated before decisions had been made concerning the exact geometric constraints. In general, this is not an issue and it can be even beneficial because it makes approximations near the constraints more accurate. The initial population of 200 individuals was filled to the desired size with random points.

The final population after 500 generations is shown in Figure 3.10. The figure shows that the obtained Pareto solutions all have a height of less than 33 m and have relatively large furnace floors. Their smallest widths are approximately 24 m, smallest depths are approximately 19 m and most of the Pareto solutions have non-square rectangular floors with a larger width than depth. Additionally, the solutions are clearly clustered in few different regions of the design space. While it might be that the real Pareto front is discontinuous, the clustering can also be an indication of convergence to a false optimum front resulting from weak training. It can be speculated that the used training database of 63 entries is too small for this

Table 3.8: The program parameters used in the geometry optimization.

Parameter	Symbol	Value
NSGA-II		
Population size	n_{ind}	200
Generations	-	500
Decision variable string length	b	10
Crossover probability	p_c	1.0
Mutation probability	p_m	$1.0/(bn_{dv})$
RBF network		
Initial database size	-	63
Attenuation coefficient	e	100
Local database size	d	10

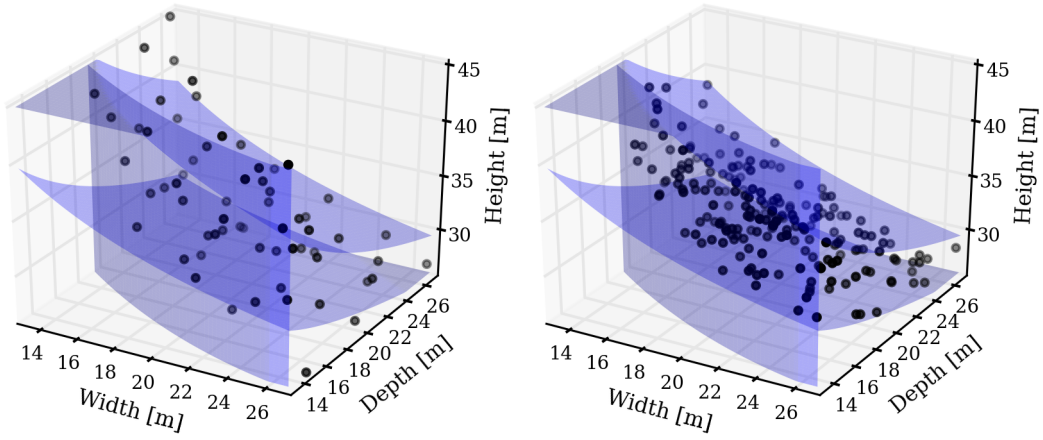


Figure 3.9: The training database and initial population along with the geometric constraints.

method.

The minimum and maximum design variable and objective function values the final population obtains are shown in a parallel coordinate representation in Figure 3.11. The design variable and objective function values of the base design are also shown for comparison. The values of the base design are based on the computations done for the analysis of Section 3.1. The ranges shown give some initial insight on the spread and performance of the obtained solutions. Visualizing all the 200 individuals, that are mathematically equally good, together is not feasible. The individual solutions

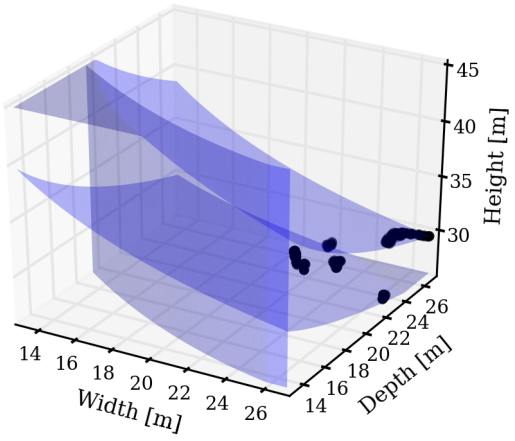


Figure 3.10: The final population obtained using the uncoupled method and the geometric constraints. The population is clustered around few different regions of the design space.

on the Pareto front obtain values that are between the bounds given in the figure. The meanings of the functions are explained in Section 2.2.

In Figure 3.11, it can be seen by looking at the minimum values that some solutions have values that round to zero in f_{carry}, f_{CO}, $f_{yVel\sigma}$ and $f_{T\sigma}$ objectives. It is possible to obtain values close to zero in the f_{carry} and f_{CO} objectives in a real boiler but it is not likely that $f_{yVel\sigma}$ or $f_{T\sigma}$ get values near zero. This would signify perfectly even upward velocity and temperature fields on the nose level. While this behavior is theoretically possible, it might also indicate that there were few CFD evaluations done in the regions where these design points are and because of this the predictions are inaccurate. If the learner inaccurately predicts very good f values in some region, the whole population starts to move in that direction. The uncoupled method is not able to correct the inaccurate predictions because it does not do any CFD evaluations after training. Individuals in the regions of inaccurate good predictions might obtain very good f values, dominate all other individuals and cause them to be removed from the population. This phenomenon results in converging to a Pareto front where the design points cluster to the areas of these

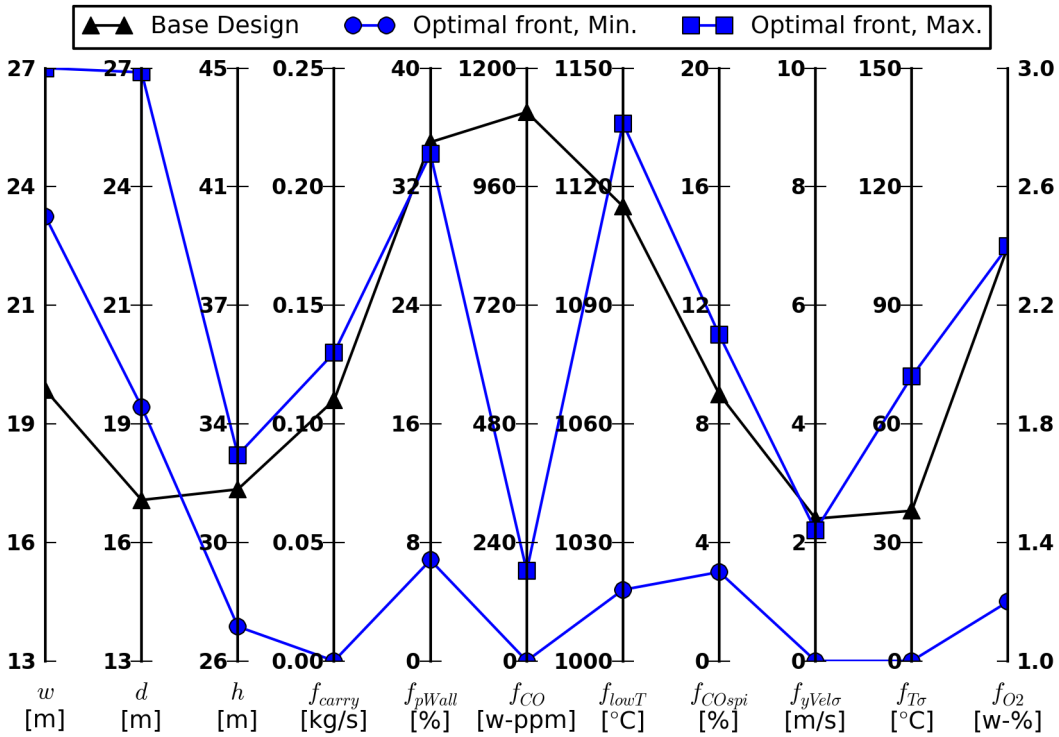


Figure 3.11: Ranges (minimum and maximum values) of the design variables and objective functions on the Pareto front obtained using the uncoupled method. The design variable and objective function values of the base design are also shown.

.

inaccurate good predictions. This problem is not present in the coupled method, because the CFD evaluations are done in the most promising areas of the design space every generation. This procedure eventually corrects the inaccurate good predictions.

Figure 3.11 also shows the boiler base design and its performance in the objective functions. It is seen in the figure that the value $f_{CO} = 1111$ ppmw does not satisfy the constraint that the CO emission should be less than 200 ppmw. This results in the fact that the base design is clearly worse than any of the obtained solutions on the Pareto front, which all satisfy the constraint. Compared to the ranges obtained on the front, the base design also has bad values in carryover, amount of particles landing on walls, upward velocity standard deviation and O_2 emission. In the other objectives, the base design has similar values as some of the solutions on the Pareto front. It is easy to find many solutions on the front which deliver better performance than the base design.

It is challenging to compare the 200 (corresponding to the population size used) obtained solutions against each other because the data is high-dimensional. After the total number of solutions has been pruned down to a more manageable size (approximately five), the performances of different design points can be compared against each other, for example, by using a parallel coordinate representation. The pruning process is discussed in more detail in Subsection 3.3.3. For the discussion of this subsection, four designs that have good overall performances were chosen from the final population for further inspection. They are not necessarily the most suitable solutions in the population for any real-life situation but they are interesting because of the values they obtain in the objectives. The objective function values of the four designs are shown in a parallel coordinate representation in Figure 3.12. The base design is also plotted for reference. The four designs have widths between 24 m and 27 m, depths between 19 m and 25 m and nose heights between 27 m and 30 m. The representation in Figure 3.12 can be used together with Figure 3.11 that gives the ranges of the objective functions on the Pareto front.

It is seen in Figure 3.12 that the designs one and four give good values in the most important f_{carry} and f_{pWall} objectives. The design one has also a low f_{CO} value and a high f_{lowT} value. In the other objectives, it has moderate values. The design four has a high CO emission, near 200 ppmw, but good or moderate values in the rest of the objectives. The design three has a relatively high carryover but a very low f_{pWall} value. Compared to the other designs, it has a low lower furnace temperature and a high value in CO spikes. In the rest of the objectives, it has moderate or good values. The design two has relatively high values in the f_{carry} and f_{pWall}

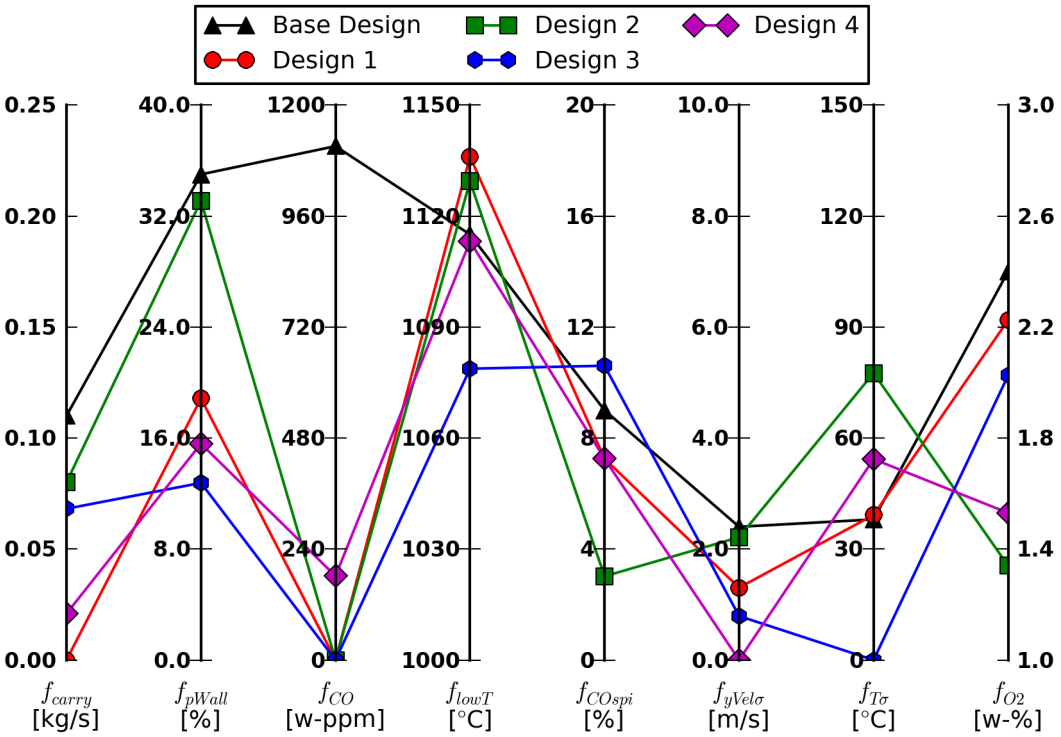


Figure 3.12: The objective function values of the base design and the four solutions chosen from the final population obtained using the uncoupled method. Each polyline corresponds to a different design point on the Pareto front.

objectives. It has good values in the rest of the objectives, expect in the $f_{yVel\sigma}$ and $f_{T\sigma}$. This discussion illustrates that trade-off decisions need to be made when the most suitable design for some particular situation is chosen. When the designs on the obtained Pareto front are compared it needs to be chosen which objectives are favored over others.

A scatterplot matrix representation of the objective function values of the final population is shown in Figure 3.13. The figure shows pairwise two-dimensional projections of the eight-dimensional data. Some solutions might look dominated in the plots but all the solutions are equally good when all the dimensions are taken into account. The scatterplot matrix is useful for finding trade-offs between the objectives. If a two-dimensional plot shows a clear front where one objective gets better when the other gets worse then there is a trade-off relationship between them. On the other hand, if no front is visible then the relationship between the objectives is either more complex or there is none. The objectives can also be concurrent. This means that when one of them gets better values the other one also does. This is a signal that the dimensionality of the optimization problem could possibly be reduced by including only one of them. It should be stressed that the relationships found

can not be said to be mathematical correlations between the objective functions. They have relevance only with respect to these particular results. The trade-off and concurrency relationships can be utilized when improvements are desired in some particular objective.

In Figure 3.13, a trade-off can be identified between the carryover and the amount of particles landing on the walls (row one, column two). It is not evident why there is a trade-off like this. It might be that when the geometry is changed so that the amount of particles landing on the walls is reduced, it causes flow patterns in the furnace which raise the total carryover. There is a concurrency between the

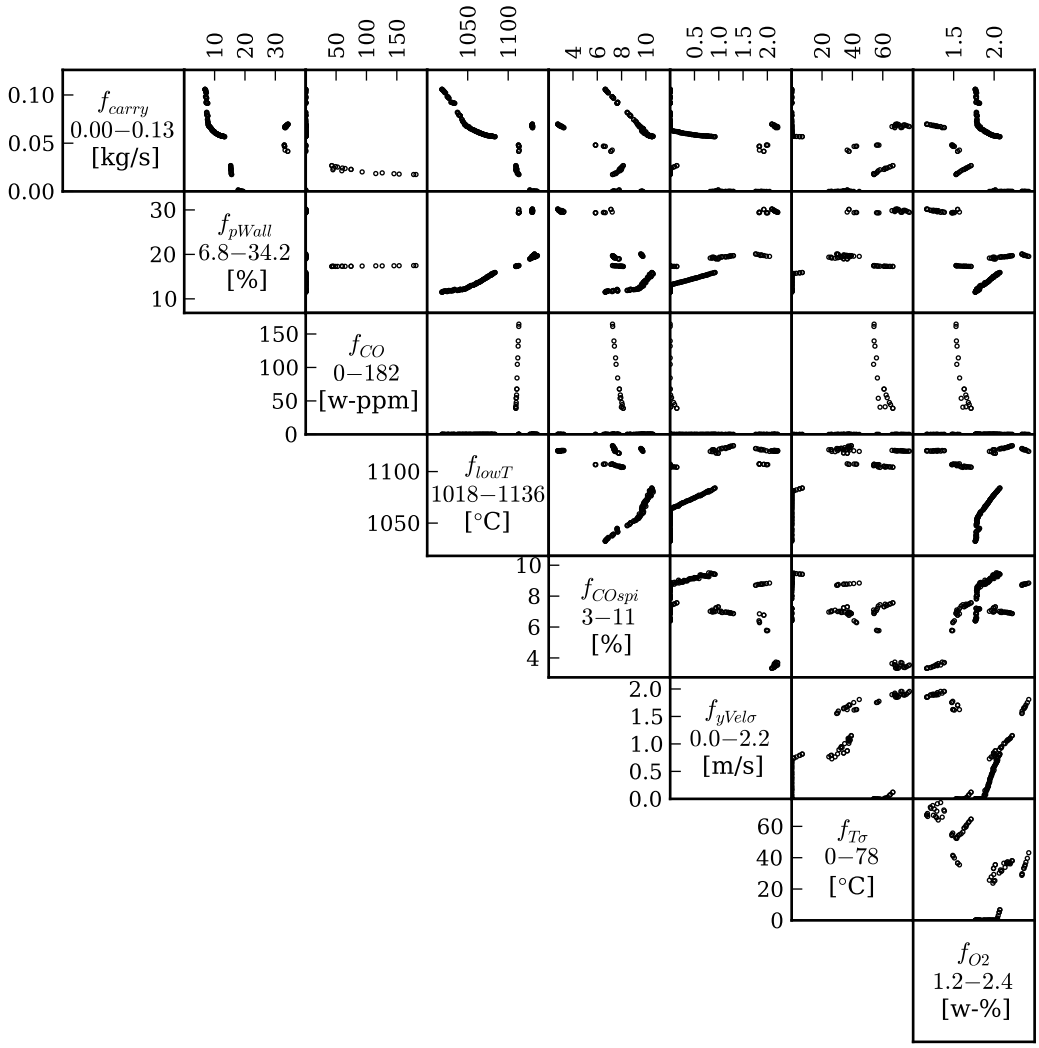


Figure 3.13: Pairwise two-dimensional projections of the final population obtained using the uncoupled method to each pair of the objective functions. The labels of the functions and their ranges are shown in the diagonal boxes. Fronts visible in the plots signify trade-off or concurrency relationships between the objectives.

carryover and the lower furnace temperature (row one, column four). This is a concurrency, because the lower furnace temperature is wanted to be maximized. The phenomenon is understandable because a lower carryover often results in better combustion. A trade-off exists between the amount of particles landing on the walls and the lower furnace temperature (row two, column four). This can be explained by the trade-off that was noticed between the particles landing on the walls and the carryover. In this case, the higher amount of particles landing on the walls results, because of other factors, in a better combustion in the lower furnace. There is also a concurrency identifiable between the particles landing on the walls and the standard deviation of upward velocity (row two, column six). It might be that the particles combusting near the walls cause disturbances in the y-velocity profile.

In Figure 3.13, the plots related to the mean CO emission (complete column two and complete row three) show strange behavior with respect to the other objectives. The emission is very close to zero in almost all of the design points and it does not seem to have any clear relationships to the other objectives. This might be because of the iteration errors associated with the solution of the CO that were detected in the results of the error study in Subsection 3.1.1.

As a conclusion, it can be said that, based on the results presented, optimization using the uncoupled method works as intended and it converges to a Pareto optimal front of designs in the objective space. It was shown by visualizing the Pareto solutions by using the ranges they obtain in the objectives (Figure 3.11) that the Pareto solutions are clearly better than the base design that was used as the starting point. According to these results, the performance of the boiler can be improved by basing the design on one of the geometries on the Pareto front. Also, four designs chosen from the the obtained Pareto front were inspected further and compared to the base design (Figure 3.12). Trade-offs and concurrencies were identified in the objectives using a scatterplot matrix (Figure 3.13). These relationships can be used when improvements are desired in a particular objective. It was seen in the visualization of the final population in the design space (Figure 3.10) that it is clearly clustered in a couple of relatively small regions in the design space. When the visualization in the design space (Figure 3.10) was looked at together with the ranges of the objective functions (Figure 3.11), it was noticed that the design points obtain very good objective function values in the clusters. This behavior might indicate that the learner predicted inaccurately good values in these regions because of a too small training database used.

3.3.2 Coupled Method Optimization

The furnace geometry optimization problem defined in Section 2.2 was solved using the coupled method presented in Subsection 2.5.2. The population was monitored as the generations advanced so that the performance of the method can be examined. The spread of the final population is inspected in the design space and it is compared to the results of the uncoupled method. The predicted CO content values at the nose level obtained using the learner after different numbers of generations are analyzed to see how the learner approximates the important f_{CO} function. Using this information, it is possible to show how the iteration errors in the f_{CO} values computed using CFD may affect the results of the geometry optimization.

Ranges the final geometries obtain in the design variables and objective functions are inspected. They are used to compare the results obtained using the coupled method to the performance of the base design presented in Subsection 2.1.1 and also to the performance of the designs obtained using the uncoupled method. Four solutions from the final population are chosen and compared in detail. The population is also analyzed pair-wise in the objective functions to identify trade-offs and concurrencies in the objectives.

Same program parameters were used as in the uncoupled method optimization, except for the number of generations run. The number of generations run was limited to 40 generations because of computing time available, but at this point the population was not substantially moving in the design space anymore when the generations advanced and the optimization seemed to be converged. The parameters are shown in Table 3.8. Also, the same training database and initial population were used. These are visualized in Figure 3.9.

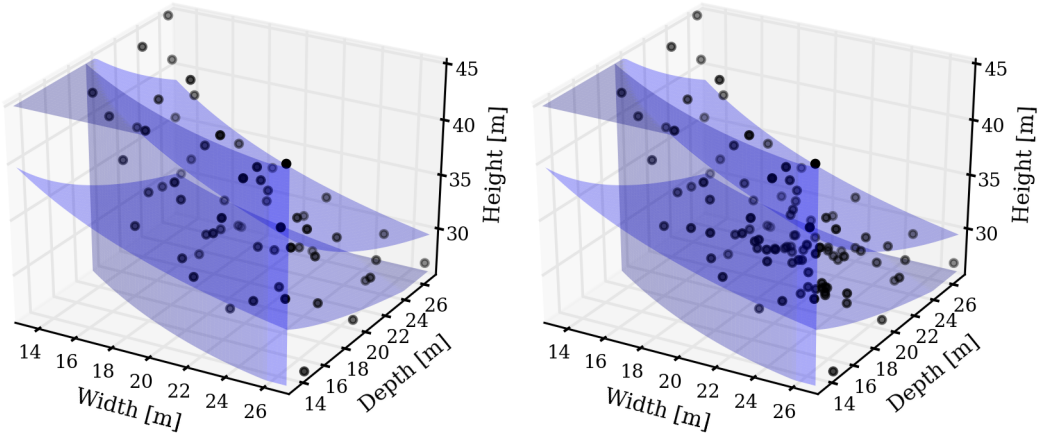
The training database and final database are shown in Figure 3.14 with the geometric constraints. The final database contains 40 more design points evaluated by CFD than the training database. This is because the program was run for 40 generations and one individual was evaluated using CFD each generation. The CFD simulations done by the program behaved well and the convergence monitoring code found that all of them converged before the upper limit of 15 000 iterations was reached. This means that all of the conducted simulations could be used in the optimization process. When the visualizations of the databases are compared, it is evident that the evaluations during the optimization run have mostly been conducted in regions of the design space where the nose height has relatively low values (less than 35 m). This shows that according to the approximate evaluations conducted using the information from training, the designs with large nose heights have been seen to deliver

non-optimal performance. Most likely, these designs have a low nose temperature because they have an excessively large nose height and in these boilers the combustion has completed well below the nose. Because of this, they do not satisfy the nose temperature constraint $g_T \geq g_T^{min}$.

The final population obtained using the coupled method after 40 generations is presented in Figure 3.15. The obtained optimal solutions all have a height of less than 31 m and mostly have large furnace floors. The widths are all larger than 20 m and the depths range from 17 m to 25 m. Solutions with both larger widths than depths and smaller widths than depths can be found. None of the solutions has a square furnace floor. The figure also shows a comparison of the final populations obtained using the uncoupled and coupled methods. There is very little overlap in the populations and it is clear that the methods have converged to different sets of solutions in the design space.

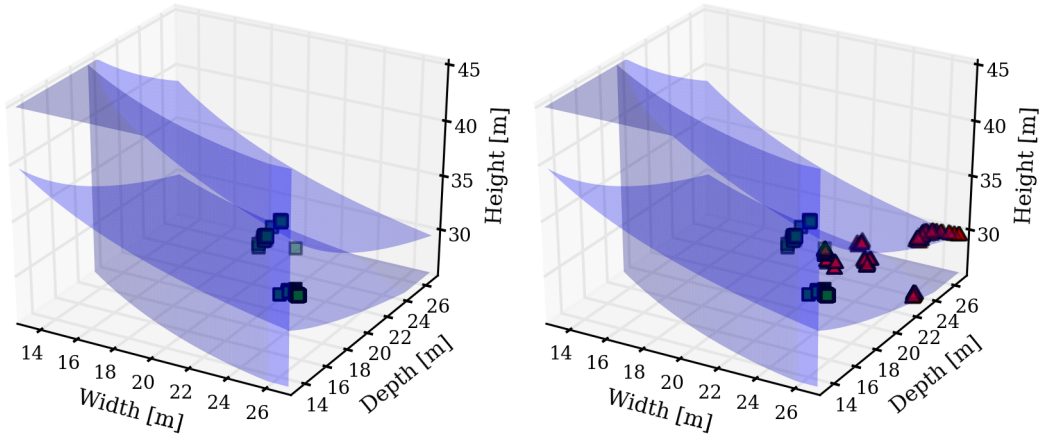
It is seen in Figure 3.15 that the solutions are clearly clustered, which results in a discontinuous Pareto front. The same phenomenon was detected in the uncoupled method optimization. It was speculated that weak training caused the clustering when the uncoupled method was used, but this is not a likely reason in the present case. The real front might be discontinuous or the clustering can be caused iteration errors.

It was found in the iteration error study in Subsection 3.1.1 that the CO value on the nose has a substantial iteration error and in the optimization the CO value on the nose was constrained under 200 ppmw. In general, if the CO value is found using



Training database Final database

Figure 3.14: The training database and final database along with the geometric constraints.



Final population (coupled method) Comparison of final populations

Figure 3.15: The final population obtained using the coupled method (green squares) alone and together with the final population obtained using uncoupled method (red triangles). The geometric constraints are also visualized. Clustering of the solutions around few regions of the design space is visible with both methods.

CFD to be higher than 200 ppmw in a point of the design space, this causes the region near this point to be labeled unfeasible because of a high CO value. If the CO was incorrectly found high because of iteration errors, the region near the point is also incorrectly labeled unfeasible. This causes problems to the optimization algorithm because the CFD evaluations are assumed correct and no more simulations are done in unfeasible regions. This can result in gaps on the Pareto front and clustering of solutions. The final solutions in the clusters obtained are hypothesized to be correct because simulations are done by the coupled method in the promising areas of the design space. Most likely, many simulations have been done near the clusters and iteration errors have not taken the CO values in any simulation in these regions over 200 ppmw.

Contours of the mean CO content on the nose level are shown at nose height coordinate $h = 30.0$ m as predicted by the learner after different numbers of generations in Figure 3.16. The design space is colored gray in the figure. Contours of $f_{CO} = 200$ ppmw are drawn to indicate the feasible regions. Contours of $f_{CO} = 1500$ ppmw are plotted to show regions where the CO emission is predicted to be very high.

Figure 3.16 shows that the contours change as the generations advance. This results from points evaluated using CFD being added to the database of the learner. At generation zero, there are three moderately large regions where the f_{CO} is less than 200 ppmw and a one smaller region. After 13 generations, the regions have

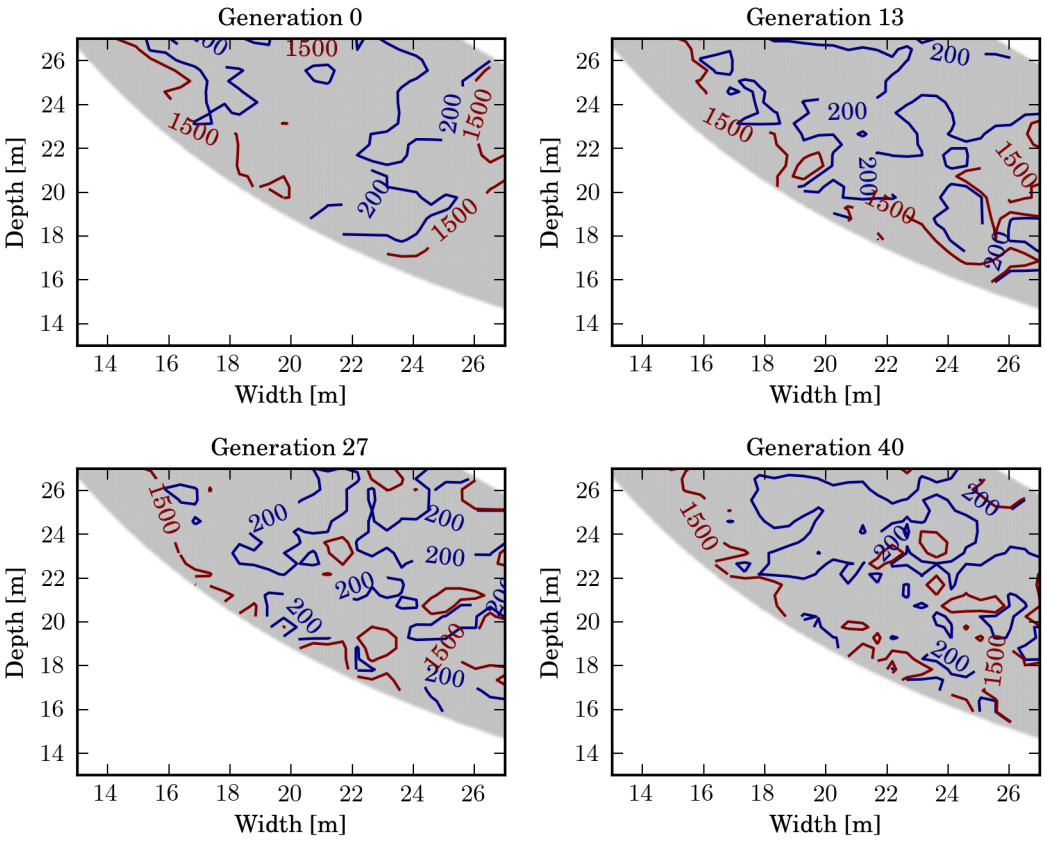


Figure 3.16: Contours of mean CO content at nose level at nose height coordinate $h = 30.0$ m as predicted by the learner after different numbers of generations. The design space is colored gray.

changed and there exists a one large region and several smaller ones. At 27 and 40 generations, the regions where f_{CO} is less than 200 ppmw are small in area, large in number and disconnected from each other. The behavior of the CO contours shown in the figure does not seem realistic. The approximations of the learner should get better as the generations advance but the predicted contours at 40 generations are most likely not accurate. The disconnected regions where f_{CO} is less than 200 ppmw and sudden changes from very low to very high CO emission values with minor changes of geometry do not seem reasonable. Furthermore, assuming at least moderately good training, the changes in the predicted contours should be minor when the generations advance. There are few similarities in the contours shown at generations zero and 40.

The phenomenon seen in Figure 3.16 can be explained using the results of the iteration error study given in Subsection 3.1.1. It was found that there are large iteration errors associated with the f_{CO} values obtained using the CFD model. The predicted contours are disturbed when high f_{CO} values are incorrectly obtained at

some points because of the iteration errors. Most likely, this causes the areas of high CO emissions inside the areas of low CO emissions in the contours. The clustering of the solutions on the Pareto front detected using Figure 3.15 can be understood using the contour plots and the results of the iteration error study. Some areas of the real feasible space are predicted as unfeasible because of the iteration errors and this causes the predicted feasible space to be disconnected into small regions where the Pareto solutions cluster. Furthermore, if some region of the real feasible space was incorrectly labeled as unfeasible in the training phase and it is later found feasible, the optimization algorithm has difficulties to move the population to that region. This is because the used algorithm investigates the whole design space only in the early generations. Because of these reasons, it can be speculated that the results obtained belong to the real Pareto front but do not present it completely. The constraint that was set in the optimization problem on the f_{CO} value was most likely not a good choice because of the iteration errors associated with the solution of the CO value using CFD.

In Figure 3.17, the minimum and maximum design variable and objective function values the final population obtains are shown in a parallel coordinate representation. For comparison, the values of the base design are also shown. The ranges in the figure can be used for insight to the performance of the solutions on the Pareto front, because visualizing the objective function values of the whole population of 200 individuals is not feasible. The ranges show that the objective function values obtained by the final population are in general good and within believable bounds. The ranges are more reasonable compared to the ranges obtained using the uncoupled method, given in Figure 3.11. The only objective functions that get values that round to zero are the f_{CO} and f_{COspi}. Both of these objectives can obtain values close to zero in a real boiler also.

The performances of the obtained designs in Figure 3.17 can be compared to the simulated performance of the base design. Because the base design has a value of $f_{CO} = 1111$ ppmw it does not satisfy the constraint on the CO emission. This means that all the obtained geometries are better than the base design because they all have a CO emission below 200 ppmw. The other objective function values of the base design are all between the the minimum and maximum values obtained by the Pareto solutions. The base design has a particularly high value in the amount of particles landing on the walls, compared to the Pareto designs. In all the other objectives the base design has values near the middle of the ranges and it is not comparatively good in any of the objectives. Many solutions on the Pareto front deliver better overall performance than the base design.

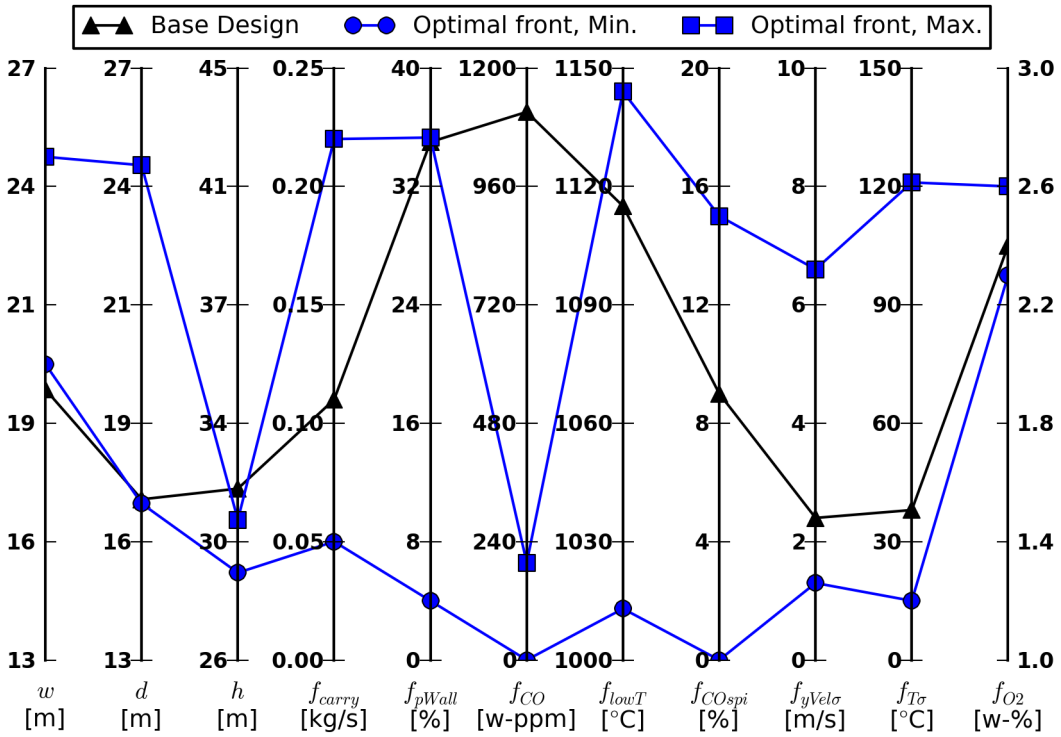


Figure 3.17: Ranges (minimum and maximum values) of the design variables and objective functions on the Pareto front obtained using the coupled method. The design variable and objective function values of the base design are also shown.

A parallel coordinate representation of the objective function values of four designs chosen from the final population is shown in Figure 3.18. The four designs were chosen for further discussion because of their good overall performances in the objective functions and are not necessarily the most suitable designs for any real-life situation. The objective function values of the base design are also plotted for reference. The four designs have widths between 20 m and 25 m, depths between 18 m and 25 m and nose heights between 29 m and 30 m. In the figure, each polyline corresponds to a different design point. The figure should be used together with Figure 3.17 that gives the ranges of the objective functions on the optimal front.

In Figure 3.18, it is seen that the design one has a high value in the f_{carry} objective but a low value in the f_{pWall} objective. It has good values in the rest of the objective functions, except in the f_{lowT} where it has a significantly lower value than the other designs. The design two has a low value in the f_{carry} but a high value in the f_{pWall}. It has good values in the other objectives, except in the f_{COspi} where it has a moderately high value of 9%. The designs three and four give moderately good values in the important f_{carry} and f_{pWall} objectives. The design three has moderate or good values in the rest of the objectives also, except in the f_{CO} where it has a

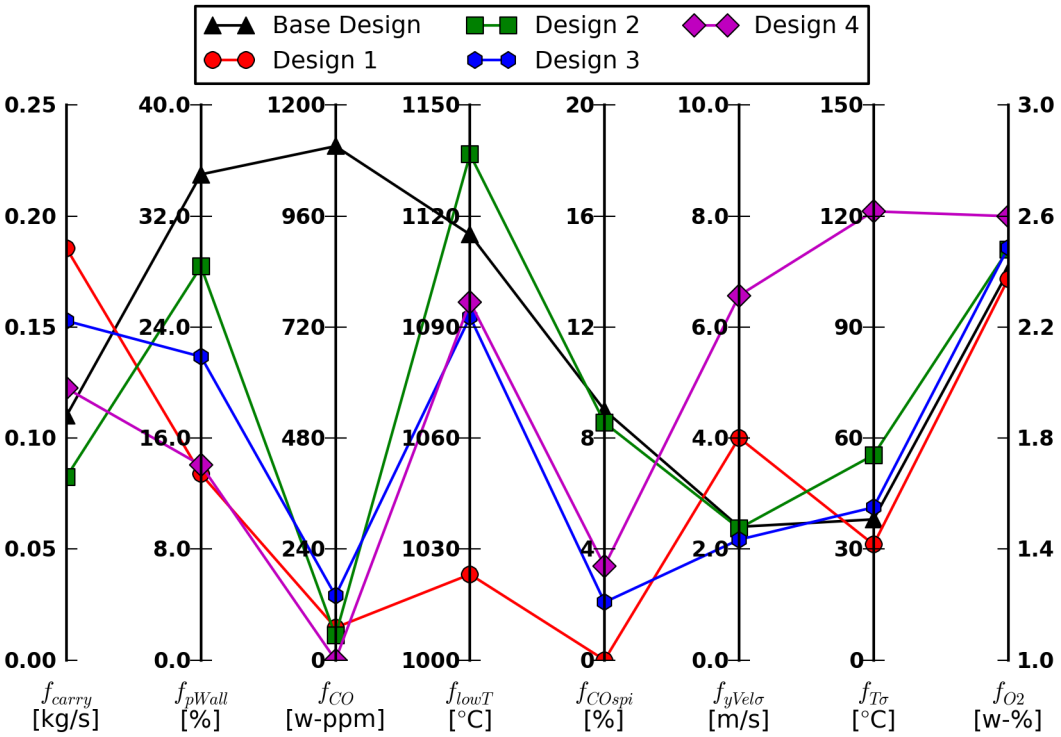


Figure 3.18: The objective function values of the base design and the four solutions chosen from the final population obtained using the coupled method. Each polyline corresponds to a different design point on the Pareto front.

value of 139 ppmw. The design four has moderately good values in the f_{lowT} and the f_{COspi} but bad values in the rest of the objective functions. According to this analysis, there are solutions on the Pareto front which deliver good performance in the objective functions. The four designs chosen from the final population for this analysis have clearly different values in the objectives and trade-off decisions need to be made when the most suitable design for some particular situation is chosen from the Pareto front.

The four chosen designs in Figure 3.18 all have acceptably low carryover values but particularly designs two and three have high values in the f_{pWall} objective. The base design has a very high value in this objective also. The landing behavior of the liquor particles is inspected further because when carryover is low it becomes important to minimize the f_{pWall} objective to obtain a good boiler performance. The particle landing patterns are inspected by studying carbon landing rates on the walls in the lower furnace. This can be done because all of the carbon originates from the black liquor particles. Carbon landing rates on the walls in the furnace obtained for the four inspected designs and the base design using CFD simulations are shown in Figure 3.19. The designs in the figure are drawn on the same scale.

The differences in the floor sizes and floor dimensions between the designs are clearly visible.

Figure 3.19 shows that there are large regions on the walls of the base design where carbon landing rate is high. This means that in these regions a large amount of liquor is landing on the walls. Particularly, there is a lot of liquor landing on the front and rear walls in the base design. The design two shows similar overall behavior, even though the landing patterns are clearly different. In the designs one, three and four there is not as much liquor landing on the walls. The rear wall shows moderate values for the carbon landing rate in all of these three designs, which means that there is clearly liquor landing there. The amount of liquor landing on the rear wall in these three designs is most likely the largest contributor to their total f_{pWall} values. The f_{pWall} values in Figure 3.18 support these observations. The base design and the design two have clearly higher values in the f_{pWall} than the other designs.

Using Figure 3.19, it has been identified where the f_{pWall} values of the considered designs originate. In all of the designs, there is a significant amount of liquor landing on the rear wall. Additionally, in the base design and in the design two, which have smaller depth values than the other designs, a large amount of liquor is landing on the front wall. All of the designs, except the base design, are mostly very good in the other objectives and could be greatly improved by lowering their f_{pWall} values. This could possibly be achieved by tilting the side wall liquor guns closest to the front and rear walls towards the center of the boiler. The liquor guns on the side walls could also be moved closer together and farther away from the front and rear walls.

Pairwise two-dimensional projections of the final population to each pair of the objective functions are shown in a scatterplot matrix representation in Figure 3.20. The figure can be used to find trade-off and concurrency relationships between the objectives. A trade-off relationship is identified if a front where one objective gets better when the other gets worse is found. Conversely, a concurrency relationship is found if a front is identified where both objectives simultaneously get better.

Figure 3.20 shows a trade-off front, apart from few outlier points, between the carryover and the amount of particles landing on the walls (row one, column two). The same trade-off was identified in the scatterplot matrix analysis of the results of the uncoupled method, in Figure 3.13. It is speculated that the reasons for the trade-off are the same here and that the geometry changes that reduce the amount of particles landing on the walls cause flow patterns in the furnace which raise the total carryover. Also, similarly as in the results of the uncoupled method, there is

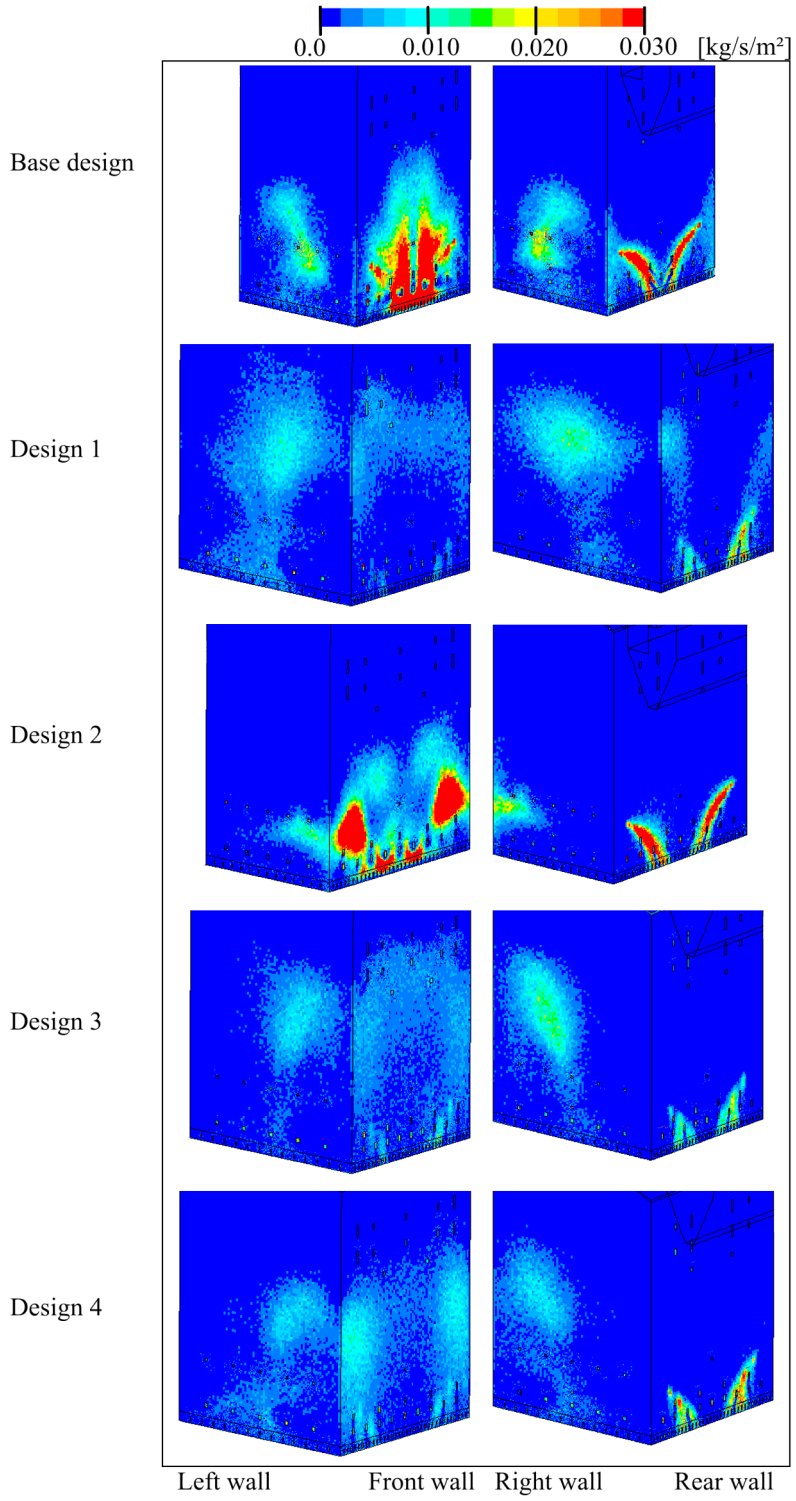


Figure 3.19: Carbon landing rates in the furnace for the base design and for the four solutions chosen from the final population obtained using the coupled method. The names of the designs are listed on the left side of the figure and the wall names are listed on the bottom of the figure. The color bar is shown on the top. The designs are drawn on the same scale.

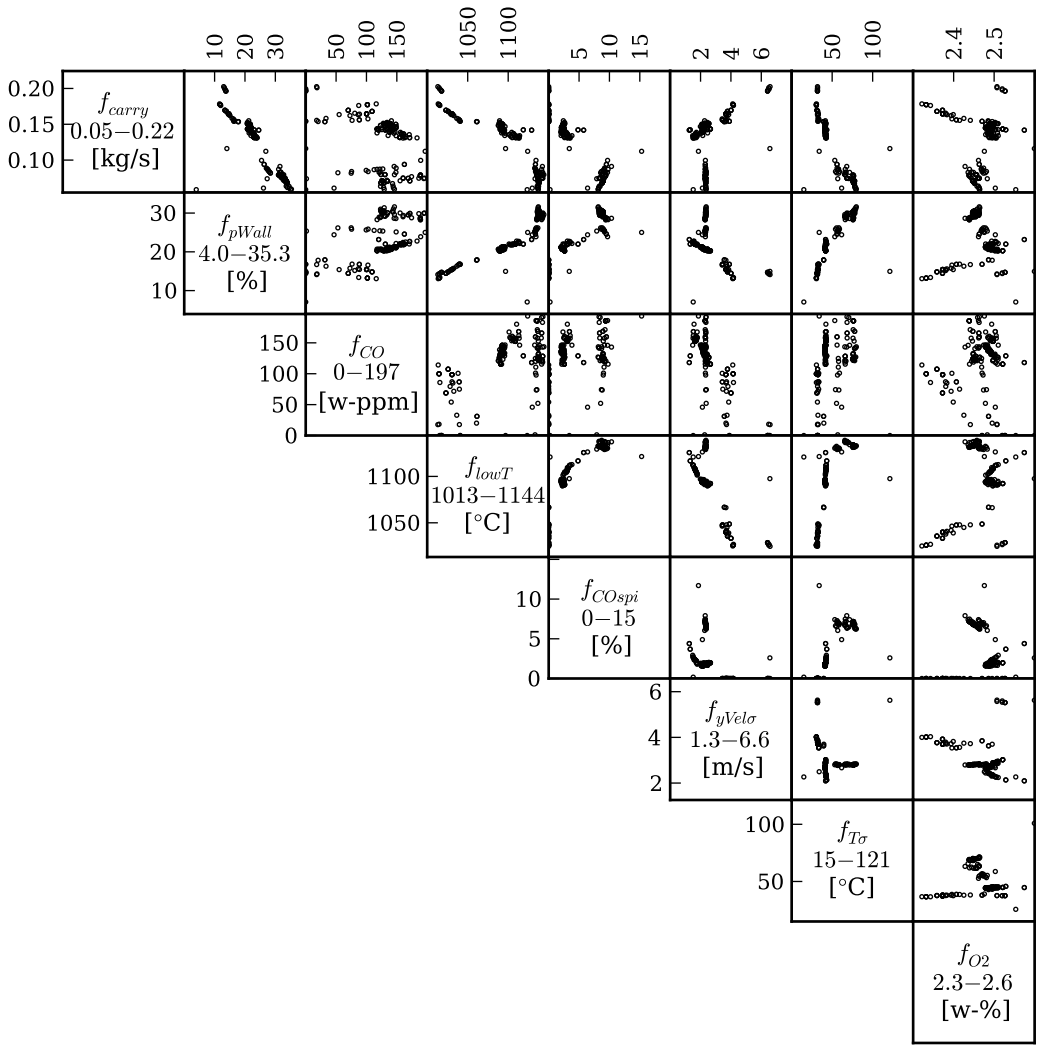


Figure 3.20: Pairwise two-dimensional projections of the final population obtained using the coupled method to each pair of the objective functions. The labels of the functions and their ranges are shown in the diagonal boxes. Fronts visible in the plots signify trade-off or concurrency relationships between the objectives.

a concurrency between the carryover and the lower furnace temperature (row one, column four). Here, a lower carryover results in better combustion. The trade-off between the amount of particles landing on the walls and the lower furnace temperature (row two, column four) which was identified in the results of the uncoupled method is also found in Figure 3.20. The phenomenon is explained by the trade-off between the f_{carry} and f_{pWall} and the concurrency of the f_{carry} and f_{lowT}.

The f_{pWall} and $f_{T\sigma}$ objectives (row two, column six) are concurrent according to Figure 3.20. This concurrency was also found in the results of the uncoupled method and most likely signals that if particles are combusting near the walls it causes

disturbances in the upward velocity profile. There also seems to be a trade-off between the f_{lowT} and f_{COspi} objectives (row four, column five). This is strange because a better combustion in the lower furnace should result in a more even CO profile on the nose level. There is a concurrency in the f_{lowT} and $f_{yVeloσ}$ objectives (row four, column six). The more even upward velocity profile corresponding to a higher lower furnace temperature signaling good combustion is understandable. There also seems to be a trade-off between the $f_{yVeloσ}$ and f_{O2} objectives (row six, column eight), which is hard to interpret.

The CO content plots (complete column two and complete row three), in Figure 3.20, show similar strange behavior as in the uncoupled method geometry optimization but it should be noted that now the values are not as often predicted to be zero. There are no clear relationships between the CO emission and the other objectives and it is hypothesized that this is caused by the iteration errors in the CO values obtained using the CFD evaluations.

It can be concluded that the CFD-optimization using the coupled method works as intended and that by using it a Pareto front of designs can be obtained. In the present optimization task, the ranges that were obtained in the objective functions (Figure 3.17) were used to compare the results to the results of the uncoupled method (Figure 3.11) and to the performance of the base design that was used as the starting point. The results of the coupled method were found significantly more reliable than the results of the uncoupled method. It was also seen that the performance of the boiler can be improved by using one of the designs on the obtained Pareto front. Four designs chosen from the final population were analyzed further and compared to the recovery boiler base design (Figure 3.18). Carbon landing rates on the walls of these four designs were also inspected and it was suggested how the total amount of liquor particles landing on the walls could possibly be reduced in them (Figure 3.19). Particular trade-offs and concurrencies were found in the objectives using a scatterplot matrix (Figure 3.20). Some of the same relationships were found as in the results of the uncoupled method (Figure 3.13) but also new relationships were identified. These can be used when improvements are desired in a particular objective.

The final population was seen to be clustered in the design space similarly as in the results of the uncoupled method (Figure 3.15). It was hypothesized that inadequate training was not the reason for the clustering this time. Instead, using the contour plots of the f_{CO} objective as predicted by the learner (Figure 3.16), it was shown that the clustering was most likely caused by the iteration errors in the f_{CO} values calculated using the CFD model. The iteration errors caused parts of the real feasible

space to be labeled incorrectly unfeasible because of erroneous computed high f_{CO} values. This phenomenon caused the gaps in the Pareto front and the clustering of the solutions. The final solutions in the clusters are believed to be reliable but not representing the real Pareto front completely.

3.3.3 Utilization of the Optimization Results in Practice

The utilization of the obtained results in practice involves the selection of a single final solution from the obtained Pareto optimal solutions. This process involves making trade-offs between the objective function values so that a good compromise can be attained. Also, all the information that could not be formulated mathematically to be included in the optimization problem should be used at this point. Finally, the uncertainties related to computational modeling need to be taken into account. The results of the CFD-optimization should be interpreted together with the results of the error estimation studies. If the attainment of a particular value is essential in some objective function, for example because of emission regulations, a margin of safety should be left between the wanted value and the one computed for the prospective design.

With the CFD-optimization methods used one obtains as a result a Pareto front that contains a large but finite number of optimal geometries. In practice, usually one single geometry is wanted that delivers a better performance than the base design that was used as a starting point. To get from the Pareto front to a single solution, manual work is needed. It is recommended that the amount of optimal solutions (200 in this work) is first pruned down to a more manageable size of approximately five solutions using some criteria. The pruning can be done, for example, by ranking the solutions by their cost and choosing some of the most inexpensive solutions for further inspection. When there is only a small number of solutions left to be considered, a parallel coordinate representation can be used to compare their objective function values against each other. A manual performance comparison of a few solutions is perfectly practical, whereas it is unfeasible to manually process the whole Pareto front of possibly hundreds of solutions this way.

The results of the CFD-optimization can also be used to check whether the original optimization problem is well-defined. The geometries obtained can be used to design the recovery boilers according to the design rules that are used in the optimization problem definition. It should be checked that these designs make sense and that there are no obvious design flaws present. Also, the designs and their predicted respective performances in the objectives can be used together to find relationships

between design choices and the performance in the objective functions. Sometimes, it can be obvious why the CFD model predicts some particular value for an objective only after the design is looked at together with the results.

It is possible that so much new information is obtained from the analysis of the results that it becomes relevant to reformulate the original optimization problem using different specifications in, e.g., the design rules, designer preferences or objectives. After the optimization problem has been reformulated, the CFD-optimization program can be run again. Sometimes, it might be necessary to carry out the described cycle a few times before a final solution is found. This is not an issue because valuable information, and also improved designs, is obtained after each of these cycles.

Chapter 4

Conclusion

In this thesis, a multi-objective optimization problem was solved using a CFD-optimization program. The task was to optimize a furnace geometry of a large capacity recovery boiler. The program was developed by integrating genetic algorithm (GA) optimization and radial basis function (RBF) network learner algorithms with an existing recovery boiler CFD model. Before the optimization task was solved, magnitudes of errors associated with the CFD model were assessed and the performance of the developed CFD-optimization program was verified in test problems.

One of the goals of the thesis was to do an error study with the existing CFD model. The analyses revealed that the simulation results have large iteration errors. It was found out that even converged simulations exhibit time dependent behavior and that iteration averaging is essential for obtaining useful results. A reporting convention based on grid convergence index (GCI) values was used for analyzing discretization errors but because of large iteration errors it was hard to draw conclusions from them. The error studies displayed that iteration errors are not reduced if iterations are continued after convergence has been detected by the monitoring code developed in this work. Finally, no indication was found that it would be necessary to use a higher density grid in the geometry optimization than was originally planned.

The goal of building a CFD-optimization program and integrating it with the CFD model was fulfilled and the program was verified in test problems. Convergence of the program towards the Pareto front was found fast and the implemented diversity and elitism preservation operators were noted to work correctly during the optimization runs. The program also worked reliably in real-world geometry optimization problems. The optimization code was developed in a general way so that it can be expanded and used in the future to perform other, possibly very different,

optimization tasks.

The third and main goal of the thesis was to use the program in a multi-objective furnace geometry optimization task of a large capacity (7000 tds/d) recovery boiler. The task was completed using an optimization method uncoupled with the CFD solver and a method coupled with it. Both methods converged to a Pareto front of optimal geometries that deliver better performance than the original boiler design, but the results obtained using the coupled method were shown to be more reliable. It was hypothesized that the relatively small training database used caused problems in the uncoupled approach. The coupled method produced good results, but the found Pareto front had discontinuities in it. The gaps in the obtained Pareto front were explained using the results of the iteration error study conducted on the used CFD model. It was concluded that most likely the results are correct but do not represent the real Pareto front completely. It was discussed how a single final solution can be selected from the obtained Pareto optimal solutions. It is recommended that the amount of solutions is first pruned down to a manageable size using, for example, cost criteria. When there is only a small number of solutions left to be considered, manual performance comparisons and trade-off decisions can be done. It was also discussed how the results of the CFD-optimization can be used to find new information about the original optimization problem and to learn how different design choices affect a boiler performance. If a large amount of new information is obtained this way, it might become relevant to reformulate the original optimization problem and run the CFD-optimization program again.

The main contribution of the work is the introduction of CFD-optimization and multi-objective optimization methods to the field of recovery boiler modeling. It was shown how the methodology can be used in this challenging application where the CFD evaluations are computationally very expensive. The work also presents practical tools usable for error reporting and advocates thorough error assessment whenever CFD simulations are done.

According to the results of this thesis, future work should be done in reducing iteration errors associated with steady state modeling of large recovery boilers. Discretization errors and suitable grid densities can only be assessed after the iteration errors have been substantially reduced in magnitude. It might be useful to do modeling with considerably denser grids than traditionally used, in the range of tens of millions of cells, and to inspect how it affects the behavior of the model. If the simulations are still iteration dependent with these grids, then the inspection of individual sub-models becomes relevant. Finally, doing time dependent instead of steady state simulations should be considered. They become increasingly practical

every year as the capabilities of available computers increase.

In the future, the possibilities of using the developed optimization program, and the CFD-optimization methodology in general, in different applications should be investigated. With respect to the development of the program, one of the most important future research areas is the investigation of how designer preferences can be better integrated with the tool. With the currently implemented method it can be laborious to determine the relative weights of the objective functions in a way that correctly represents the preferences of the designer. It might also be useful to study incorporating local optimization methods into the code to improve convergence speed and accuracy near the Pareto front. Another interesting area of study is the integration of optimization tools with open source solver codes, because they provide a more natural interface for extensions than commercial codes.

In this work, the developed CFD-optimization program was used to optimize a furnace geometry but, in general, it can be used for all kinds of design problems related to recovery boilers. The framework can be even used in model development or validation. For example, if measured data is available, one can set some parameters of the model as design variables and find the optimum parameter settings that best correspond to the measurements. The possibilities for using the developed program are endless.

References

Adams, T. (editor). 1997. Kraft recovery boilers. Atlanta, USA: TAPPI Press. 381 pp. ISBN 0-9625985-9-3.

Alander, J. 1992. On optimal population size of genetic algorithms. Proceedings of the 1992 international conference on software engineering. Vol. 1. pp. 65–70. ISBN 0-8186-2760-3.

ANSYS 2011a. DesignModeler 14.0. [Computer software].

ANSYS 2011b. Fluent 14.0. [Computer software].

ANSYS 2011c. Meshing 14.0. [Computer software].

ANSYS 2011d. Monitoring residuals. Fluent 14.0 user's guide.

ANSYS 2011e. P-1 radiation model theory. Fluent 14.0 user's guide.

Belegundu, A. & Chandrupatla, T. 2011. Optimization concepts and applications in engineering. Cambridge, USA: Cambridge University Press. 463 pp. ISBN 978-0-521-87846-3.

Beliganur, N. & LeBeau, R. & Hauser, T. 2007. Application of genetic algorithms and neural networks to unsteady flow control optimization. 18th AIAA computational fluid dynamics conference. Vol. 1. pp. 61–70. ISBN 978-156347899-4.

Bergroth, N. & Engblom, M. & Mueller, C. & Hupa, M. 2010. CFD-based modeling of kraft char beds - part 1: Char bed burning model. TAPPI journal. Vol. 9:2. pp. 6–13.

Branke, J. & Kaußler, T. & Schmeck, H. 2001. Guidance in evolutionary multi-objective optimization. Advances in engineering software. Vol. 32:6. pp. 499–507. ISSN 0965-9978.

Brizzolara, S. & Curtin, T. & Bovio, M. & Vernengo, G. 2012. Concept design and hydrodynamic optimization of an innovative SWATH USV by CFD methods. Ocean dynamics. Vol. 62:2. pp. 227–237. ISSN 1616-7341.

Bäck, T. 1996. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. New York, USA: Oxford University Press. 314 pp. ISBN 0-19-509971-0.

Chinchuluun, A. & Pardalos, P. 2007. A survey of recent developments in multiobjective optimization. Annals of operations research. Vol. 154:1. pp. 29–50. ISSN 0254-5330.

Cvetkovic, D. & Parmee, I. 2002. Preferences and their application in evolutionary multiobjective optimization. IEEE transactions on evolutionary computation. Vol. 6:1. pp. 42–57. ISSN 1089-778X.

Deb, K. 2001. Multi-objective optimization using evolutionary algorithms. Sussex, England: John Wiley and Sons. 515 pp. ISBN 0-471-87339-X.

Deb, K. & Pratap, A. & Agarwal, S. & Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation. Vol. 6:2. pp. 182–197. ISSN 1089-778X.

Duvigneau, R. & Visonneau, M. 2004. Hybrid genetic algorithms and artificial neural networks for complex design optimization in CFD. International journal for numerical methods in fluids. Vol. 44:11. pp. 1257–1278. ISSN 0271-2091.

Engblom, M. & Bergroth, N. & Mueller, C. & Jones, A. & Brink, A. & Hupa, M. 2010. CFD-based modeling of kraft char beds - part 2: A study on the effects of droplet size and bed shape on bed processes. TAPPI journal. Vol. 9:2. pp. 15–20.

Engblom, M. & Miikkulainen, P. & Brink, A. & Hupa, M. 2012. CFD-modeling for more precise operation of the kraft recovery boiler. TAPPI journal. Vol. 11:11. pp. 19–27. ISSN 0734-1415.

Fazzolari, A. & Gauger, N. & Brezillon, J. 2007. Efficient aerodynamic shape optimization in MDO context. Journal of computational and applied mathematics. Vol. 203:2. pp. 548–560. ISSN 0377-0427.

Ferziger, J. & Perić, M. 1996. Further discussion of numerical errors in CFD. International journal for numerical methods in fluids. Vol. 23:12. pp. 1263–1274. ISSN 0271-2091.

Ferziger, J. & Perić, M. 2002. Computational methods for fluid dynamics. 3rd ed. Berlin, Germany: Springer-Verlag. 423 pp. ISBN 3-540-42074-6.

Fletcher, C. 1991. Computational techniques for fluid mechanics volume 1: Fundamental and general techniques. 2nd ed. Berlin, Germany: Springer-Verlag. 401 pp. ISBN 3-540-53058-4.

Foli, K. & Okabe, T. & Olhofer, M. & Jin, Y. & Sendhoff, B. 2006. Optimization of micro heat exchanger: CFD, analytical approach and multi-objective evolutionary algorithms. International journal of heat and mass transfer. Vol. 49:5. pp. 1090–1099. ISSN 0017-9310.

Gaspar-Cunha, A. & Vieira, A. 2005. A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. International journal of computers, systems and signals. Vol. 6:1. pp. 18–36.

Georgopoulou, C. & Giannakoglou, K. 2009. A multi-objective metamodel-assisted memetic algorithm with strength-based local refinement. Engineering optimization. Vol. 41:10. pp. 909–923. ISSN 0305-215X.

Grace, T. 1995. A critical review of computer modeling of kraft recovery boilers. TAPPI journal. Vol. 79:7. pp. 182–190. ISSN 0734-1415.

Grace, T. & Lien, S. & W., S. & Tse, D. & Abdullah, Z. & Salcudean, M. 1998. Validation of CFD-based recovery furnace models. Proceedings of the 1998 international chemical recovery conference. Vol. 1. pp. 271–281.

Han, S. & Maeng, J. 2003. Shape optimization of cut-off in a multi-blade fan/scroll system using neural network. International journal of heat and mass transfer. Vol. 46:15. pp. 2833–2839. ISSN 0017-9310.

Haykin, S. 1999. Neural networks: a comprehensive foundation. 2nd ed. New Jersey, USA: Prentice-Hall. 842 pp. ISBN 0-13-273350-1.

Hämäläinen, J. & Hämäläinen, T. & Madetoja, E. & Ruotsalainen, H. 2008. CFD-based optimization for a complete industrial process: papermaking. In: Thévenin, D. & Janiga, G. (editors). Optimization and Computational Fluid Dynamics. Magdeburg, Germany: Springer-Verlag. 293 pp. ISBN 978-3-540-72152-9.

Jin, R. & Chen, W. & Simpson, T. 2001. Comparative studies of metamodelling techniques under multiple modelling criteria. Structural and multidisciplinary optimization. Vol. 23:1. pp. 1–13. ISSN 1615-147X.

Kankkunen, A. & Miikkulainen, P. 2003. Particle size distribution of black liquor sprays with a high solids content in recovery boilers. IFRF combustion journal. ISSN 1562-479X.

Kankkunen, A. & Miikkulainen, P. & Järvinen, M. 2011. Initial stages and overlapping processes of a black liquor droplet in a kraft recovery boiler. Proceedings of 2011 TAPPI peers conference. Vol. 1. pp. 434–457. ISBN 978-161839430-9.

Keane, A. & Scanlan, J. 2007. Design search and optimization in aerospace engineering. Philosophical transactions of the royal society a: Mathematical, physical and engineering sciences. Vol. 365:1859. pp. 2501–2529. ISSN 1364-503X.

Konak, A. & Coit, D. & Smith, A. 2006. Multi-objective optimization using genetic algorithms: A tutorial. Reliability engineering & system safety. Vol. 91:9. pp. 992–1007. ISSN 0951-8320.

Kriesel, D. 2007. A brief introduction to neural networks. [Online]. 226 pp. Available at: http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf [Accessed 11 February 2013].

Madsen, J. & Shyy, W. & Haftka, R. 2000. Response surface techniques for diffuser shape optimization. AIAA journal. Vol. 38:9. pp. 1512–1518. ISSN 0001-1452.

Marler, R. & & Arora, J. 2004. Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization. Vol. 26:6. pp. 369–395. ISSN 1615-147X.

Mengistu, T. & Ghaly, W. 2008. Aerodynamic optimization of turbomachinery blades using evolutionary methods and ANN-based surrogate models. Optimization and engineering. Vol. 9:3. pp. 239–255. ISSN 1389-4420.

Miikkulainen, P. & Kankkunen, A. & Järvinen, M. & Fogelholm, C.-J. 2009. The significance of velocity in black liquor spraying. TAPPI journal. Vol. 8:1. pp. 36–40.

Miikkulainen, P. & Pakarinen, L. & Metsämuuronen, N. 2010. Challenges in validating recovery boiler furnace models in practice. Proceedings of the 2010 international chemical recovery conference. Vol. 1. pp. 60–72.

Mueller, C. & Eklund, K. & Forssen, M. & Hupa, M. 2004. Influence of liquor-to-liquor differences on recovery furnace processes - a CFD study. Proceedings of the 2004 international chemical recovery conference. Vol. 2. pp. 979–997.

Oberkampf, W. & Trucano, T. 2002. Verification and validation in computational fluid dynamics. Progress in aerospace sciences. Vol. 38:3. pp. 209–272. ISSN 0376-0421.

Poloni, C. & Giurgevich, A. & Onesti, L. & Pediroda, V. 2000. Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. Computer methods in applied mechanics and engineering. Vol. 186:2. pp. 403–420. ISSN 0045-7825.

Rangaiah, G. 2010. Multi-objective optimization: techniques and applications in chemical engineering. Singapore, Singapore: World Scientific. 709 pp. ISBN 981-4299-20-0.

Regis, R. & Shoemaker, C. 2004. Local function approximation in evolutionary algorithms for the optimization of costly functions. IEEE transactions on evolutionary computation. Vol. 8:5. pp. 490–505. ISSN 1089-778X.

Roache, P. 1994. Perspective: a method for uniform reporting of grid refinement studies. Journal of fluids engineering. Vol. 116:3. pp. 405–413. ISSN 0098-2202.

Roache, P. 1997. Quantification of uncertainty in computational fluid dynamics. Annual review of fluid mechanics. Vol. 29:1. pp. 123–160. ISSN 0066-4189.

Roache, P. 1998. Verification and validation in computational science and engineering. Albuquerque, USA: Hermosa Publishing. 446 pp. ISBN 0-913478-08-3.

Saario, A. 2008. Mathematical modeling and multiobjective optimization in development of low-emission industrial boilers. Ph.D. thesis. Tampere University of Technology. Tampere, Finland. 130 pp. ISSN 1459-2045.

Saviharju, K. & Pakarinen, L. & Wag, K. & Välipakka, I. 2004. Numerical modeling feedback in recovery boilers. Proceedings of the 2004 international chemical recovery conference. Vol. 1. pp. 247–262.

Simpson, T. & Poplinski, J. & Koch, P. & Allen, J. 2001. Metamodels for computer-based engineering design: survey and recommendations. Engineering with computers. Vol. 17:2. pp. 129–150. ISSN 0177-0667.

Soto, O. & Löhner, R. & Yang, C. 2004. An adjoint-based design methodology for CFD problems. International journal of numerical methods for heat and fluid flow. Vol. 14:6. pp. 734–759. ISSN 0961-5539.

Srinivas, M. & Patnaik, L. 1994. Genetic algorithms: A survey. Computer. Vol. 27:6. pp. 17–26. ISSN 0018-9162.

Tanaka, M. & Watanabe, H. & Furukawa, Y. & Tanino, T. 1995. GA-based decision support system for multicriteria optimization. Proceedings of the 1995 IEEE international conference on systems, man and cybernetics. Vol. 2. pp. 1556–1561. ISSN 0884-3627.

Thévenin, D. 2008. Introduction. In: Thévenin, D. & Janiga, G. (editors). Optimization and Computational Fluid Dynamics. Magdeburg, Germany: Springer-Verlag. 293 pp. ISBN 978-3-540-72152-9.

Thévenin, D. & Janiga, G. (editors). 2008. Optimization and Computational Fluid Dynamics. Magdeburg, Germany: Springer-Verlag. 293 pp. ISBN 978-3-540-72152-9.

Vakkilainen, E. 2005. Kraft recovery boilers - principles and practice. Helsinki, Finland: Suomen Soodakattilayhdistys. 244 pp. ISBN 952-91-8603-7.

Vakkilainen, E. & Kjaldman, L. & Taivassalo, V. & Kilpinen, P. & Norstrom, T. 1998. High solids firing in an operating recovery boiler - comparison of CFD predictions to practical observations in the furnace. Proceedings of the 1998 international chemical recovery conference. Vol. 1. pp. 245–256.

Van den Braembussche, R. 2008. Numerical optimization for advanced turbomachinery design. In: Thévenin, D. & Janiga, G. (editors). Optimization and Computational Fluid Dynamics. Magdeburg, Germany: Springer-Verlag. 293 pp. ISBN 978-3-540-72152-9.

Zhou, A. & Qu, B. & Li, H. & Zhao, S. & Suganthan, P. & Zhang, Q. 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and evolutionary computation. Vol. 1:1. pp. 32–49. ISSN 2210-6502.

Zhou, H. & Cen, K. & Fan, J. 2004. Modeling and optimization of the NOx emission characteristics of a tangentially fired boiler with artificial neural networks. Energy. Vol. 29:1. pp. 167–183. ISSN 0360-5442.

Zhou, Z. & Ong, Y. & Lim, M. & Lee, B. 2007. Memetic algorithm using multi-surrogates for computationally expensive optimization problems. Soft computing. Vol. 11:10. pp. 957–971. ISSN 1432-7643.

Zitzler, E. & Thiele, L. & Laumanns, M. & Fonseca, C. & Da Fonseca, V. 2003. Performance assessment of multiobjective optimizers: An analysis and review. Evolutionary computation, ieee transactions on. Vol. 7:2. pp. 117–132. ISSN 1089-778X.

Appendix A

Discretization Error Estimation Using GCI

Discretization error in a simulation can be estimated by calculating a so called grid convergence index (GCI) for the grid used. It is based on discretization error estimators derived from the theory of Richardson extrapolation. The basic idea is to approximately relate the results from any grid refinement test to the expected results from a grid doubling when a second order method is used. The method is presented here according to Roache (1994).

Richardson extrapolation is based on the idea that that discrete solutions f are assumed to have a Taylor series representation

$$f = f_{exact} + g_1 h + g_2 h^2 + g_3 h^3 + ..., \tag{A.1}$$

where f_{exact} is the exact solution, g_1, g_2, etc., are related to all orders of the derivatives of the solution and h is the grid spacing. If a second order method is used, then $g_1 = 0$ and two discrete solutions f_1 (fine grid) and f_2 (coarse grid) on two different grids with spacings h_1 (fine grid) and h_2 (coarse grid) can be combined to eliminate the g_2 term, which results in

$$f_{exact} = \frac{(h_2^2 f_1 - h_1^2 f_2)}{h_2^2 - h_1^2} + H, \tag{A.2}$$

where H are the higher-order terms. Using the grid refinement ratio $r = h_2/h_1$ and dropping the higher-order terms gives

$$f_{exact} \approx f_1 + \frac{f_1 - f_2}{r^2 - 1}, \tag{A.3}$$

which in general is third-order accurate approximation for f_{exact} if $r = 2$, and if f_1 and f_2 are second-order accurate. The method can be derived in a similar way for methods of order p and thus generalized as

$$f_{exact} \approx f_1 + \frac{f_1 - f_2}{r^p - 1}, \tag{A.4}$$

which is generally $(p + 1)$ order accurate. It can be seen in Equation (A.4) that a correction is made to the fine grid solution f_1 to obtain a more accurate approximation. Richardson extrapolation can be applied to the solution at each grid point or to solution functionals. The value of f_{exact} computed from the equation can be used as an improved estimate for the solution or to obtain an estimate of the discretization error band for f.

When examining the generalized Richardson extrapolation Equation (A.4), the second term on the right-hand side can be considered to be an an error estimator of f_1. The equation can be expressed as

$$A_1 = E_1 + \mathcal{O}(h^{p+1}, E_1^2), \tag{A.5}$$

where A_1 is the actual fractional error and E_1 is the estimated fractional error for f_1. They are defined as

$$A_1 = \frac{f_1 - f_{exact}}{f_{exact}}, \tag{A.6}$$

$$E_1 = \frac{\epsilon}{r^p - 1}. \tag{A.7}$$

In Equation (A.7), the relative error ϵ is obtained from

$$\epsilon = \frac{f_2 - f_1}{f_1}. \tag{A.8}$$

Relative error ϵ is the quantity commonly reported in grid refinement studies when $r = 2$ and $p = 2$ are used and it is then understood as an error band for the fine grid solution. It should not be used in general because it does not take into account the values of r and p. One can note that, for example, by choosing $r \approx 1$, values of f_2 and f_1 are close to each other and ϵ becomes arbitrarily small.

An error estimate is needed for the coarser grid also if it is used for simulations. The Richardson extrapolation can then be expressed as

$$f_{exact} \approx f_2 + \frac{(f_1 - f_2)r^p}{r^p - 1}, \tag{A.9}$$

and the estimated fractional error E_2 for f_2 is defined as

$$E_2 = \frac{\epsilon r^p}{r^p - 1}. \tag{A.10}$$

GCIs for the fine and coarse grids are derived from the fractional errors E_1 and E_2, respectively, by adding a factors of safety to the expressions. The GCI can be considered as an error bound or an estimate of the percentage the computed value is away from the value of the asymptotic numerical value. The GCI on the fine grid is defined as

$$\mathrm{GCI}_{fine} = \frac{F_s|\epsilon|}{r^p - 1}, \tag{A.11}$$

where F_s is a factor of safety, recommended by Roache (1997) to be $F_s = 3.0$ for comparisons of two grids and $F_s = 1.25$ for comparisons over three or more grids. The coarse grid GCI is obtained from

$$\mathrm{GCI}_{coarse} = \frac{F_s|\epsilon|r^p}{r^p - 1}, \tag{A.12}$$

$$\mathrm{GCI}_{coarse} = \mathrm{GCI}_{fine} + 3|\epsilon|. \tag{A.13}$$

The method of using GCIs depends on the assumption that the grids are in the asymptotic range, where the error decreases with a speed relative to the order of the discretization method used as the grid is refined. If the asymptotic range has not been achieved the error estimates obtained using the GCIs are not valid. According to Roache (1994), it can be checked whether a grid is in the asymptotic range by computing the same case using two progressively refined grids and by examining their GCI values. All of the grids are in the asymptotic range if $\mathrm{GCI}_{23} \approx r^p\mathrm{GCI}_{12}$, where GCI_{23} is computed from the intermediate to the coarse grid and GCI_{12} from the fine grid to the intermediate grid. This idea is developed further in this work and the relative difference of GCI_{23} and $r^p\mathrm{GCI}_{12}$ is formulated into an asymptotic range indicator S

$$S = \frac{|\mathrm{GCI}_{23} - r^p\mathrm{GCI}_{12}|}{\mathrm{MAX}(|\mathrm{GCI}_{23}|, |r^p\mathrm{GCI}_{12}|)}, \tag{A.14}$$

where MAX is a function that returns the larger of its arguments. The absolute difference $|\mathrm{GCI}_{23} - r^p\mathrm{GCI}_{12}|$ is scaled with a reference value that is always the larger of the two quantities. This makes sense because the quantities should be equal to each other and neither one of them can be considered to be a definite reference value. The quantity S obtains values that are between 0 and 1 and signifies how much smaller the smaller of the two values is relative to the larger one. If the value of S is close to 0, it is an indication that the grids are in the asymptotic range.

Based on this theory, uniform reporting of the results of grid-refinement studies with arbitrary r and p are possible. This is particularly useful when using $r = 2$ would make the computational cost of the study too high, as in the present situation. One should regardless note that when using small values for r the leading truncation error term might become too small and be masked by errors from other sources. Roache (1994) suggests that $r = 1.1$ is a suitable practical minimum value for the grid refinement ratio.

Appendix B

Locally Trained RBF network

A methodology of using radial basis function (RBF) networks proposed by Duvigneau and Visonneau (2004) is used in the learner subprogram in this work. The RBF networks are trained locally using only a number of the nearest database entries every time an approximate evaluation is called for. In the method, the hidden layer consists of as many neurons as there are local database entries solved by CFD. The local database size d is a parameter defined by the user. The approach is described below as given by Duvigneau and Visonneau (2004).

In the RBF network each neuron N_i has a center $\vec{c}_i$ associated with it, which has the value of the i:th local database entry,

$$\vec{c}_i = \vec{x}_i, \qquad i = 1, 2, ..., d. \tag{B.1}$$

The propagation function of each neuron is the Euclidean distance between an input $\vec{x}$ and the RBFs center $\vec{c}_i$. The total input signal E_i to a neuron is given by

$$E_i = ||\vec{x} - \vec{c}_i||_2. \tag{B.2}$$

The activation function of the RBF neuron is a radial function f_{RBF} of the input signal. Since RBF neurons have the identity as the output function, the output signal S_i is given by the activation function directly as

$$S_i = f_{RBF}(E_i; e), \tag{B.3}$$

where e is an attenuation coefficient chosen by the user, which defines the domain of influence of each RBF. When the Gaussian function is chosen as the activation

function, the output signal can be given as

$$S_i = \exp(-\frac{E_i^2}{e^2}). \tag{B.4}$$

The output value of the whole network with an input $\vec{x}$ is obtained through the weighted sum

$$S(\vec{x}) = \sum_{i=1}^{d} w_i S_i, \tag{B.5}$$

where w_i is the weight of the connection between the neuron N_i of the hidden layer and the output neuron.

The weights of connections w_i between the hidden layer and the output layer need to be solved when the network is trained, and for this an interpolating condition for the solutions in the local database y_t is written as

$$y_t = \sum_{i=1}^{d} w_i S_i, \qquad t = 1, 2, ..., d. \tag{B.6}$$

Training consists only of solving this linear system, for example by the method of least squares, to determine the weights w_i. The attenuation coefficient e and local database size d are parameters defined by the user.