Numerical optimisation of vortex generators in fin and tube heat exchangers

Florian Marcus Fries, Palle Lang Nielsen

Energy Technology, PECT4-1-F19, 2019-06

Master's Project

In compilation of this report, LaTeX is used as writing program. The same program has been used for constructing tables in the report. All pictures occurring in the report, have been created by the authors.

Energy
Aalborg University Esbjerg
http://www.esbjerg.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Numerical optimisation of implementation of vortex generators in fin and tube heat exchangers

Theme:
Master thesis

Project Period:
Spring semester 2019

Project Group:
PECT10-1-F19

Participant(s):
Florian Marcus Fries
Palle Lang Nielsen

Supervisor(s):
Matthias Mandø
Jakob Hærvig

Copies: 2

Page Numbers: 101

Date of Completion:
31st May 2019

Abstract:

This thesis sought to investigate the optimal configuration for vortex generators (VG's) for increase of performance in fin and tube heat exchangers. This was done with a conjugate heat transfer CFD model and numerical optimisation based on a surrogate model and a genetic algorithm. A total of five parameters were investigated: Angle of attack, VG -length and height and x - and y-coordinates of the VG's relative to the tube centre. The numerical optimisation produced a configuration, that yielded a heat transfer increase of $\approx$ 24% with a pressure drop increase $\approx$ 7% in comparison to a plain fin geometry. The new configuration was compared with results from the CFD model. The CFD model showed that the optimisation had a deviation of 1.91% on the loss coefficient and 0.06% of the Nusselt number.

Preface

This report was written as the Master's Thesis on 4th Semester on the Process Engineering and Combustion Technology master of science program at Aalborg University Esbjerg (AAU Esbjerg), 2019.

The software used to model the thermo-fluid dynamics in the heat exchanger is OpenFOAM with FOAMExtend 4. The import and export from the model will be done using Python 3. Optimisation will be done in Matlab.
The report is written to a reader with basic knowledge about fluid dynamics and CFD.

Acknowledgements

Thanks go out to company contact Claus Ibsen.

Aalborg University Esbjerg, May 31, 2019

Palle Lang Nielsen
pnie14@student.aau.dk

Florian Marcus Fries
ffries14@student.aau.dk

Nomenclature

Description of symbols appearing in the report. Symbols not included or duplicate will be described at their respective equations.

Table 1: Nomenclature for symbol in the report

Symbol	Description	Unit
A	Area	m^2
d	Coefficient vector	-
D_i	Inner tube diameter	mm
D_o	Outer tube diameter	mm
d_x	Longitudinal position of VG	mm
d_y	Transversal position of VG	mm
e	Performance evaluation ratio	-
f	Loss coefficient	-
Δf	Crowding distance	-
F_p	Fin pitch	mm
F_S	Safety factor	-
g	Gravitational acceleration	m/s^2
g_s	Grid cell multiplication factor	-
GCI	Grid convergence index	-
Gr	Grashof number	-
H	Transversal tube pitch	mm
h	VG height	mm
h_c	Convective heat transfer coefficient	W/m^2 K
j	Colburn factor	-
k	Turbulent kinetic energy	m^2/s^2
JF	Thermal performance	-
L	Longitudinal tube pitch	mm
l	VG length	mm
lim	Limit	
Nu	Nusselt number	-
p	Convergence rate	-
P	Pressure	Pa
Pr	Prandtl number	-
Pr_t	Turbulent Prandtl number	-

Symbol	Description	Unit
q	Fractional design level	-
r	Refinement ratio	-
rand	Random number between zero and one	-
Re	Reynolds number	-
Ri	Richardson number	-
S	Channel height	mm
s	Amount of levels	-
T	Temperature	K
u_{max}	Average velocity at the smallest cross-sectional area	m/s
u_{bulk}	Bulk velocity across the computational domain	m/s
w	Amount of factors	-
α	VG angle of attack	°
β	Thermal expansion coefficient	1/K
γ	Temperature gradient in flow direction	K/m
δ	Kroneckers delta	-
ϵ	Relative error	-
ε	Model error	-
η	Distribution index	-
θ	VG roll angle	°
λ	Thermal conductivity	W/m K
ν	Kinematic viscosity	m^2/s
ξ	Variable vector	-
ρ	Density	kg/m^3
σ	Spread factor	-
ϕ	Solution parameter	-
χ	Simplex operation coefficient	-
ω	Turbulent frequency	1/s

Table of Contents

Chapter 1

Introduction

Fin and tube heat exchangers are widely used in different industries, such as process industry, automotive, and air-conditioning, among others, as mentioned by Lei et al. (2010). This thesis deals with a design of a heat exchanger proposed by Vestas aircoil, where both material usage and volume are a concern. The application of interest in this thesis is the cooling of the charged air from two stroke marine engines, which utilizes water as a cooling medium. This gives a restriction in varying the flow speed of air going into the cooler, as this parameter is set by the turbocharger. Finding a way to optimise the design as well as streamlining the design process can save the company resources. The common application has a working pressure of 4 bar. An example of such a cooler can be seen in figure 1.1.

Figure 1.1: This figure shows a charged air cooler produced by Vestas aircoil A/S.

Based on the fixed inlet flow rate, the approach to increase the efficiency will be to adjust the geometry of the cooler. The air-side of the cooler is the side with

the highest thermal resistance, based on relative low convective heat transfer. As an example, for refrigerant-air heat exchangers, the air side can account for up to 90% of the thermal resistance of the system, according to Bacellar et al. (2016). This makes the air-side the most promising area to optimise geometry. The overall layout of tubes in the given cooler in figure 1.1 is fixed based on a product line. A passive flow enhancer is, therefore, a desired way to increase the convective heat transfer. One way to increase the efficiency of the heat exchangers is to implement small vortex generators (VG's) in the layout to increase fluid mixing in the boundary layer, without increasing weight, but at the cost of an increase in pressure drop. This pressure drop is the cause of the optimisation problem, as a compromise between low pressure drop and high heat transfer is needed. This report will look into optimising the implementation of VG's in a fin-tube heat exchanger with a staggered tube configuration. Many different variables for implementation of VG's in fin-tube heat exchangers have been investigated in literature, both for staggered and inline tube configuration. The number of variables and method of the investigations vary.

1.1 Vortex Generators

A Vortex Generator (VG) is a thin piece of material that is attached to a surface, in this instance on the fins of the cooler. The purpose of a VG is to induce longitudinal vortices through the domain, to enhance heat transfer, by disrupting the thermal boundary layer. According to Fiebig (1998), the increase in heat transfer enhancement from longitudinal vortices is superior to the one from transverse vortices. Another property is to direct a secondary flow into the tube wake; this can be seen in figure 1.2
There multiple ways to produce the VG's, but in this case, the VG's of interest are Delta winglet Vortex Generators (DVG) lanced from the fin, which will produce a hole in the fin afterwards. Figure 1.3 serves to illustrate a DVG with its characteristic parameters and the hole created by the lancing procedure.
The implementation of VG's comes side effect of an increase in pressure drop. Therefore it is critical to determine a design that enhances heat transfer, but at the same time keeps the pressure drop increase compared to a plain fin geometry relatively low.

1.2 Investigated parameters

Some literature investigate the behaviour of thermal efficiency by performing a parametric study on the same type of geometry. An aspect that all sources in this section have in common is that they all implemented a variation in Reynolds number for their geometry. Common is also that they use a variation of the j/f factor, which describes the ratio of heat transfer compared to friction pressure loss, to evaluate results. The factor will be explained further in chapter 5.
Lei et al. (2010) investigated the optimal angle of attack and wing aspect ratio on

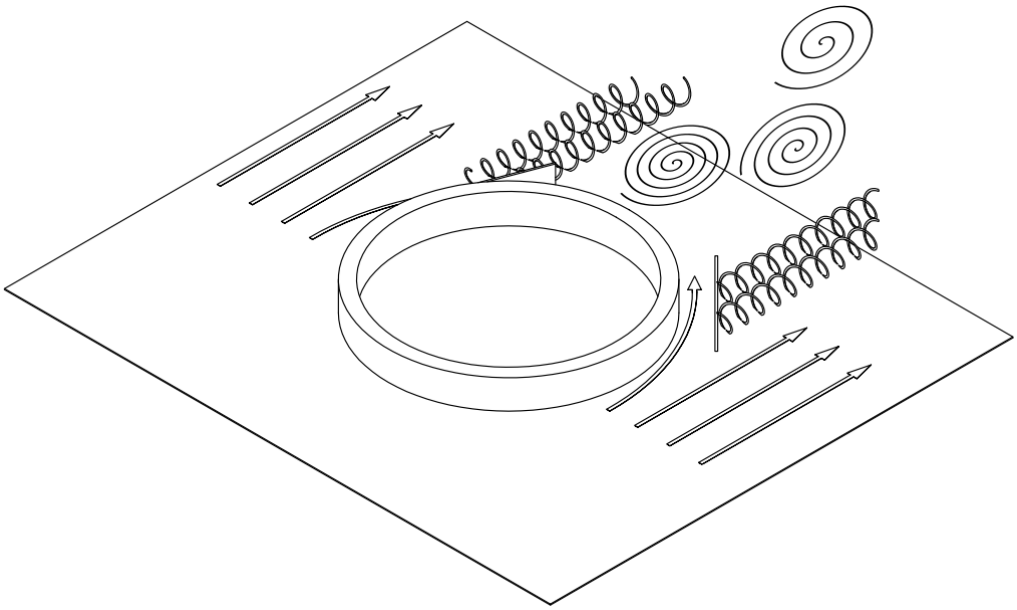


Figure 1.2: This figure illustrates how the VG's generate longitudinal vortices and direct some of the flow into the tube wake. Meanwhile, the tube itself creates transversal vortices, which are shed behind it.

delta-winglets in a staggered configuration. This was done as a parametric study in the CFD software ANSYS Fluent and found the given location to be best at 20° and a winglet aspect ratio of 2. Investigation of the angle of attack and winglet variation was also done by He et al. (2012). They looked at the effects of splitting the VG's into two, smaller delta-winglets in an inline configuration, and with the holes in the lanced fin included. This was done as a parametric study in ANSYS Fluent and found that the objective function decreased for split VG's.
Different winglet types were also investigated by Saha et al. (2014) that also looked at common flow up vs. common flow down configuration of the vortex generators for rectangular and delta shaped winglets. This was done by solving the full unsteady Navier-Stokes equations using Marker and Cell algorithm. The conclusion was that rectangular VG's perform better than the delta shaped ones. In regards to the position of VG's a common flow down configuration gave the best results. Khanjian et al. (2017) investigated the effect of the roll angle on a rectangular winglet pair in a laminar channel as a parametric study in the CFD software STAR-CCM+. They found that the optimal angle went from 90° to 70° when increasing the Reynolds number from 465 to 911.
Other authors decided to investigate both the height and length of the vortex generators, instead of looking at type or aspect ratio. One was Välikangas et al. (2018), that investigated the potential heat transfer enhancement by the implementation of vortex generators in herringbone fin and tube heat exchangers, with the lanced hole in the fin included. This was done by a conjugated heat transfer CFD study conducted in FOAM-extend 4, with $k-\omega$ SST to model turbulence. The conclusion was that VG's with a height equal to 0.6 times the channel height and length of 0.5 times the tube diameter, produced the highest thermal performance increase,

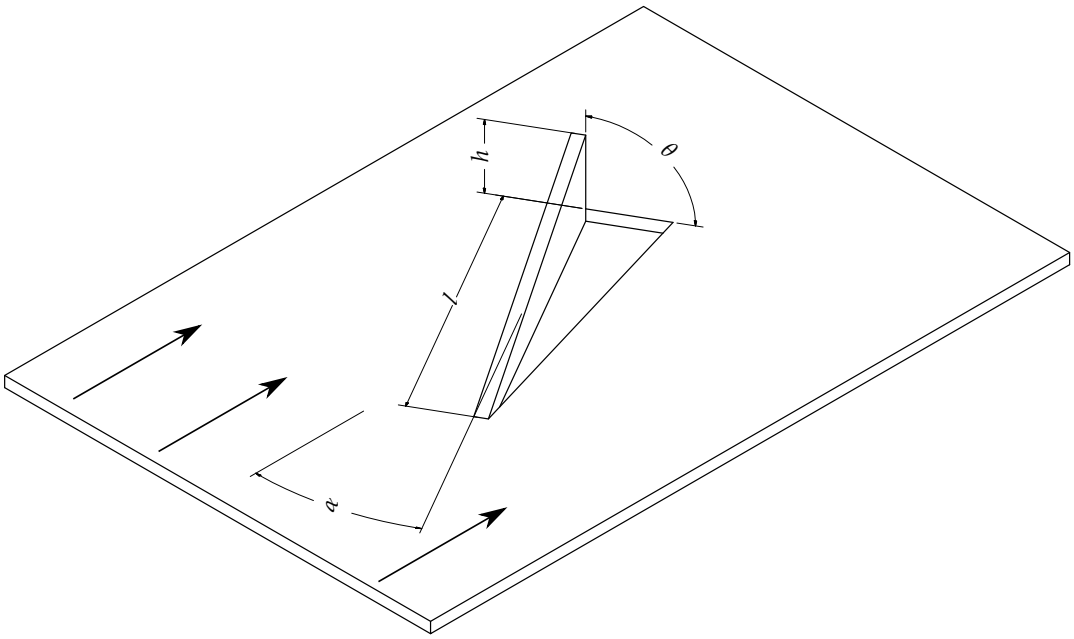


Figure 1.3: Relevant dimensional parameters for a DVG: The angle of attack α, the roll angle θ, the length of the VG l, and the height h.

of 5.23%. This was found based on a parametric study. They also concluded that the modelling of conjugate heat transfer is essential for accurate heat transfer predictions.
Another article was from Qian et al. (2018) that did a parametric study for rectangular vortex generators for a plain fin. Aside from length and height, the angle of attack was also instigated. This was done by CFD analysis in ANSYS Fluent 13 with $k-\varepsilon$ for turbulence modelling. The simulations were conducted on a periodic domain. The conclusion was that longer VG's performed better than smaller ones and that a large angle of attack helped to disrupt the wake zone behind the tubes.
Another parameter to investigate further is the placement of the vortex generators, which was done by Arora et al. (2016). The variables of interest were angle of attack as well as placement of the VG's. Twenty-two possible locations were investigated, taking into account infeasible placements due to the angle of attack. The study was conducted by CFD in ANSYS Fluent 14 with RNG $k-\varepsilon$ for turbulence modelling. The results were validated with experimental data. Results indicate that two types of optimal setup exist, one with the lowest investigated angles, being 15° and 30°, and one for 45° and 60°.

1.3 optimisation methods

Another category of articles looks into the problem of optimising the heat exchanger with many variables and data points. Some utilize the j/f factor as well, while others implement multi-objective optimisation, solving for both low friction and high heat transfer.

The Taguchi method was used by Zeng et al. (2010) to optimise delta winglet VG's in a staggered configuration for fin pitch, longitudinal and transversal tube pitch, VG length and height, and angle of attack. The lanced hole in the fin was also included. The investigation was conducted using ANSYS Fluent, and the model validated through experimental data. The analysis was conducted on an $L_{18}(2^1 \times 3^7)$ orthogonal array.
Salviano et al. (2015) investigated a delta-winglet in staggered tube configuration, with variable winglet position, angle of attack, and roll angle in ANSYS Fluent with $k - \omega$ SST to model turbulence. When optimising using a Genetic Algorithm (GA), the option of using a surrogate model as Response Surface Model (RSM) compared to Direct optimisation (DO), showed that DO gave a better solution for the given objective function. The response surface was generated from 385 points from Latin Hypercube Sampling. In another article, Salviano et al. (2016) compared the GA to the Nelder-Mead SIMPLEX optimisation method, adding three variables for the shape of the winglet for the same CFD method. They found that the SIMPLEX method converged in a third of the time, compared to GA, as well as optimal shapes and locations for VG's in both inline and staggered configuration. This was again done for single-objective optimisation.
The use of a surrogate model was also investigated by other articles. Tang et al. (2019) optimised rectangular VG's for elliptical tube H-fin heat exchangers for length, height, position, and angle of attack, using RSM validated through multiple regression. The model was solved as multi-objective optimisation using Non-dominating Sorted Genetic Algorithm (NSGA II), finding the Pareto optimal set of solutions. Lemouedda et al. (2010) followed the same general procedure for delta-winglet VG's with both staggered and inline tube configuration, varying angle of attack, and Reynolds number. The RSM was determined using the kriging method, and NSGA II solved the multi-objective problem for the Pareto optimal set.

1.4 Summary

To give an overview of the literature study, articles investigating tendencies using a parametric study include Reynolds number and 1-3 other parameters. These include angle of attack α, roll angle θ, VG length l and height h, VG aspect ratio, winglet type, or VG position. These approaches find tendencies and improvements but not optimal points.
Articles implementing optimisation algorithms either take a single-objective approach, using Taguchi S/N-ratio or another combined performance evaluation criteria or use a surrogate model to determine a Pareto-optimal set. These studies vary in the number of samples used for generating surrogate models, as more samples require proportional computational resources. Most surrogate models were determined using less than 100 samples, except for Salviano et al. (2016), which used 3080, along with a direct solution. The overview can also be seen in table 1.1.

Around half of the investigated articles implemented conjugate heat transfer, which Vä-

Table 1.1: Summary of investigated parameters in existing literature. Positioning of the VG is abbreviated to Pos. Middle line separates parameter studies and optimisation studies. *Investigated Aspect ratio. **Investigated Fin pitch.

Source	Re	α	θ	l	h	Pos.	Winglet type
Lei et al., 2010*	X	X					
He et al., 2012	X	X					X
Khanjian et al., 2017	X		X				
Välikangas et al., 2018	X			X	X		
Arora et al., 2016	X	X				X	
Saha et al., 2014	X	X				X	X
Qian et al., 2018	X	X		X	X		
Zeng et al., 2010**		X		X	X		
Salviano et al., 2015		X	X			X	
Salviano et al., 2016		X	X			X	X
Tang et al., 2019		X		X	X	X	
Lemouedda et al., 2010	X	X					

likangas et al. (2018) concluded to be essential, in their model. Salviano et al. (2016), and Tang et al. (2019) are the articles implementing it and also investigating optimisation. The lancing hole in the fins from the VG is not always included, perhaps due to the production approach of the VG's. Only three of the mentioned articles include them; He et al. (2012), Välikangas et al. (2018), and Zeng et al. (2010). Of these, only Zeng et al. (2010) investigated it using an optimisation algorithm. The applied Taguchi method is a single objective approach, meaning that a multi-objective approach has not been attempted, according to investigated literature. This could be an opportunity for further investigation.

Chapter 2

Problem analysis and description

This thesis seeks to optimise the design of Vortex Generators (VG's) in the given fin-tube heat exchanger using an OpenFOAM CFD model. Included in the model will be the hole lanced in the fin to create the VG, as well as conjugate heat transfer in the fluid-solid interface. optimisation will be done using a surrogate model, allowing for multi-objective optimisation to be carried out without high computational cost. This will result in a set of Pareto-optimal solutions. The combination of these goals is not apparent in the investigated literature. Sample solutions from the surrogate model will then be validated in the CFD model to ensure they are close to optimal. This leads to the following research question:

Which combination of the VG's placement, angle, height, and length in staggered fin-tube heat exchangers gives an optimum between maximal heat transfer and minimal pressure drop?

2.1 Describing the problem

As mentioned in the introduction, the task given for this thesis by Vestas aircoil is to implement VG's to increase the heat transfer for a charged air cooler. The vortex generators will be mounted on the fins in the cooler. Therefore it is of interest to determine a suitable simulation domain based on the fin geometry. The domain chosen can be seen in figure 2.1:
This choice of domain is done under the assumption of periodic behaviour and reduces the computational domain considerably. With the domain determined, the geometric parameters are introduced; these can be seen in figure 2.2.
Parameters that will remain constant in this thesis are as follows: Domain length L, which is equal to two times the longitudinal tube pith, domain height H which is equal to half of the transversal tube pitch, fin pitch F_p, channel height S, and inner and outer tube diameter D_i and D_o. Parameters that will be taken as variables are X- and Y-coordinates of the VG tip d_x and d_y, angle of attack α, VG height h, and VG length l. To limit the number of available parameters, the VG's will have identical parameters and locations relative to their respective tube. This means that the right half of the computational domain is similar to the left half mirrored horizontally. The values for the parameters are listed in table 2.1:

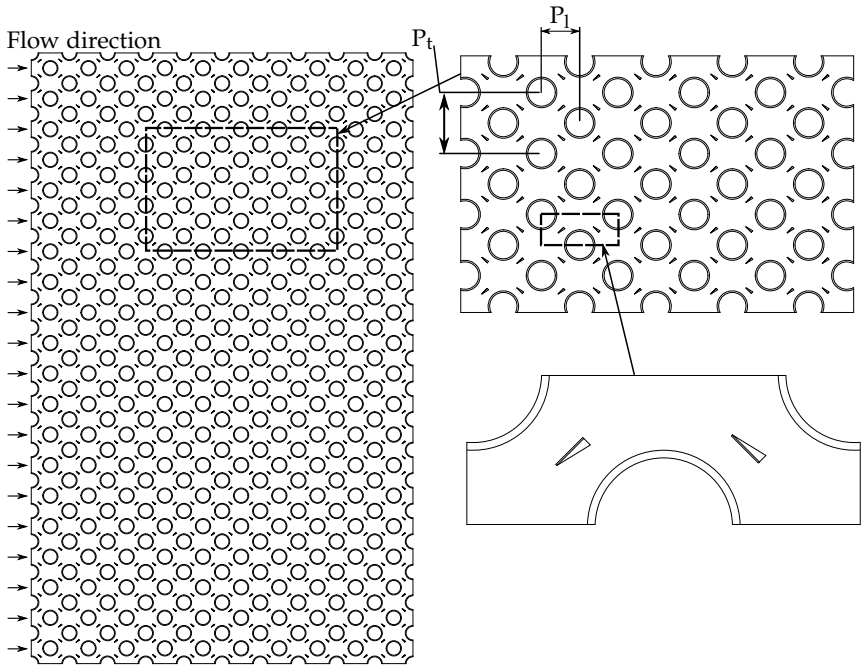


Figure 2.1: Top-down view of tube bank fin, as well as the zoomed-in view of the domain piece of interest.

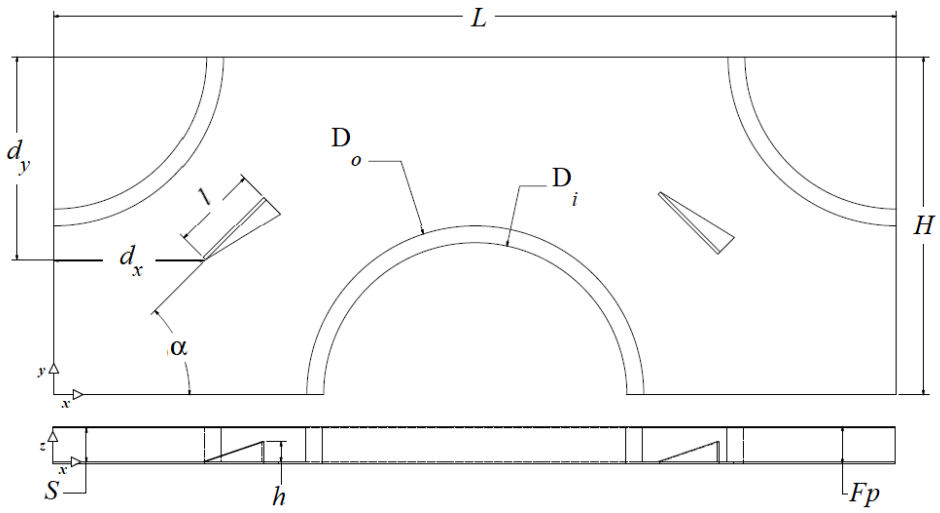


Figure 2.2: Schematic of the geometry in the computational domain.

2.1.1 Characteristic parameters

To evaluate the performance of the geometry, a couple of characteristic parameters are introduced. To describe the flow regime, the Reynolds number is included, to evaluate heat transfer, the Nusselt number is used, and to evaluate pressure drop the loss coefficient f. The description and calculation of these parameters are contained in this section.

Table 2.1: Geometrical values for the design space

Parameter	Symbol	Value [mm]
Height	H	20
Length	L	50
Diameter, inner	D_i	18
Diameter, outer	D_o	20
Channel height	S	2
Fin pitch	F_p	2.2

Reynolds number

The Reynolds number is calculated based on the velocity of the lowest cross-sectional area u_{max}.

$$Re = \frac{u_{max} \cdot D_o}{\nu} \tag{2.1}$$

where ν is the kinematic viscosity and the outer tube diameter D_o is used as the characteristic length.

Nusselt number

The evaluation of heat transfer is done based on the Nusselt number, which is defined as:

$$Nu = \frac{h_c \cdot D_o}{\lambda_f} \tag{2.2}$$

where λ_f is the thermal conductivity of the fluid and h_c is the convective heat transfer coefficient
Including Newton's law of cooling and Fourier's law of heat conductions, to describe the relationship between heat convection and conduction yields:

$$\frac{h_c}{\lambda_f} = -\frac{\mathrm{d}T}{\mathrm{d}n}\frac{1}{\Delta T} \tag{2.3}$$

Substituting this into equation 2.2 gives:

$$Nu = -\frac{\mathrm{d}T}{\mathrm{d}n}\frac{D_o}{\Delta T} \tag{2.4}$$

The wall normal temperature gradient is calculated as an area-averaged value, based on the tubes and fins surface area in contact with the fluid:

$$\frac{\mathrm{d}T}{\mathrm{d}n} = \frac{\sum_{i=1}^{N}\left(\frac{\mathrm{d}T}{\mathrm{d}n}\right)_i A_i}{\sum_{i=1}^{N} A_i} \tag{2.5}$$

where N is the total numbers of faces on the tubes and fins, and i is the face number. The overall temperature difference ΔT can be described similarly:

$$\Delta T = T_{wall} - T_{bulk} = \frac{\sum_{i=1}^{N} T_i A_i}{\sum_{i=1}^{N} A_i} - \frac{\sum_{j=1}^{M} u_{1,j} T_j A_j}{\sum_{j=1}^{M} u_{1,j} A_j} \tag{2.6}$$

where M indicates the total number of faces at the inlet and j is the face number, the velocity component in the streamwise direction, is denoted by the subscript 1.

Loss coefficient

The loss coefficient is used as a measure to determine the pressure drop of a certain geometry. Based on the work by Esmaeilzadeh et al. (2017) this can be expressed as:

$$f = \frac{2 \cdot D_o}{\rho \cdot u_{bulk}} \frac{\mathrm{d}P}{\mathrm{d}x} \tag{2.7}$$

where $\mathrm{d}P/\mathrm{d}x$ is the streamwise pressure gradient throughout the domain and u_{bulk} is the bulk velocity across the domain.

Performance evaluation ratios

The overall potential performance enhancement is determined based on the relations between Nusselt number and loss coefficient of a given design, compared to the reference case, which for this thesis is a plain fin geometry without any vortex generators or holes in the fin.

$$e_{Nu} = \frac{Nu}{Nu_0} \tag{2.8}$$

$$e_f = \frac{f}{f_0} \tag{2.9}$$

where the subscript 0 indicates that the value is the one from the reference geometry.

2.1.2 Materials and properties

For materials, the fins consist of copper, while the tubes are of Cu-Ni 90/10. Their properties will be given at the wall temperature of 25°C. The fluid will be modelled with the properties of dry air, treated as an ideal gas at the working pressure of 4 bar and at bulk temperature.

Temperature-dependent variables As the bulk temperature can vary, the temperature dependent properties will be investigated in a temperature range from 0°C to 100°C. Values are taken from the Engineering Equation Solver (EES) library. The normalized properties for air can be seen in figure 2.3, where it is apparent that the heat capacity can be taken as a constant value. The values and linear functions describing their dependencies are shown in table 2.2.

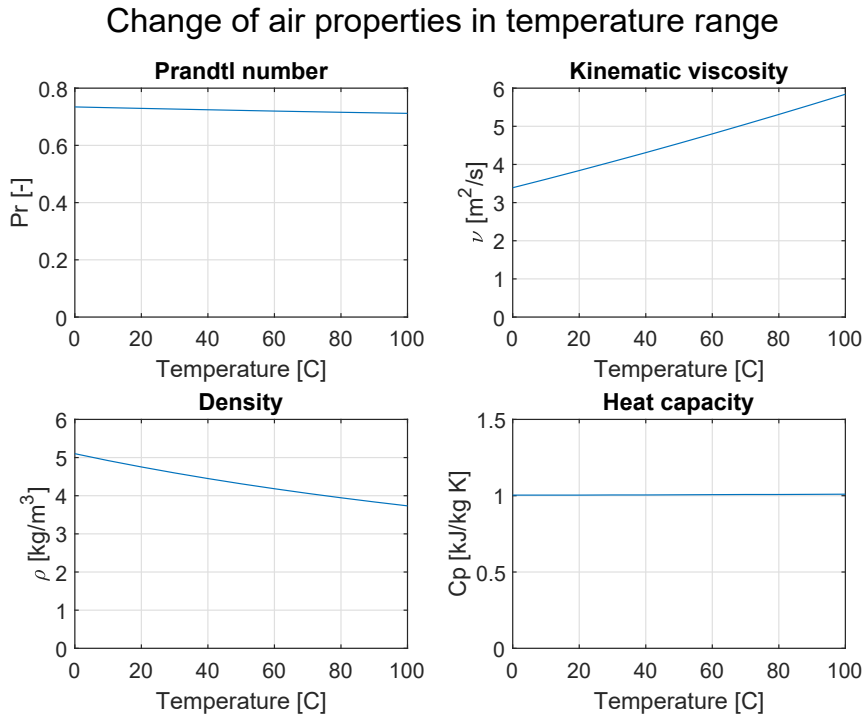


Figure 2.3: Properties for air in the temperature range of $0 - 100°C$.

2.2 Assumptions

When setting up the CFD model, several assumptions have been made to reduce complexity. Turbulence models may also need to be determined, depending on the given case and computational ambition. The choices are explained in the list below.

Fixed geometry The case in this thesis is given as optimisation of an existing product, resulting in some of the parameters being fixed from a production standpoint. Due to the proprietary nature of the case, parameters have been changed and/or generalized. The fixed parameters relate to the size and position of both the fins and tubes.

Flow regime Based on the already established tube bank design, the flow regime is well over a Reynolds number of 10000. This sets the flow regime to be modelled as a turbulent flow.

Steady-state Investigations done by Moulinec et al. (2004) indicate that transverse vortices generated by the tubes can be neglected for a staggered tube configuration at a Reynolds number of 6000 or higher, if the following statement is fulfilled:

$$P_t \leq 2 \cdot D_o \tag{2.10}$$

The geometry chosen in this work, fulfils this statement, with a transverse pitch of 40 mm and a tube diameter of 20 mm.

Table 2.2: Thermophysical properties for used materials. T is in unit K. Cp for Cu-Ni 90/10 is evaluated at 16.85°C.

Parameter	Value	Unit
Fins (Copper)		
Initial temperature	293	K
Conductivity	396.5	$W/(m \cdot K)$
Density	8958	$kg/(m^3)$
Heat capacity, cp	0.3893	$kJ/(kg \cdot K)$
Tubes (Cu-Ni 90/10)		
Inner tube wall temperature	293	K
Initial temperature	293	K
Conductivity	51.76	$W/(m \cdot K)$
Density	8897	$kg/(m^3)$
Heat capacity, cp	0.377	$kJ/(kg \cdot K)$
Dry Air (Ideal gas)		
Prandtl number	$0.7338 - 0.0002 \cdot T$	-
Turbulent Prandtl number	0,85	-
Reynolds number	≈ 10755	-
Streamwise bulk velocity, u_s	1.6	m/s
Streamwise temperature gradient	-50	K/m
Initial temperature	303	K
Working pressure	4	bar
Density	$5.0333 - 0.0136 \cdot T$	$kg/(m^3)$
Heat capacity, cp	1,005	$kJ/(kg \cdot K)$
Kinematic viscosity	$(3.3525 + 0.0245 \cdot T) \cdot 10^{-6}$	m^2/s

As this case will handle larger Reynolds numbers, the assumption of no vortex shedding is deemed acceptable, making it possible to approximate the flow as steady state.

Incompressible Compressibility of flow depends on the speed at which it flows. At a Mach number of 0.3 the density will encounter a change of around 5%. As the flow speeds, in this case, will be in the range of 2 m/s from the established design, compressibility can be approximated to be insignificant, and the flow will be modelled as incompressible.

Periodic behaviour As the modelling area will only be a fraction of a larger tube bank; it is necessary to know how similar the flow can be expected to be in a single segment. According to data from Balabani and Ylannekis (1997), flow in a staggered tube bank starts showing periodic behaviour after 4-5 tube rows.

Identical vortex generators As each periodic domain contains two VG's, optimal

configurations might exist were their parameters are different. Modelling parameters individually would drastically increase the complexity of the problem, so this thesis will only look at cases where the VG's have identical parameters.

No radiation Within each computational domain, a temperature difference of $\Delta T = 5$ K is expected between solid and fluid. As radiation heat flux between blackbody surfaces are determined as

$$Q = 5.67 \cdot 10^{-8} \frac{\mathrm{W}}{\mathrm{m}^2 \cdot \mathrm{K}^4} \cdot \left(T_{hot}^4 - T_{cold}^4 \right)$$

with emissivity 1, heat transfer through radiation is seen as negligible.

No gravitational force With the given temperature difference, the influence of buoyancy forces are investigated using the Richardson number in equation 2.11.

$$Ri = \frac{Gr}{Re^2} = \frac{g\beta \left(T_{hot} - T_{ref} \right) D_o}{u_{max}^2} \tag{2.11}$$

The value is found with gravity g, the average velocity at the smallest cross-sectional area u_max, characteristic length D_o, and a thermal expansion coefficient β. With a conservative estimate of $\beta = 3.43 \cdot 10^{-3} \mathrm{K}^{-1}$ at 20°C, the Richardson number is $8.6 \cdot 10^{-4}$. As buoyancy usually can be neglected at Ri = 0.1, the gravity term is neglected in this work.

Turbulence model Different approaches are available when handling the modelling of the turbulent flow. Available approaches are Direct Numerical Simulation (DNS), Large Eddy Simulation (LES) or Reynolds Averaged Numerical Simulation (RANS). DNS Resolves all turbulence scales, while LES resolves large scales and models scales, and RANS only resolves the mean flow and models all scales. The better resolution also requires a finer mesh, as well as having a higher numerical cost. RANS is chosen due to the computational requirement and the amount of simulations needed for optimisation. The trade-off in accuracy is deemed acceptable.

Closure model When working with a staggered fin and tube configuration, Bhuiyan et al. (2012) found that the use of the k-ω turbulence model best aligned with experimental results, comparing to the k-ϵ turbulence models.

Heat transfer approach

To get a better description of the heat transfer problem, it is treated as a conjugate heat transfer problem, solving the temperature field in both the solid and fluid phase. According to Välikangas (2016), this can be solved with faster convergence using the software FOAM-extend. The solver is "a steady-state solver for buoyancy-driven turbulent flow of incompressible Newtonian fluids with conjugate heat transfer, complex heat conduction and radiation."

Modelling of the tube bank geometry will, among others, be done with periodic boundary conditions based on the work of Patankar et al. (1978). The method is applicable as the temperature profile for each geometry segment is assumed to be identical, though for a temperature difference instead of a specific temperature.
To reduce the computational cost, the water flow inside the tubes is not modelled. Instead, the inner tube walls are assumed to be isotherm. This is done based on the assumption that the convective heat transfer coefficient is significantly larger than the one on the air side and this will result in an almost constant temperature.

Readers guide

To begin with, the nomenclature is presented from page VII, describing subscripts and symbols appearing in this thesis.
As this thesis will apply two methods in investigating the problem, some chapter can be read separately. The reading guide for the chapters can be seen in figure 2.4.
In chapter 1 the overall problem is described, along with the intended solution, as well as state of the art studies done on this subject. Chapter 2 describes how the problem, in this case, will be handled, and under what assumptions. In chapter 3 the bounds of the problem will be defined, as well as what parameters will be investigated. Constraints for the optimisation problem will also be made.
The CFD modelling part of the thesis is collected in chapter 4, starting with defining the governing equations used. Afterwards, the meshing strategy for the geometry is described as well as the determination of the appropriate refinement rate. Finally the implemented boundary conditions and discretization schemes will be listed.
The optimisation is described in chapter 5, where performance evaluation criteria, as well as an optimisation algorithm, is determined. As a surrogate model is required, 'design of experiment' techniques are investigated, and the viability of the chosen one is checked. Finally, the implemented optimisation algorithm is described.
In chapter 6, models and methods from the previous chapters are validated on either literature correlations for the CFD model or test functions for the optimisation algorithm. Results are presented and discussed in chapter 7, both for a sample model and the surrogate model. In the end, further studies are proposed. Finally, the thesis is concluded in chapter 8.

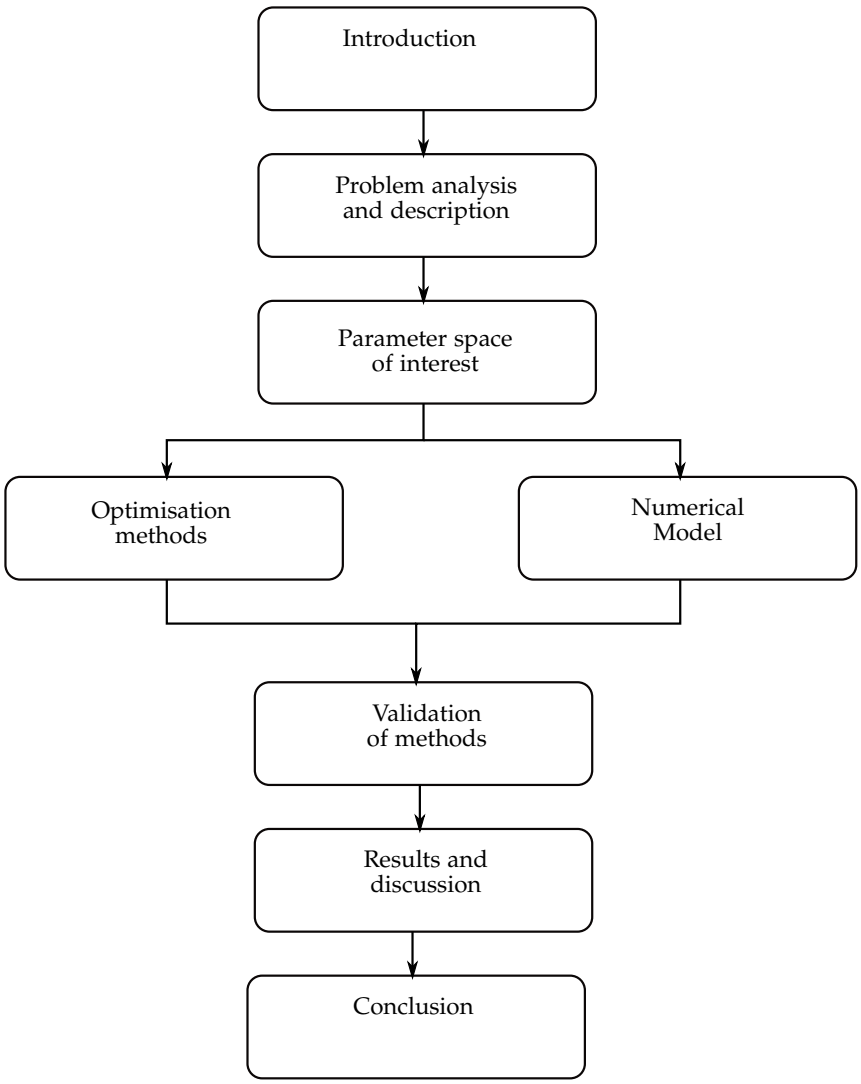


Figure 2.4: Reading guide to the chapters in this thesis

Chapter 3

Parameter space of interest

3.1 Significant variables

A large number of relevant parameters in a model helps to optimise all aspects of an optimisation problem, but also greatly increases the number of data points required to explore the entire design space. If computational power and time is a limiting factor, it makes sense to omit the less significant parameters in the investigation. As this report focuses on the application of Vortex Generators (VG's), parameters related to the heat exchanger, in general, will not be investigated. In the following part, the most common parameters, in this case, will be explained:

Angle of attack, α The angle between VG and the longitudinal flow.

Roll angle, θ This is the angle between the fin and the VG.

Longitudinal position of VG, d_x How far along the general flow direction the VG will be placed.

Transversal position of VG, d_y Where the VG will be placed perpendicular to the general flow direction.

VG length, l Increasing the length gives more surface to exchange heat and direct the flow, but also increases the pressure drop.

VG height, h The share of the channel height that the VG reaches. Most articles find an optimum near 0.6 of the fin pitch.

VG aspect ratio This value is a combination of VG height and length in one, in cases where the VG design is fixed.

The angles, length, and height can also be found in figure 3.1 and 2.2.
Among the listed parameters, only two of the three parameters determining the shape and size of the VG are needed, since the aspect ratio is dependent on the height and length. The roll angle is also a particular case, since only the article by Khanjian et al. (2017) investigates the effect, and does so in a different geometry. The article also found the optimal angle to be close to or at 90° at the investigated

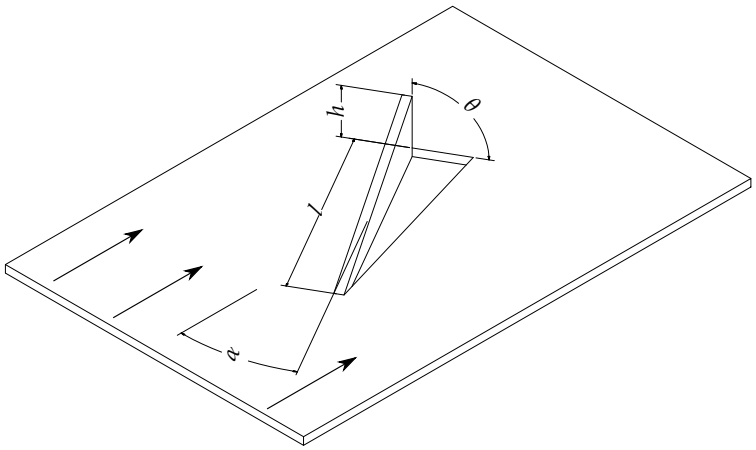


Figure 3.1: Angles and sizes of a arbitrary VG.

flow regimes. As the parameter may have some significance but will increase the requirement for computational power, it will not be investigated in this thesis. This leaves five parameters to be investigated as design variables: angle of attack, position of VG tip, VG height, and VG length.

3.2 Boundaries

To make sure a feasible design is found through optimisation, proper limits must be set up for the chosen design variables. As the variables only relate to the VG, limits must prevent it from getting too close to the boundaries of the computational area. This is also necessary to ensure proper mesh generation for the geometries. Several sources have had different approaches to limiting their variables to fit the given geometry. The limits of the selected articles can be seen in table 3.1.
For the position, Salviano et al. (2015) and Salviano et al. (2016) looked at both the x- and y-coordinate and later expanded the design region. They also implemented constraints as in equation 3.1, to prevent the centre of the VG from entering the tubes.

$$\sqrt{\left(d_x - d_{x,tube}\right)^2 + \left(d_y - d_{y,tube}\right)^2} \geq [1.1 - 1.6] \cdot \left(\frac{D_o}{2}\right) \tag{3.1}$$

The dimensions of the VG vary depending on design, rectangular- and delta-winglet. From Tang et al. (2019) and Zeng et al. (2010), the length was bounded due to geometry, though the height was more uniform. Salviano et al. (2016) had a larger area for length and height.
The angle of attack, α, was investigated by the same authors, as well as Arora et al. (2016). Most kept in the range $(15° - 60°)$, except for Salviano et al. (2015) and Salviano et al. (2016) that increased the range to $(\pm 75°)$ and $(\pm 180°)$ for their optimisation.

From these sources, initial boundaries can be defined. In case most optima are found on one of these boundaries, they can be extended with additional simulations. The chosen bounds can be seen in table 3.2.

Table 3.1: Boundaries investigated in the mentioned literature. Normalized with outer diameter, D_o, longitudinal tube pitch, L, or fin pitch, S.

	Salviano et al. (2015)	Salviano et al. (2016)	Tang et al. (2019)	Zeng et al. (2010)	Arora et al. (2016)
$d_{x,min}$	$1/3D_o$	$0.0111L$	-	-	-
$d_{x,max}$	$7/3D_o$	$0.99L$	-	-	-
$d_{y,min}$	$1/3D_o$	$0.0111L$	-	-	-
$d_{y,max}$	$17/15D_o$	$0.99L/2$	-	-	-
l	-	$2/3S$ - $6S$	$0.35S$ - $0.6S$	$1.3S$ - $2S$	-
h	-	$0S$ - $0.97S$	$0.35S$ - $0.7S$	$0.5S$ - $0.7S$	-
α	$\pm 180°$	$\pm 75°$	20° - 40°	30° - 60°	15° - 60°

Table 3.2: Boundaries for the design variables.

Design variable	Bounds
d_x	$0.1D_o$ - $0.8D_o$
d_y	$0.1D_o$ - $0.8D_o$
l	$1.2S$ - $3.0S$
h	$0.4S$ - $0.7S$
α	20° - 60°

3.3 Constraints

The geometry bounds work well to prevent the VG's from being placed outside of the square fin area. This cannot take the placement and size of the tube into account, so these are implemented as constraints in the same way as in equation 3.1. To take into account the different shapes and orientations of the VG's, constraints will be made for the VG origin and the two opposing VG points in the fin. This can also be shown in figure 3.2, where the three blue points representing the hole from lancing must not cross the red boundaries representing the two tubes. For two tubes, this results in 6 constraints, as shown in equations 3.2. The first three equations for the top left constraint, and the other three for the lower right constraint.

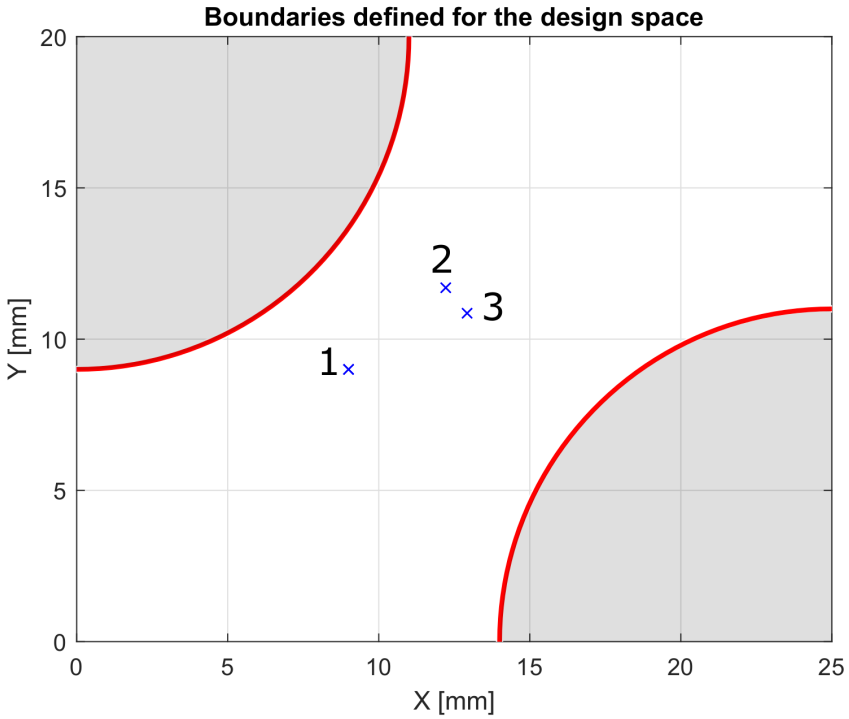


Figure 3.2: The boundaries for the lancing hole visualized as three blue points with constraints shown in red and grey.

$$0 \geq \left(1.1 \cdot \frac{D_o}{2}\right)^2 - \left[(d_x - 20)^2 + (d_y - 0)^2\right] \tag{3.2a}$$

$$0 \geq \left(1.1 \cdot \frac{D_o}{2}\right)^2 - \left[(d_x + l \cdot \cos(\alpha) - 20)^2 + (d_y + l \cdot \sin(\alpha) - 0)^2\right] \tag{3.2b}$$

$$0 \geq \left(1.1 \cdot \frac{D_o}{2}\right)^2 - \left[(d_x + l \cdot \cos(\alpha) + h \cdot \cos(\alpha - 90°) - 20)^2 \ldots + (d_y + l \cdot \sin(\alpha) + h \cdot \sin(\alpha - 90°) - 0)^2\right] \tag{3.2c}$$

$$0 \geq \left(1.1 \cdot \frac{D_o}{2}\right)^2 - \left[(d_x - 0)^2 + (d_y - 20)^2\right] \tag{3.2d}$$

$$0 \geq \left(1.1 \cdot \frac{D_o}{2}\right)^2 - \left[(d_x + l \cdot \cos(\alpha) - 0)^2 + (d_y + l \cdot \sin(\alpha) - 20)^2\right] \tag{3.2e}$$

$$0 \geq \left(1.1 \cdot \frac{D_o}{2}\right)^2 - \left[(d_x + l \cdot \cos(\alpha) + h \cdot \cos(\alpha - 90°) - 0)^2 \ldots + (d_y + l \cdot \sin(\alpha) + h \cdot \sin(\alpha - 90°) - 20)^2\right] \tag{3.2f}$$

Chapter 4

Numerical model

To evaluate the model parameters mentioned in chapter 2, being Nusselt number, and loss coefficient for the geometry, a conjugate heat transfer (CHT) model is used. The model chosen, is an incompressible, three-dimensional steady-state solver, that uses the Boussinesq's approximation to estimate the density of the fluids based on temperature variations. The pressure-velocity coupling is done with the SIMPLE-algorithm; the algorithm used can be seen in appendix IV.

4.1 Governing equations

The governing equations for the model are divided into two groups, on for the fluid and one for the solid.

4.1.1 Fluid

To describe the behaviour of the fluid through the domain, a set of five equations are solved. These equations contain the continuity equation, three momentum equations, and the energy equation. In this case the equation are solved for a three-dimensional incompressible flow, and look like the following:

Continuity equation

Based on the assumptions mentioned in chapter 2 the continuity equation for the flow can be stated as:

$$\frac{\partial(\rho u_i)}{\partial x_i} = 0 \tag{4.1}$$

Momentum equation

The three momentum equations for the three-dimensional problem can be expressed by the following equation:

$$\frac{\partial}{\partial x_i}(\rho u_i u_j) = \frac{\partial}{\partial x_i}\left(\mu \frac{\partial u_j}{\partial x_i}\right) - \frac{\partial p}{\partial x_j} + \frac{\partial P}{\partial x_1}\delta_{1,i} \tag{4.2}$$

To consider periodic behaviour, $(\partial P/\partial x_1)\delta_{1,i}$, pressure gradient source is added to the equation. This allows specifying the desired average velocity through the domain. Subscript 1 indicates streamwise direction, and $\delta_{1,i}$ is Kronecker's delta.

Energy equation

To determine the temperature in the fluid, the energy equation is added. To describe the thermal properties of the fluid, the following parameters are included: The density ρ, the specific heat capacity $C_{p,f}$ and the conductivity λ_f.

$$\frac{\partial}{\partial x_i}\left(\rho C_{P,f}\cdot(u_i T + u_s\gamma)\right) = \frac{\partial}{\partial x_i}\left(\lambda_f \frac{\partial T}{\partial x_i}\right) \tag{4.3}$$

Here $u_1\gamma$ is added to implement periodic heat transfer in; this is done based on the work done by Patankar et al. (1978), where u_1 is the velocity component for the direction, where the periodic behaviour is occurring. The temperature gradient in the flow direction is noted as γ. The implementation of the periodic heat transfer is done to investigate the temperature distribution in the domain.

4.1.2 Solid

To determine the heat transfer through the solid, the solid is modelled only based on heat diffusion:

$$\frac{\partial}{\partial x_i}\left(\lambda_{s,m}\frac{\partial T}{\partial x_i}\right) = 0 \tag{4.4}$$

where subscript s indicates solid, and m is used to describe the material number. This is done based on the fact that the conductivity in the two materials in the solid are different.

4.1.3 Turbulence modelling

To model turbulence, the $k-\omega$ Shear Stress Transport (SST) model is used. The argumentation for this choice is mentioned in chapter 2. Default values for used parameters can be found in table 4.1. The equations shown are for three-dimensional steady-state without dissipation and source terms:

k-equation

$$\frac{\partial}{\partial x_i}(\rho k u_i) = \frac{\partial}{\partial x_j}\left(\Gamma_k \frac{\partial k}{\partial x_j}\right) + G_k \tag{4.5}$$

ω-equation

$$\frac{\partial}{\partial x_i}(\rho \omega u_i) = \frac{\partial}{\partial x_j}\left(\Gamma_\omega \frac{\partial \omega}{\partial x_j}\right) + G_\omega \tag{4.6}$$

where the generation of turbulence based on mean velocity is described by G and Γ describes effective diffusivity, and is defined by:

$$\Gamma_k = \mu + \frac{\mu_t}{Pr_{t,k}}, \Gamma_\omega = \mu + \frac{\mu_t}{Pr_{t,\omega}} \tag{4.7}$$

where Pr_t indicates the respective turbulent Prandtl numbers and μ is the dynamic viscosity.
Turbulent dynamic viscosity is defined as:

$$\mu_t = a_1 \frac{\rho k}{max(a_1\omega, b_1 F_3 \mathbf{S})} \tag{4.8}$$

where $\mathbf{S}$ is the symmetric tensor.

Table 4.1: Default coefficients for the model.

α_{k1}	α_{k2}	$\alpha_{\omega1}$	$\alpha_{\omega2}$	β_1	β_2	γ_1	γ_2	β^*	a_1	b_1	c_1	F_3
0.85	1	0.5	0.856	0.075	0.0828	5/9	0.44	0.09	0.31	1.0	10.0	0

4.2 Meshing

As the optimisation will involve many different configurations, a script is made to generate a parametric mesh of the geometry. This should be able to ease the task of creating a good quality mesh for every required configuration of Vortex Generators (VG's). It is decided to have the mesh be a structured hex mesh.

4.2.1 Meshing strategy

As the thesis works with conjugate heat transfer, a mesh is required for both the fluid part and the solid part. As the two meshes should be conformal, the same strategy of meshing should be applied to them. To do so, the geometry is partitioned into blocks, that separately can be meshed to give an overall high quality mesh. To ease the meshing process and avoid highly skewed cells, the outer 10% of the VG tip is removed from the geometry. This will change the triangular prism into a hexahedron, allowing a structured mesh of only hexahedra. The trade-off in accuracy for an overall higher mesh quality is deemed acceptable.
The partitioning in the z-direction is done as four layers. The top and bottom layer consist of the solid fins and have the same thickness throughout. This also includes the lanced holes for the VG's in the fluid mesh. The middle layers, as seen on the fluid blocks in figure 4.2, are shaped according to the placement of the VG's. As they are each connected to a solid fin, only their joint separation surface changes shape. The surface starts at the smaller VG height, then increases across the VG, and decreases towards midway through the flow path. This allows it to repeat for the next VG, and concurrent VG's in the periodic flow.

Partitioning in the x- and y-direction is done, so a path runs along the flow direction, with different layers set to include VG's depending on the position of those. An example can be seen in figure 4.1.

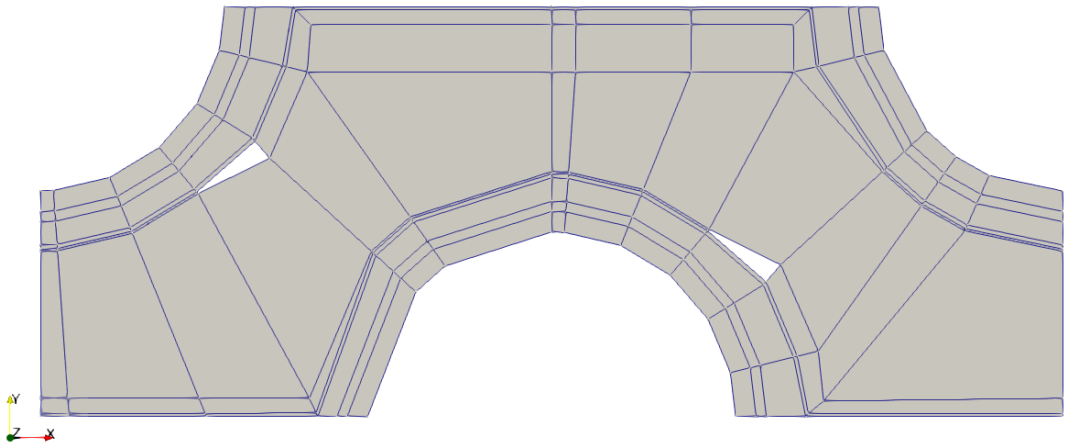


Figure 4.1: Sample top view of the blocks in the solid mesh. The holes in the mesh indicate wherein the VG is lanced.

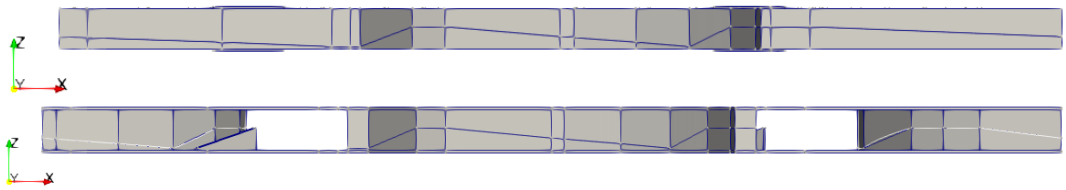


Figure 4.2: Sample side view of the blocks in the fluid and solid mesh, respectively.

For partitioning across the flow direction, a layer for each solid tube is made, as well as on for each VG with accompanying hole. The tubes are given a boundary layer, and the rest is filled in with buffer layers.

Partitioning in the flow direction is done with regard to the VG. The angle that the VG fill for the upstream tube is repeated to either side and made to fit with the geometry boundaries. One line from each tube quarter must also be set to coincide with the corner of the opposing VG layer, to keep the mesh as hex.

With all the blocks defined, a refinement ratio is put into the script to make sure the meshing of the blocks gives the cells around the same size. For the given mesh, a refinement ratio of 1 creates cells with a side length of approximately 0.25 mm in the x- and y-direction, with no refinement ratio implemented in the z-direction. Figure 4.3 shows the same view as figure 4.1, but with the meshing of the blocks implemented. Some of the blocks are modified to have a minimum amount of cells independently of the refinement ratio, as they otherwise would be too small for more than one cell in a given direction. The meshing of the fluid block is identical, except for the lack of tubes and the inclusion of the VG hole, hence it not being shown.

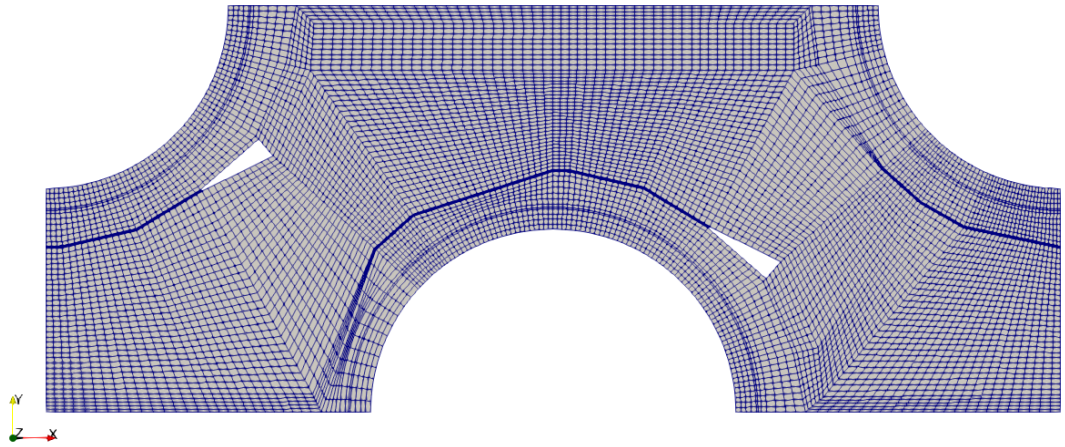


Figure 4.3: Sample view of the meshing of the solid. VG has an origin (7.25; 10.75), a length of 4.2 mm, a height of 1.1 mm, and angle of attack 40°. Blocks like the ones for the VG have smaller cells to accommodate the small block size.

4.2.2 Sample mesh parameters

After the mesh has been generated, they can be checked for different quality parameters using the OpenFOAM quality utility command `checkMesh`. The resulting values should be within certain guideline limits. The limits and values for the mesh in figure 4.3 can be seen in table 4.2. The parameters skewness and non-orthogonality are as described by Jasak (1996): An example is shown in figure 4.4 between two cells in 2D. Cell centres are indicated by I and II, while the dotted lines meet in the centre of the joint face. Skewness is skewness error on the face. It is defined as:

$$\text{Skewness} = \underline{m}/\underline{d}$$

with $\underline{\text{m}}$ and $\underline{\text{d}}$ being the length of the blue and green line, respectively. Non-orthogonality is the deviation from 90° for the angle between the red line and the green.

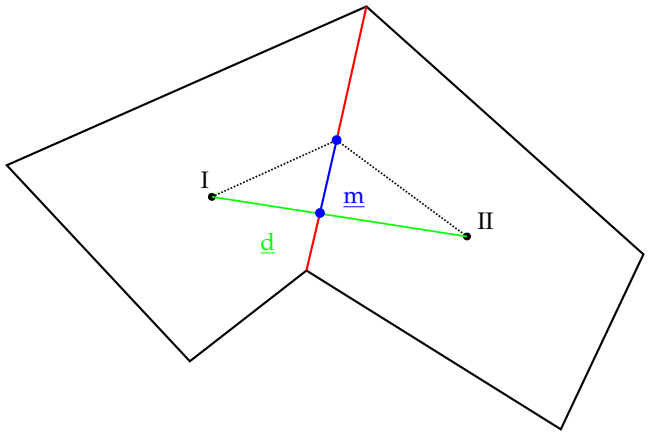


Figure 4.4: Visualization for describing skewness error on the face

Table 4.2: `checkMesh` maximum values for both meshes.

Parameter	Limit	Solid mesh	Fluid mesh
Cell count	-	27,618	82,362
Aspect ratio	1000	15.2	19.6
Skewness	4	1.663	1.714
Non-orthogonality	65	50.451	50.453

4.2.3 Grid convergence index

To identify how close the simulation results, based on a given mesh, are to the exact solution, the grid convergence index (GCI) is introduced. The method used was proposed by Roache (1997) and builds on the Richardson extrapolation. To determine the GCI, some parameters are needed. The first one is the refinement ratio r, which tells how much the mesh is refined uniformly in each direction. Based on the fact that the problem is three dimensional, r is given by:

$$r = \sqrt[3]{g_s} \tag{4.9}$$

where g_s is the overall multiplication of grid cells between refinements.
Taking the change in results, based on mesh refinement, into consideration, a measure for the converging speed can be obtained. This parameter is called the convergence rate p. Keeping the refinement ratio constant between different meshes, p can be described as:

$$p = \frac{\ln\left(\frac{\phi_3 - \phi_2}{\phi_2 - \phi_1}\right)}{\ln(r)} \tag{4.10}$$

where ϕ_3, ϕ_2 and ϕ_1 are solution parameters for three different mesh sizes; coarse, medium, and fine respectively. The solution parameters chosen for this case are Nusselt number and loss coefficient.
The two parameters that were just introduced can now be used to describe an estimate of the exact solution based on a generalized Richardson extrapolation:

$$\phi_{exact} = \phi_1 + \frac{\phi_1 - \phi_2}{r^p - 1} \tag{4.11}$$

As a measurement of how far away a solution is from the asymptotic numerical solution, the GCI is introduced. GCI should be as close as possible to zero, showing that the solution is within the asymptotic range. Taking a look at the fine mesh, the GCI can be expressed as:

$$GCI_{fine} = \frac{F_S|\epsilon|}{r^p - 1} \tag{4.12}$$

where F_S is a safety factor, which is recommended to be 1.25 when comparing three or more meshes, the relative error ϵ is given by:

$$\epsilon = \frac{\phi_2 - \phi_1}{\phi_1} \tag{4.13}$$

The same procedure can be used for the medium and coarse mesh, by changing the ϕ-values one index up.
Two different GCI-values can then be used to check if the solutions are within the asymptotic range of convergence, by the following equation:

$$C = \frac{GCI_{medium}}{GCI_{fine} \cdot r^p} \tag{4.14}$$

Here a C-value of unity indicates that the solutions are within the asymptotic range of convergence.

An arbitrary mesh is chosen for this case. The refinement ratio is not entirely consistent, with $r = 1.238$ between the coarse and medium mesh, and $r = 1.229$ between the medium and fine mesh. The solution parameters are the loss coefficient and the Nusselt number, given in table 4.3 along with mesh size and convergence time.

Table 4.3: Solution parameters for the GCI

Parameter	Description	Value			
		Coarse	Medium	Fine	Exact
n	Number of cells in mesh	145,980	277,268	514,708	
t	Time until convergence [hr]	9.5	24.4	53.9	
Nu	Nusselt number	67.9979	63.0155	61.4916	60.8201
ϵ	Relative error	7.91%	3.61%	1.10%	-
f	Loss coefficient	0.5963	0.6510	0.6929	0.8215
ϵ	Relative error	28.28%	21.71%	16.66%	-
		Nusselt number		Loss coefficient	
p	Order of convergence	5.640		1.260	
GCI_{fine}	GCI for medium and fine mesh	0.014		0.250	
GCI_{med}	GCI for coarse and medium mesh	0.044		0.347	
C	Asymptotic range of convergence	0.976		1.064	

For both solution parameters, the C-value is close to unity, which would indicate that the solution is close to the asymptotic range. An error band can also be estimated using GCI, according to Roache (1994). The error band is calculated as $\phi_{exact} \pm \phi_{exact} \cdot GCI$, and can be seen in figure 4.5 on page 28.

The GCI for the friction is higher, which may be caused by the lower order of convergence. The medium resolution is, however, still within the error band on the figure. For the friction, the medium mesh is deemed acceptable. When observing the sensitivity for the Nusselt number, only the fine resolution falls within the fine GCI error band, while the medium resolution is outside. As the GCI and relative error are below 5% for the Nusselt number in the medium size mesh in table 4.3, this model is also deemed acceptable. This is also caused by the fact that

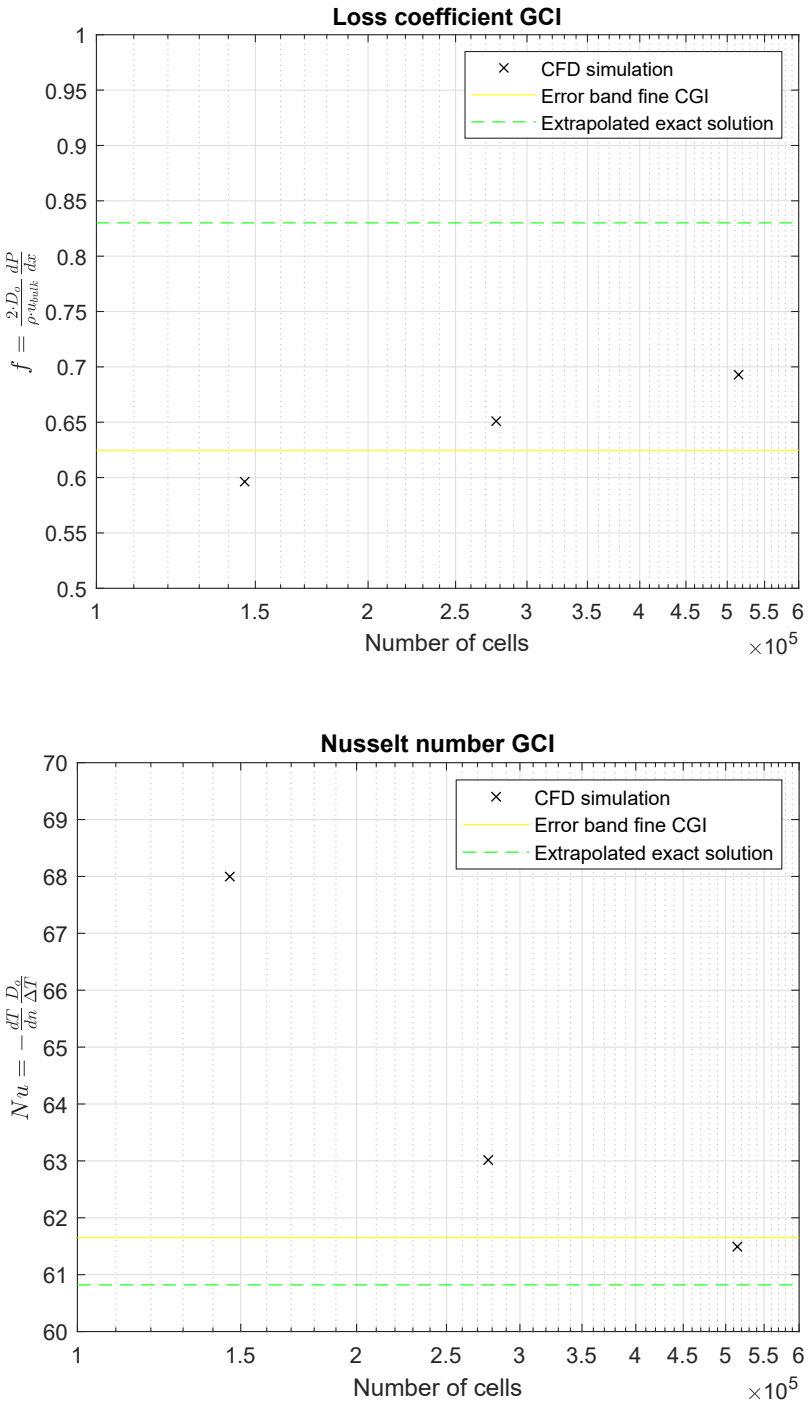


Figure 4.5: Sensitivity of the loss coefficient and Nusselt number to the grid resolution.

the computational demand for the finer mesh is infeasible within the given time frame.

4.3 OpenFOAM set-up

As many different options exist for using OpenFOAM, the selected are mentioned in this chapter.

4.3.1 Discretization schemes

The applied discretization schemes are listed in table 4.4. They can also be seen briefly be explained in Appendix B:

Table 4.4: Discretization schemes used in the OpenFOAM CFD model

Scheme type	Method applied
Gradient	Gauss <Interpolation>
Divergence:	
U/T/k/ϵ/ω	Gauss upwind
$\nu_{eff} \cdot devT(\nabla U)$	Gauss linear
Laplacian	Gauss <Interpolation> <Surface normal gradient>
Interpolation	Linear
Surface normal gradient	Corrected

4.3.2 Applied solvers

For every variable, a solver can be specified. As the solver finds the solution to a matrix set of equations, a preconditioner is used to make solutions more effective. These are listed in table 4.5 and are explained briefly in Appendix B.

Table 4.5: Solvers used for each parameter in the OpenFOAM CFD model

Variable	Solver	Preconditioner
P_{rgh}/P	PCG	DIC
U	BiCGStab	DILU
T_T	BiGCStab	Cholesky
G	PCG	DIC
ϵ	BiCGStab	DILU
ω	BiCGStab	DILU
k	BiCGStab	DILU

4.3.3 Boundary conditions

For both the solid and fluid mesh, boundary conditions must be defined. The description here is indicated in figure 4.6. The conditions are listed in table 4.6. The fluid/solid wall will be all surfaces were the solid and fluid are in contact. How the individual boundary conditions are set up for variables can be seen in Appendix B.

Table 4.6: Boundary conditions set for each type of model surface

Surface type	Boundary applied
Fluid mesh	
Opening Top/Bottom	Symmetry plane
Hole Front/Back	Periodic
Fluid/Solid wall	Coupled wall
In/Out	Periodic
Solid mesh	
Tubes	Wall
Fin Front/Back	Symmetry plane
Solid Top/Bottom	Symmetry plane
Fluid/Solid wall	Coupled wall
In/Out	Periodic

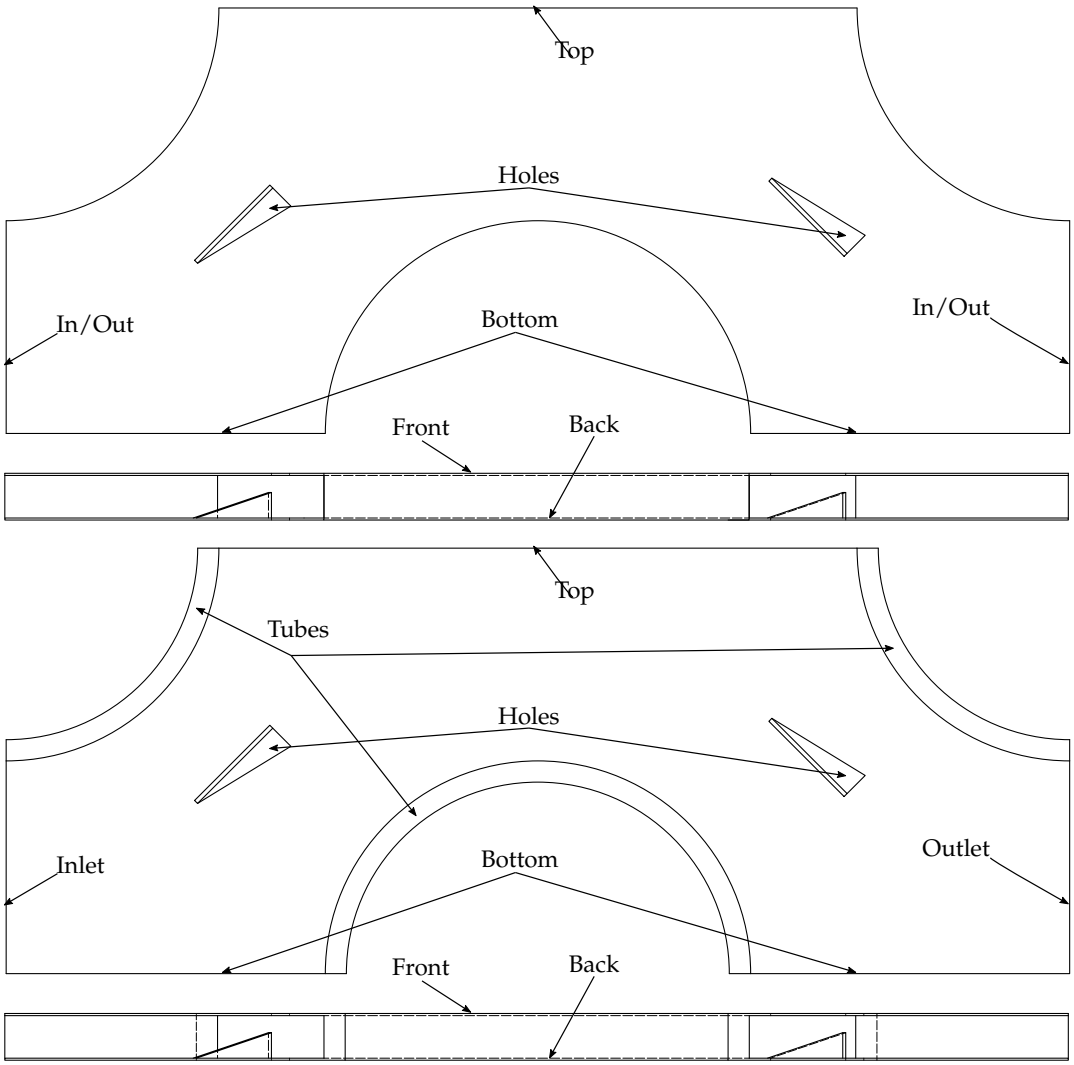


Figure 4.6: Boundaries defined for the fluid and solid mesh, respectively

Chapter 5

Optimisation methods

Since there are many ways to formulate an optimisation problem, there also exist many algorithms to do the optimisation. Some methods focus more on identifying relevant design parameters, such as a parametric study. These methods will usually find a way to improve the design but are not guaranteed to find an optimum. Other optimisers will be able to find an optimum, but the type will depend on whether the optimisation problem is given as a single- or multi-objective problem. Multi-objective optimisers have a higher computational cost at a high number of design variables, compared to those for single-objective optimisers. Because of this, multi-objective problems are often modified to single- objective problems by changing the performance evaluation criteria. This should be done carefully, as details will be lost in this simplification procedure.

Performance Evaluation Criteria First of all, the Performance Evaluation Criteria (PEC) must be determined. For a heat exchanger, interesting parameters could be the pressure loss across the heat exchanger and the heat transfer. CFD analysis allows these values to be determined directly, but in an academic setting, it may be advantageous to describe these effects as dimensionless numbers, like friction factor f and Colburn factor j, as seen in equations 5.1 and 5.2. These are found using the pressure drop ΔP, fluid density ρ, the average velocity at the smallest cross-sectional area u_{max}, outer diameter D_o, Nusselt Number Nu, Reynolds number Re, and Prandtl number Pr.

$$f = \frac{2 \cdot \Delta P}{\rho \cdot u_{max}^2} \cdot D_o \tag{5.1}$$

$$j = \frac{Nu}{Re \cdot Pr^{1/3}} \tag{5.2}$$

To further simplify the optimisation problem, the two factors can be combined to an expression for thermal performance, like equation 5.3, changing the multi-objective problem into a single objective one. Here the subscript 0 indicates a flow without VG's, and with Re and Pr set as constants, j can be replaced by Nu.

$$JF = \frac{j/j_0}{(f/f_0)^{1/3}} = \frac{Nu/Nu_0}{(f/f_0)^{1/3}} \tag{5.3}$$

Pareto optimality For multi-objective optimisation problems a predominant way of defining optimal solutions is through Pareto optimality. A feasible solution is Pareto optimal if there are no other points in the solution set that improves one objective function without worsening another. The set of Pareto optimal solutions is often called a Pareto frontier, and can be a line or surface depending on the amount of objective functions.

With the PEC determined, and Pareto optimality explained, a solution method can be sought.

Parametric study A popular and straightforward approach is to do a parametric study, investigating different configurations of the variables and noting which configuration gives the best results. This can determine a better solution but does not guarantee an optimum.

Taguchi The Taguchi method, as explained in Arora (2017), was developed to improve products and processes by reducing a loss function or maximize the Signal-to-Noise (S/N) ratio, where the signal is the property to improve, and noise is uncertainty. This results in a robust design. For instance, the S/N ratio can be defined as in equation 5.4 for N samples when minimizing the loss function.

$$S/N = -10 \cdot \log\left(\frac{1}{N}\sum_{i=1}^{N} JF_i^2\right) \tag{5.4}$$

In the Taguchi method, sampling is done in an orthogonal array, meaning that it will not consider the interaction between variables. For numerical simulations, artificial noise must be added to each data point. The solution will then be the combination of the highest S/N ratio for each variable.

Simplex The Nelder-Mead Simplex method, which is also explained in Arora (2017), is a direct search method for one objective function. It uses the idea of a simplex, a figure formed of n+1 points in n-dimensional space. The values are calculated in each vertex, and moving the figure in the direction of the optimal point. The solution will converge at a local optimum but is not guaranteed to find the global optimum. The relevant vertices are the best ($\mathbf{x}^L$), second worst ($\mathbf{x}^S$), and worst ($\mathbf{x}^W$). Four operations are used: *reflection*($\mathbf{x}^R$), *expansion*($\mathbf{x}^E$), *contraction*($\mathbf{x}^Q$), and *shrinkage*. As an example, a 2D simplex is shown in figure 5.1 with points from all operations except shrinkage. Here the original points form a triangle with the full red line, and the line between $\mathbf{x}^S$ and $\mathbf{x}^L$ will be there consistently. $\mathbf{x}^R$ shows the figure with a dashed red line for reflection, while $\mathbf{x}^Q$ and $\mathbf{x}^E$ indicate the options for expansion or contraction using a full black line. All the new points lie on the dashed black line, indicating the direction of translation for the operations.
First, the centroid ($\mathbf{x}^C$) and reflection point are found with their corresponding function value. If the reflection is better, expansion is tested, moving further along. If reflection fails, then contraction is tested between reflection and the worst point. This is either outside or inside contraction, depending on whether the current

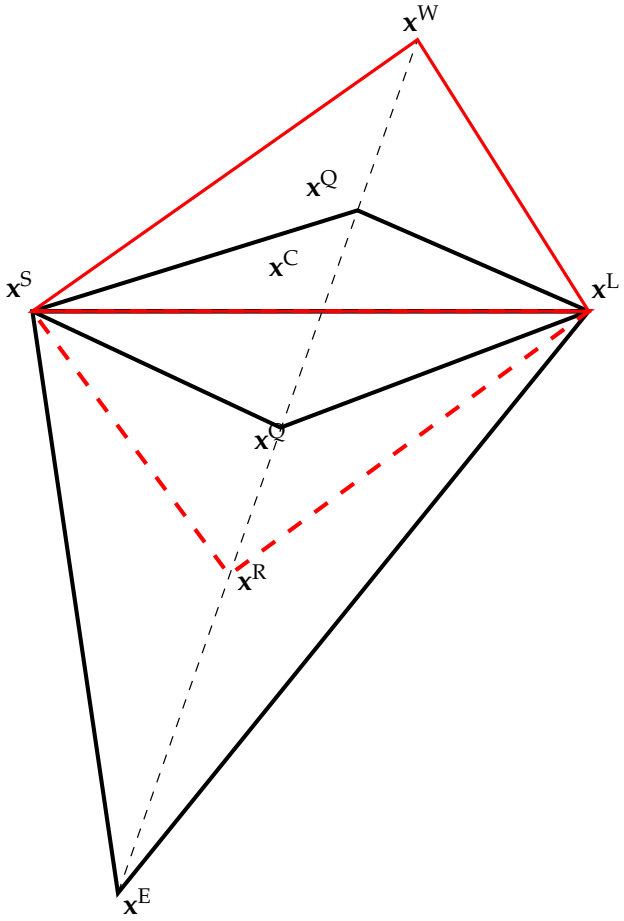


Figure 5.1: Different transformations explored by the Nelder-Mead Simplex Method. The original 2D simplex with reflection point is shown in red.

centroid is still in the simplex or not. If none of the other operations are successful, the simplex is shrunk around the best point, generating a whole set of n new points. All the operations can be seen in the equations 5.5, with χ being the individual operation coefficients, and χ_S indicating the shrink factor:

$$\mathbf{x}^C = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}^{(k)} \tag{5.5a}$$

$$\mathbf{x}^R = (1+\chi_R)\mathbf{x}^C - \chi_R\mathbf{x}^W \tag{5.5b}$$

$$\mathbf{x}^E = (1+\chi_E)\mathbf{x}^C - \chi_E\mathbf{x}^W \tag{5.5c}$$

$$\mathbf{x}^Q = (1+\chi_Q)\mathbf{x}^R - \chi_Q\mathbf{x}^C \tag{5.5d}$$

$$\mathbf{x}^Q = (1+\chi_Q)\mathbf{x}^C - \chi_Q\mathbf{x}^W \tag{5.5e}$$

$$\mathbf{x}^{(j)} = \mathbf{x}^L + \chi_S\left(\mathbf{x}^{(j)} - \mathbf{x}^L\right) \tag{5.5f}$$

The cycle then replaces the worst point, and this continues until a stopping criteria is met.

The solution of combining PEC to one parameter may not be sufficient in some optimisation cases, as the solution depends heavily on which variable is weighted

most. To prevent this, both functions have to be optimised at the same time. The algorithm must find Pareto solutions, where the improvement of one function leads to the degradation of the other, resulting in a Pareto optimal set of solutions. It is then up to the operator to decide which solution to take.

Monte Carlo A method based on random sampling for solving complicated problems. For optimisation, the method is not good at giving an exact solution but is an easy way to get a good estimate. The procedure is first to define the optimisation problem with bounds. The method then samples solutions for randomly sampled variables and collects them in a set. The best solution in the set is then taken as the solution. As this method was developed in the 1940s, as told by Eckhardt (1987), aspects of this method has later been used to develop more refined ways of applying randomness in optimisation algorithms.

Genetic Algorithm According to Arora (2017), Genetic Algorithm (GA) takes inspiration from nature by combining current designs to generate a new "generation" of designs, weighted in favour of the better function values and with added random mutations to cover more of the design space. This can also be expanded to include several PEC, allowing it to find a Pareto optimal set. As the solution is also a set of designs, the problem of not finding a global optimum is minimal.
When deciding how to generate the next set of designs, a fitness function can be used to give each design a value reflecting how good its function values are. Another way to determine is to rank the set on domination. A design is dominated if another design exists with both function values equal or better. Non-dominated designs are given a rank of 1 and are then removed. The new non-dominated designs are then given a rank of 2 and removed, and this process continues until all designs are ranked. The fitness is then inversely proportional to the rank. Elitist strategy can also be implemented, where a set amount of the best designs are carried over directly to the next set, to not lose any optimal solutions. Among the current GA's a popular one is the Non-dominated Sorting Genetic Algorithm (NSGA II) which has also been apparent in relevant literature by Lemouedda et al. (2010) and Tang et al. (2019). A problem with using GA is the amount of function calls they require. Even though the required population and generation number depend on the problem, several thousand function calls can be required. Given that this report deals with CFD, an amount of designs of this order of magnitude is inappropriate.

Response Surface Modelling To enable the use of algorithms that require many function evaluations, the complexity of the given functions must be decreased. As it is not reasonable to decrease the accuracy of the CFD model; a meta-model can be made, giving similar responses at a lower computational cost. The Response Surface Method (RSM), as explained by Arora (2017), can generate meta-models as e.g., a linear or quadratic function of variables. A function is generated that minimizes the error between the meta-model and sample points from the original

model. A quadratic approximation can look like equation 5.6, with n variables, the error ε and $a_{i,j}$ $i,j = 0,\ldots,n$ as coefficients.

$$\begin{aligned} f =& a_{00} + a_{10}x_1 + a_{20}x_2 + \cdots + a_{n0}x_n + a_{11}x_1^2 + \cdots + a_{nn}x_n^2 \\ &+ a_{12}x_1x_2 + \cdots + a_{n-1,n}x_{n-1}x_n + \varepsilon \end{aligned} \tag{5.6}$$

To generate a surface from k function evaluations, the coefficients and variables are converted to vector form:

$$\begin{aligned} &a_{00} \to d_0, a_{10} \to d_1, \ldots, a_{11} \to d_{n+1}, \ldots, a_{n-1,n} \to d_l \\ &x_1 \to \xi_1, \ldots, x_1^2 \to \xi_{n+1}, \ldots, x_{n-1}x_n \to \xi_l \end{aligned}$$

with l as the total number of linear and quadratic terms given as

$$l = 1/2n(n+1) + n$$

The surface coefficients can then be calculated by solving the matrix in equation 5.7, with ξ_{ij} $i = 1,\ldots,k$ $j = 1,\ldots,l$.

$$\begin{bmatrix} k & \sum_{i=1}^{k}\xi_{i1} & \sum_{i=1}^{k}\xi_{i2} & \cdots & \sum_{i=1}^{k}\xi_{il} \\ \sum_{i=1}^{k}\xi_{i1} & \sum_{i=1}^{k}\xi_{i1}^2 & \sum_{i=1}^{k}\xi_{i1}\xi_{i2} & \cdots & \sum_{i=1}^{k}\xi_{i1}\xi_{il} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{k}\xi_{il} & \sum_{i=1}^{k}\xi_{il}\xi_{i1} & \sum_{i=1}^{k}\xi_{il}\xi_{i2} & \cdots & \sum_{i=1}^{k}\xi_{il}^2 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_l \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{k} f_i \\ \sum_{i=1}^{k}\xi_{i1}f_i \\ \vdots \\ \sum_{i=1}^{k}\xi_{il}f_i \end{bmatrix} \tag{5.7}$$

One thing to note is that the number of sample points should be equal to or greater than the number of unknown coefficients (l+1). Sampling can be done at random, or e.g., by orthogonal arrays.

Kriging This method is also known as Design and Analysis of Computer Experiments (DACE). Aside from modelling a response surface, the method also incorporates the statistical uncertainty in-between sample points. This allows for the refinement of the response surface by investigating points with a high probability of improvement. Jones (2001) describes the different aspects of response surface optimisation, including ones that incorporate DACE. In engineering applications DACE has found limited use. According to Simpson et al. (2001) and Simpson et al. (1998) this might be explained by lack of readily available software, complexity in fitting the model, or the additional effort required. Giunta (1997) found that DACE was less accurate compared to RSM, even for highly non-quadratic test cases.

5.1 Choice of optimiser

With the given optimisation problem, it will be an advantage to be able to have the flexibility to choose between increasing heat transfer capacity and reducing pressure drop. This means that the solution strategy will have to be multi-objective.

This will exclude optimisation algorithms like Nelder-Mead that focus on one objective function. As optimisation is desired, methods that only work to improve will not suffice, leaving out a parametric study and the Taguchi method as well. This leaves the Monte Carlo, GA and DACE. All of these will require RSM, as the number of function evaluations in a CFD software will require too much time or computational power. The Monte Carlo method may be able to find optimal solutions, but as the process is random, it is not guaranteed, this method will not be used. As DACE is a more complex method and does not seem to give a proportional benefit, this report will be doing optimising using GA. As the response surface is an approximation, solutions from the Pareto-frontier can then be returned to the CFD model for verification of an optimum.

5.2 Design of Experiments

With multi-variable optimisation, it is essential to reduce the number of variables and to that extent, the number of evaluated configurations, to a manageable level. In chapter 3 it was decided that a reasonable amount of variables should be five in this case. To reduce the number of evaluated configurations, Design of Experiments (DoE) is a relevant subject to study.

5.2.1 Design methods

With the investigated parameters determined, a sampling method to efficiently capture the entire design space will be investigated. To limit the computational cost in acquiring the data, only model complexity up to quadratic will be investigated. The different DoE listed below come mainly from the book by Cavazzuti (2013).

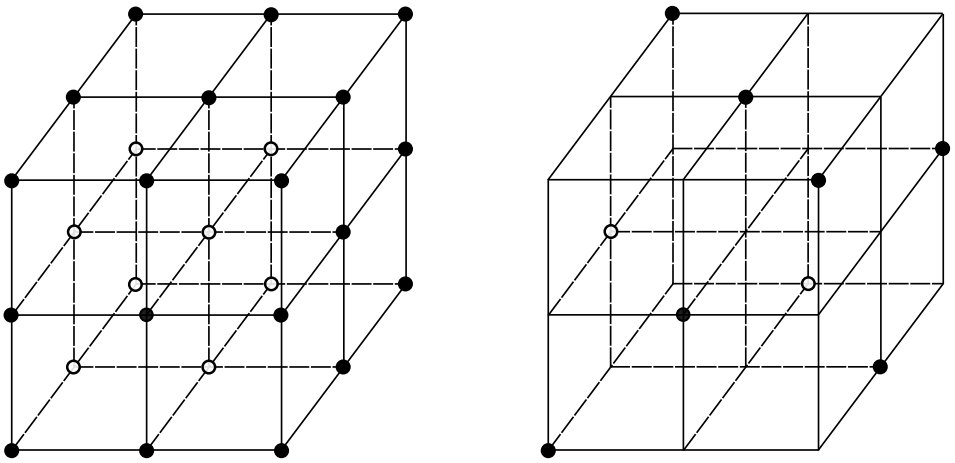
Full factorial design consists of all possible combinations of w factors for a given s levels for each factor. This results in a design cost of:

$$n = s^w$$

The method does not confound the effects of the parameters, but the sample size grows exponentially with the amount of factors and levels. It is efficient in parameters when these are used for polynomial response surfaces, as a 2^w design can give a complete bi-linear design, 3^w a complete bi-quadratic design, and so on. An example of a 3^3 factorial can be seen in figure 5.2a

Fractional factorial design, Khan (2013) This method seeks to reduce the full factorial design in varying degrees. For a fraction size of q, the fractional level is $1/2^q$. The method is confounding, meaning interaction factors cannot be separately determined from major factors in an experiment. This reduces the amount of design points, under the assumption that only lower order terms are significant, by having the higher order and interaction terms confounded by the lower order terms. If no higher order or interactions exist, this method gives the same result as a full factorial design.

$$n = 2^{w-q}$$



(a) Full factorial of 3 levels and 3 factors (3^3)

(b) Fractional factorial of 3 levels and 3-1 factors (3^{3-1})

Figure 5.2: Effect of factorial design on a cube of 3 variations (levels) of 3 design variables (factors)

The different levels are categorized in resolutions, depicted by a roman numeral, e.g.

- a resolution III design has the main effect aliased with the 2-factors effects. $(x_1 = x_2 \cdot x_3)$
- a resolution IV design has the main effect aliased with the 3-factors effects, and the 2-factors effects aliased with each other. $(x_1 = x_2 \cdot x_3 \cdot x_4 \quad x_1 \cdot x_2 = x_3 \cdot x_4)$
- a resolution V design has the main effect aliased with the 4-factors effects, and the 2-factors effects aliased with the 3-factors effects.

The higher the resolution, the better the results can be expected to be. An example of a 3^{3-1} design can be seen in figure 5.2b. Another example is shown in table 5.1 has the reduction of a 2^3 full factorial to a 2^{3-1} fractional factorial. The interaction between A and B is confounded as variable C, as the product of the levels gives an identity vector. The two-factor interactions are confounded by main effects, giving a resolution III design.

Table 5.1: Fractional factorial for a 2^{3-1} design. The products of the levels gives the identity matrix I.

Experiment number	Factor level $x_1(A)$	$x_2(B)$	$x_3 = x_1 \cdot x_2(C)$	$I = x_1 \cdot x_2 \cdot x_3$
1	-1	-1	$+1$	$+1$
2	-1	$+1$	-1	$+1$
3	$+1$	-1	-1	$+1$
4	$+1$	$+1$	$+1$	$+1$

Orthogonal array, Arora (2017) An orthogonal array is represented with the notation in equation 5.8, with N number of experiments, n design variable groups, s_i levels, k_i design variables in the ith group. As an example, $L_{18}(2^1 3^7)$ would be 18 samples of one two-level variable and seven three-level variables.

$$L_N \left(\prod_{i=1}^{n} s_i^{w_i} \right) \tag{5.8}$$

The functionality of an orthogonal array is named as such because if high, low and neutral points are changed to 1, -1, and 0 respectively, the dot product of any two columns in the array will be orthogonal (0). This method can significantly reduce the number of design points, but can also omit interactions between terms. An example of an $L_9(3^4)$ orthogonal array can be seen in table 5.2

Table 5.2: Orthogonal array for 4 design variables with each 3 levels $-1, 0, 1$.

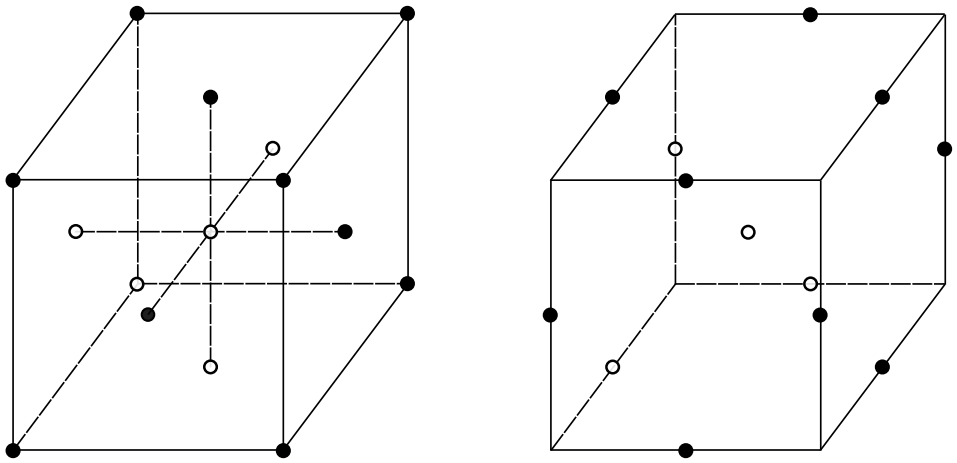
Experiment #	Design variables and levels			
number	x_1	x_2	x_3	x_4
1	-1	-1	-1	-1
2	-1	0	0	0
3	-1	1	1	1
4	0	-1	0	1
5	0	0	1	-1
6	0	1	-1	0
7	1	-1	1	0
8	1	0	-1	1
9	1	1	0	-1

Central composite design This design combines a two-level full factorial design with a central point and axial points from the centre to each of the factors. The resulting number of points required are:

$$n = 2^w + w \cdot 2 + 1$$

An advantage in this method is that it can be done as a continuation from a two-level full factorial screening that can have determined significant parameters. The design can be done with different distances for the axial points. Central Composite Circumscribed (CCC) has the axial points the same distance from the centre as the corner points. Central Composite Faced (CCF) the values remain on the same level as the corner points. If the corner points already reach the limits of the design space, CCF prevents the axial points from breaking these limits. An example of CCF can be seen in figure 5.3a.

Box-Behnken design This design only has central points as well as points on the centre of the edges of the given hypercube. For a given amount of factors, the required Box-Behnken points can be seen in table 5.3:



(a) Central Composite Faced design of three levels and three factors

(b) Box-Behnken design of three levels and three factors

Figure 5.3: Additional designs reducing 3^3 full factorial, while still containing enough information for a quadratic model

Table 5.3: Examples of required points in a Box-Behnken design

Factors	Experiments (w. centre point)	Coef. in quadratic model
3	13	10
4	25	15
5	41	21
6	49	28
7	57	36

An example of a 3-dimensional Box-Behnken design can be seen in figure 5.3b

Latin Hypercube The design space is subdivided into an orthogonal grid with N elements of the same parameter length. Within the grid, N sub-volumes are set so that along each dimension, only one sub-volume is found.

5.2.2 Summary of DoE Techniques

To decide on which type of DoE should be used, Cavazzuti (2013) has given a table with an overview of the relevant applications. Mentioned techniques are shown in table 5.4. Since this report will focus on quadratic response surface modelling, Box-Behnken is chosen as the DoE technique, as it reduces the number of required experiments, while still delivering the required solution of the design space.
The Box-Behnken design and Central composite methods both allow for quadratic models with interaction, as well as requiring a low number of experiments. The advantage of screening with Central composite does not contribute if all variables are significant, so the number of experiments will be the main choosing criterion. For 4 variables the number is the same, but for 5, Box-Behnken requires 41 compared to Central composites 43. For this reason, Box-Behnken will be the applied

Table 5.4: Summary of different DOE techniques

Method	Number of experiments	Suitable for:
Full factorial	s^w	Computing main and interaction effects, building response surface
Fractional factorial	s^{w-q}	Estimate the main and interaction effects
Orthogonal Array	Chosen	Compute main and quadratic effects, no interaction
Central composite	$2^w + 2 \cdot w + 1$	Building response surface
Box-Behnken	From table 5.3	Building quadratic response surface
Latin Hypercube	Chosen	Building response surfaces

DoE technique.

5.2.3 Feasibility of chosen design

To prevent the sampled designs from the Box-Behnken method from appearing in the tubes, they are visualized with the given boundaries in figure 5.4. Here a Vortex generator (VG) is shown in all given configuration, along with the hole acquired from lancing, in a clockwise direction. The tubes are also shown in red, with a blue margin of 10% extra to see if the VG will interfere with boundary meshing.
As can be seen in the figure, two designs are completely inside a tube, making them infeasible. For the two clusters of six designs in the top left, most are barely infeasible, while a few only infringe on the safety margin in blue. Of the mid right designs, one infringes on the margin area. Infeasible designs should not be removed, as this will give less information to the optimisation. Instead, the possibility of moving the points elsewhere is proposed.
The infeasible clusters are moved $0.25 \cdot D_o$ in the X and Y direction, towards the flow diagonal. The infeasible corner designs are moved in the same direction until they are free from the infeasible zone, with $0.5 \cdot D_o$ for the bottom right, and $0.75 \cdot D_o$ for the top left. These changes can be seen in figure 5.5. Only two of the 41 designs infringe on the safety margin, but as this is mainly put in place for ease of meshing, they should not cause problems in manufacturing. With the given levels and limits to variables, the applied design (with results) can be seen in appendix A.2.
It should be noted that the new design does not guarantee the same resolution of the objective space as the original Box-Behnken design. As the infeasible designs were moved to positions of interest, the resolution should hopefully still be satisfactory.

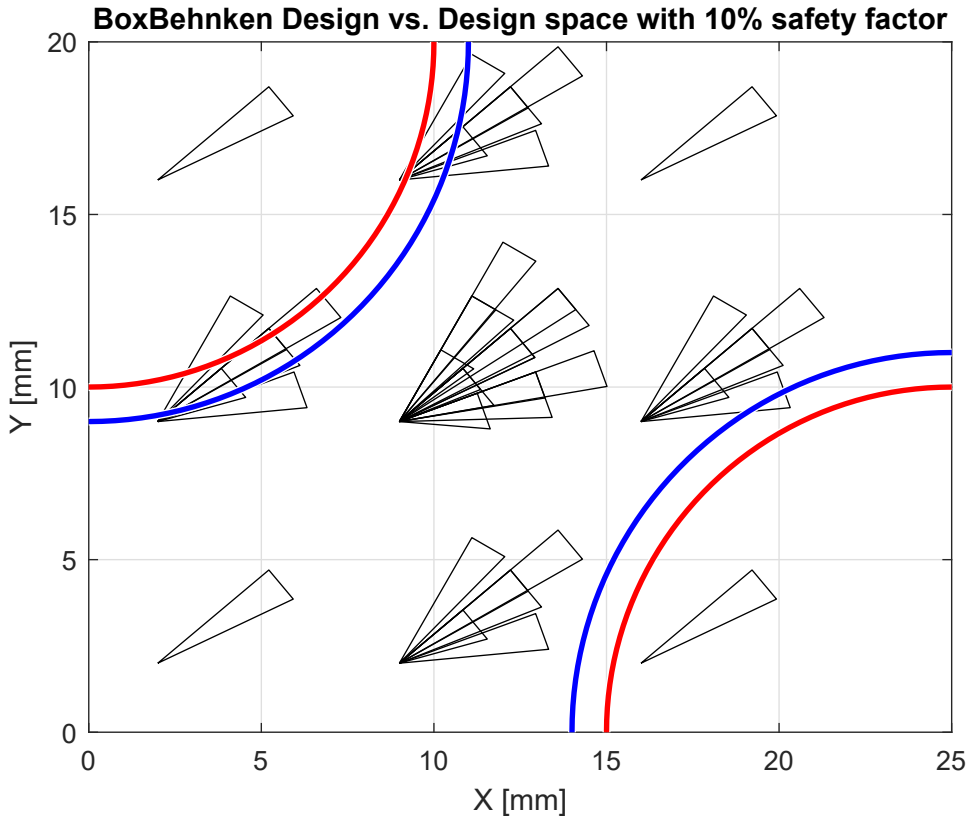


Figure 5.4: All possible VG configurations for the given boundaries and Box-Behnken design method. VG with holes are shown as black triangles, tubes in red, and safety factor in blue.

5.3 Non-dominated Sorting Genetic Algorithm II

A popular genetic algorithm that has been proposed by Deb et al. (2002) is the NSGA II. It is based on the original NSGA from Srinivas and Deb (1994), but improves it by reducing the sorting complexity, introducing elitism, and adding a crowding distance to prevent clustering on the Pareto frontier. At the time of the publication of the algorithm, it was found to outperform two other similar algorithms; Pareto-Achieved Evolution Strategy (PAES) and Strength Pareto Evolutionary Algorithm (SPEA). A flowchart of the implemented algorithm can be seen in figure 5.6. A MATLAB implementation by the name of NGPM (an NSGA-II Program in Matlab) was found from Lin (2011) and modified to fit the current purpose better. In the following sections, the relevant operations are explained.

5.3.1 Non-dominated Sorting Approach

To determine which solutions to promote as parents for generating the next generation of children, a 'fitness' value must be determined for the entire solution set. Fitness is determined by 'domination', where one solution is dominant over another if all its objective values are equal or better than the others. Each solution is given a scalar n_p showing how many other solutions dominate it, and a vector s_p with the identifiers of the solutions, it dominates. Solutions with $n_p = 0$ are given

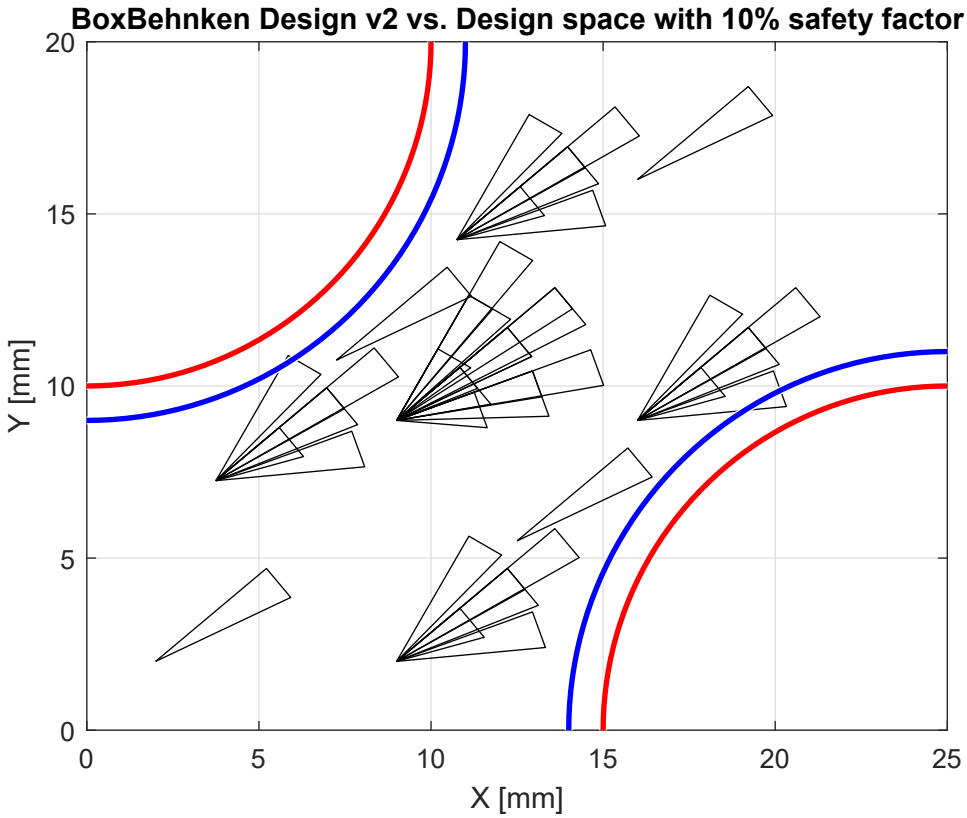


Figure 5.5: View of modified Box-Behnken design space to prevent infeasible designs. VG with hole is shown as black triangles, tubes in red, and safety factor in blue.

the rank 1. For the rest of the set, if a solution was dominated by one of rank 1, its n_p loses a counter. Solutions that now have $n_p = 0$ are given rank 2, and this continues until the solution set is ranked.

If constraints are also included in the objective space, the number of violated constraints and the sum of the violations must also be used. For ranking, infeasible solutions are dominated by feasible ones. If both solutions are infeasible, the one with the smallest violation sum is dominating.

Each rank forms a front in the objective space that will move toward the Pareto-front. To prevent these fronts from clustering, a crowding distance is calculated for each rank. Solutions with the highest and lowest objective values are the endpoints of the front and are given a crowding distance of infinity. For the rest, the distance is the sum of the normalized distance between neighbouring solutions. An example of how a set of feasible ranks would look like can be seen in figure 5.7. The crowding distances Δf is shown for the circled individual. It is taken as the sum of the normalized distances.

The entire population is then sorted according to the lowest rank, with the highest crowding distance resolving ties. The code for sorting can be found in appendix III.2.

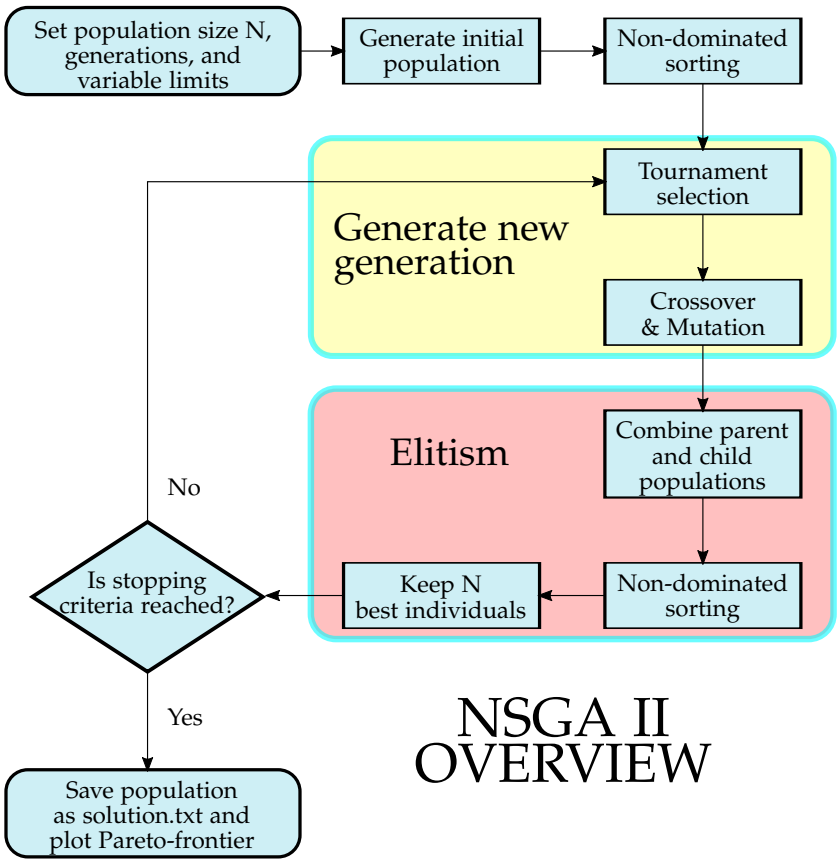


Figure 5.6: Flowchart of the implemented NSGA II. Dark blue boxes indicate dub-function calls.

5.3.2 Tournament Selection

With the parent solutions generated and ranked, the best are chosen for crossover and mutation. This is done using binary tournament selection. Two random solutions are compared, and the best are added to the mating pool. The evaluation criteria are primarily rank, followed by crowding distance in case of ties. The two parents are not removed from the set of potential parents, meaning that they can be chosen again. This process continues until there are parent solutions equal to the population. Constraints need not be considered here, as it is inherent in the ranking from the nondominated sorting. Matlab code for this part can be seen in appendix III.3

5.3.3 Crossover and Mutation

NSGA II utilizes the simulated binary crossover (SBX) operator for generating offspring. Crossover is set to be the major factor for generating a new set of designs, while mutation is only a minor part to expand the search area. In this implementation, for every variable of every parent in the mating pool, there is a 90% probability of crossover, while there only is a 10% probability of mutation. Crossover is done sequentially on the randomly sampled parent pool, taking two parents at a time. The same is then done for mutation, for the same pool already contain-

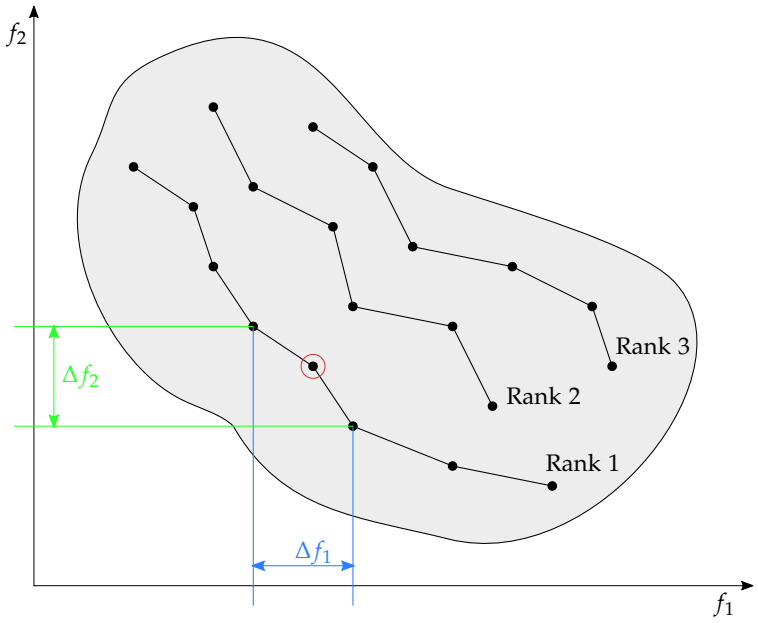


Figure 5.7: Different fronts generated during NSGA-II, and ranked according to domination. Crowding distance is shown for a single individual of rank one.

ing crossover results. This gives a 9% probability for both no operations and both operations. After each operation, the solutions are compared to the variable limits. Variations that exceed the design space limits are set to the limits, to prevent solutions that violate the design space. The code used in Matlab can be seen in appendix III.4 and III.5

Crossover The SBX Operator is set with a distribution index, η_c, of 5, where small values generate a flatter distribution, resulting in offspring far away from the parent solutions. Equation 5.9 and 5.10 determine the k^{th} design variable of each child c from parents p. The spread factor σ_c comes from a distribution in equation 5.11, that is formed by η_c and sampled from a random number *rand* between zero and one. The function and resulting distribution density can be seen in figure 5.8a and 5.8b on page 46.

$$c_{1,k} = 0.5\left[(1+\sigma_c)\, p_{1,k} + (1-\sigma_c)\, p_{2,k}\right] \tag{5.9}$$

$$c_{2,k} = 0.5\left[(1-\sigma_c)\, p_{1,k} + (1+\sigma_c)\, p_{2,k}\right] \tag{5.10}$$

$$\sigma_c = \begin{cases} (2\cdot rand)^{\frac{1}{\eta_c+1}} & \text{if } rand \leq 0.5 \\ \left(\frac{1}{2\cdot(1-rand)}\right)^{\frac{1}{\eta_c+1}} & \text{otherwise} \end{cases} \tag{5.11}$$

Mutation Mutation is done through polynomial mutation, meaning every design variable is changed by a random amount, with the strength determined from a distribution with the mutation distribution index η_m of 5. Only one parent is used, determining each design variable k using equation 5.12, with the spread

factor $\sigma_{m,k}$, and the upper and lower variable limits $lim_{u,k}$ and $lim_{l,k}$. The variation is determined from the distribution equation 5.13, which is formed by η_m and sampled from a random number $rand_k$ between zero and one. The function and resulting distribution density can also be seen in figures 5.8a and 5.8b on page 46.

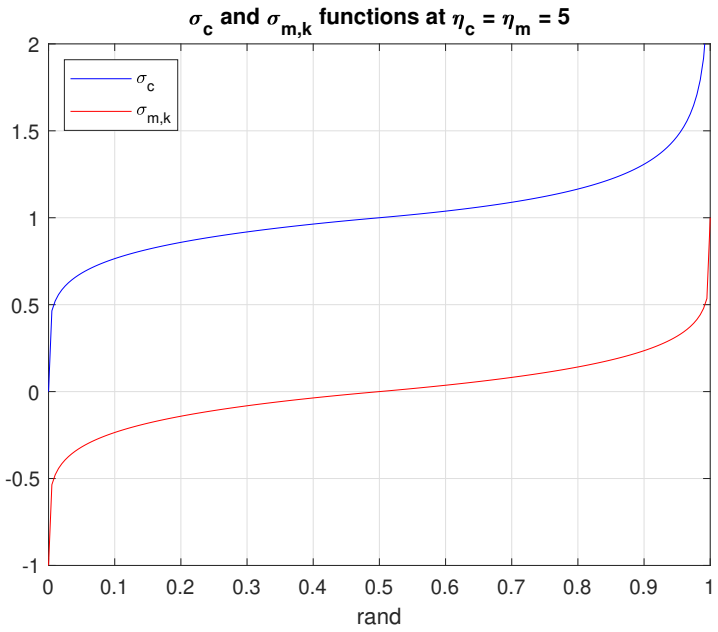
$$c_k = p_k + (lim_{u,k} - lim_{l,k}) \cdot \sigma_{m,k} \tag{5.12}$$

$$\sigma_{m,k} = \begin{cases} (2 \cdot rand_k)^{\frac{1}{\eta m+1}} - 1 & \text{if } rand_k < 0.5 \\ 1 - [2(1 - rand_k)]^{\frac{1}{\eta m+1}} & \text{otherwise} \end{cases} \tag{5.13}$$

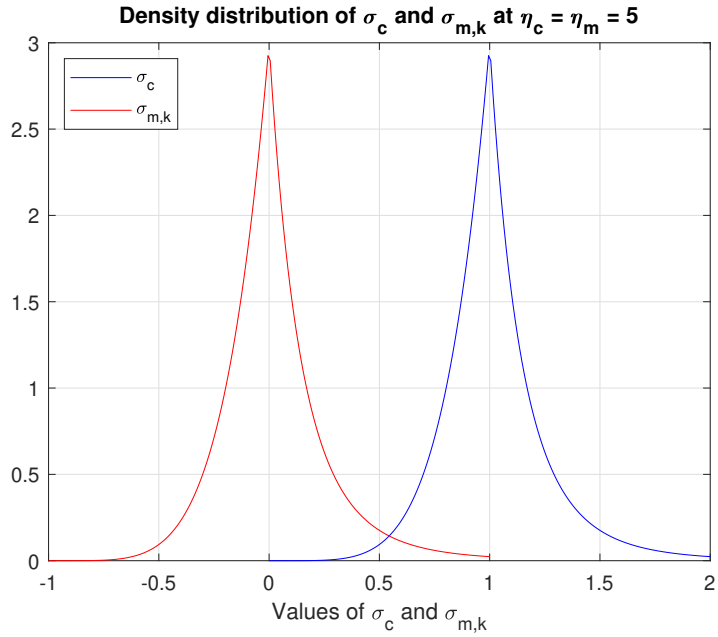
5.3.4 Combination and culling

To make sure that elitism is implemented in the algorithm, the offspring and the original population are added to the same intermediate set. This prevents the best designs from being discarded. From here the intermediate set is sorted using the same nondominated sorting method as earlier.
To keep the population at a stable number, the intermediate population is reduced to the size of the original population. Each rank in ascending order is added to the next population. When the size of the next rank is larger than the remaining places, only the ones with the best crowding distance are added. A visual representation of the elitism can be seen in figure 5.9.

After this procedure, the stopping criteria is checked, and the algorithm either returns to generate the next generation, or it stops, saves the current generation and plots the resulting Pareto-solutions. The current stopping criterion is that the designated number of generations is reached.



(a) Representation of equation 5.11, and 5.13



(b) Density distribution of σ_c and $\sigma_{m,k}$ in figure 5.8a

Figure 5.8: Distribution and functions used to determine the extend of crossover and mutation

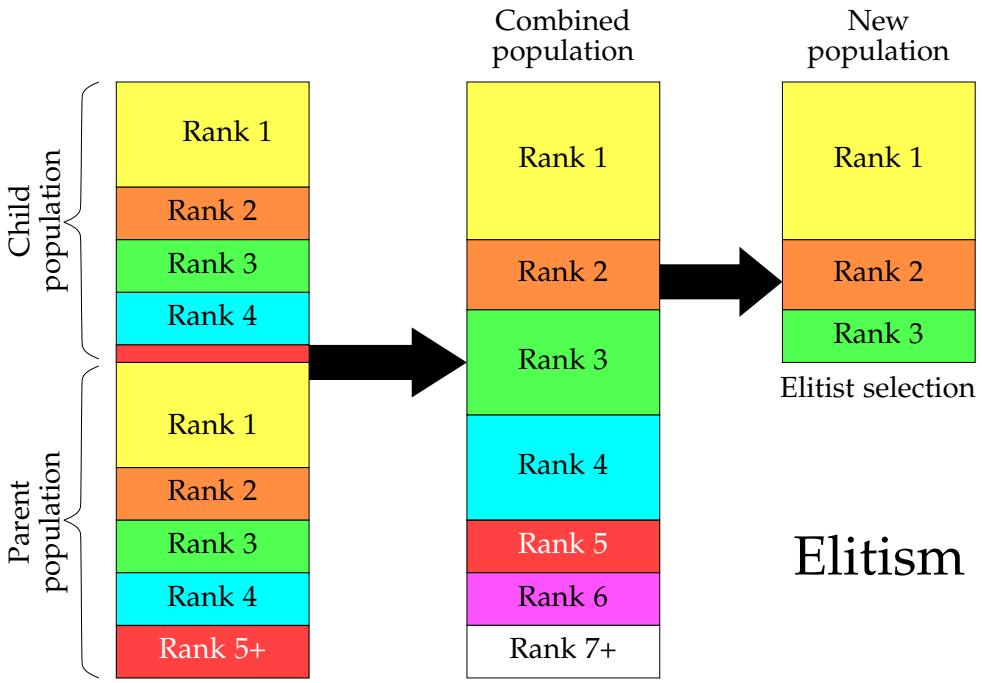


Figure 5.9: Implementation of elitism in NSGA-II. Combination of populations, Nondominated sorting, ended by only keeping the best population-sized part.

Chapter 6

Validation of methods

To make sure the chosen methods are applicable, they will be tested for simple cases. The CFD model will be tested for validity on a staggered tube bank without implemented Vortex Generators (VG's). The multi-objective optimisation algorithm, as well as the surrogate model, will be tested on two different test functions. These functions are chosen to best reflect aspects of the given problem; continuous behaviour and constraints. As the RSM gives a quadratic function to be solved, the problem is assumed to be convex. The utilized generations and populations for the solutions will be given for each test function.

6.1 CFD model

To validate the OpenFOAM model, a version without VG's is constructed. The model is then run with the same settings, and the result can be compared to reference work. The geometry, in this case, is equivalent to flow in a staggered tube bank with continuous fins, with data for validation taken from VDI (2010). The modelled region is shown in figure 6.1.

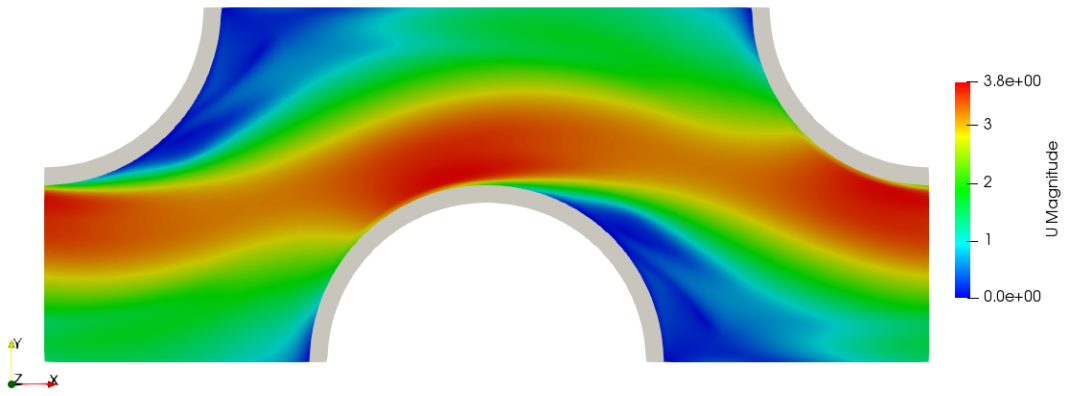


Figure 6.1: Velocity profile of the modelled tube bank without VG implementation.

The model indicates a bulk Nusselt number of 64.9185 for the entire region, with

a friction factor of 0.6415 and at a Reynolds number of 10,754.7. For the given geometry size, the book indicates a Nusselt number of 62.6873. This is valid for in a range of $10^3 \leq \text{Re} \leq 10^5$ with an accuracy of $\pm 10\% - \pm 25\%$. The numbers result in deviance of 3.56% between the model and the reference work. A deviation of this magnitude is deemed reasonable, as it falls within the accuracy range.

6.2 Constraint Handling

To test how constraints are handled in NSGA II for the direct solution, as well as on a quadratic approximation, a test function is taken from Deb et al. (2002). The function is called CONSTR and features a multi-objective optimisation problem, which is also influenced by constraints and is shown in table 6.1.

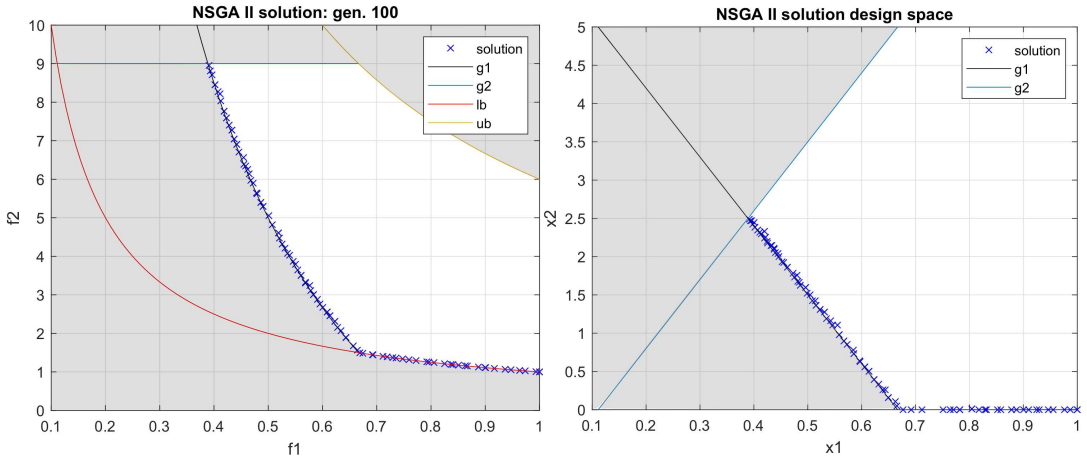
Table 6.1: Test problem for multi-objective optimisation with constraints.

Range	Objective function	Constraints
$x_1 \in [0.1, 1.0]$	$f_1(\mathbf{x}) = x_1$	$g_1(\mathbf{x}) = x_2 + 9x_1 \geq 6$
$x_2 \in [0, 5]$	$f_2(\mathbf{x}) = (1 + x_2)/x_1$	$g_2(\mathbf{x}) = -x_2 + 9x_1 \geq 1$

The solution is first done directly in the NSGA-II algorithm. Afterward, a response surface is made from a full factorial design, where the new function values are also translated into the constraints. This means that the result from the quadratic approximation will search for another optimum under the same constraints. Both solutions in both objective- and design space can be seen in figure 6.2 after 100 generations with a population of 50. The objective space is limited by the search domain; the upper boundary ub and the lower boundary lb. This is also the case for the design space, but in this case, this is determined by the plot limits.
For the direct solution on figures 6.2a and 6.2b, the Pareto frontier is found on the constraint $g1$ and lb, with endpoints at the constraint $g2$ and ub.
The quadratic response surface solution is also within the feasible domain, but as seen on figures 6.2c and 6.2d, the Pareto frontier is different. As lb has a different shape in the objective space, solutions along this boundary are not included, giving only solutions on $g1$. The solution also brings new bounds for the objective space, hence the different y-axis in figure 6.2c.
The set of solutions in figure 6.2d are included in the set of solutions from figure 6.2b. This means that in this case, the response surface method can find similar Pareto optimal solutions to a direct case, albeit not necessarily with the same spread.

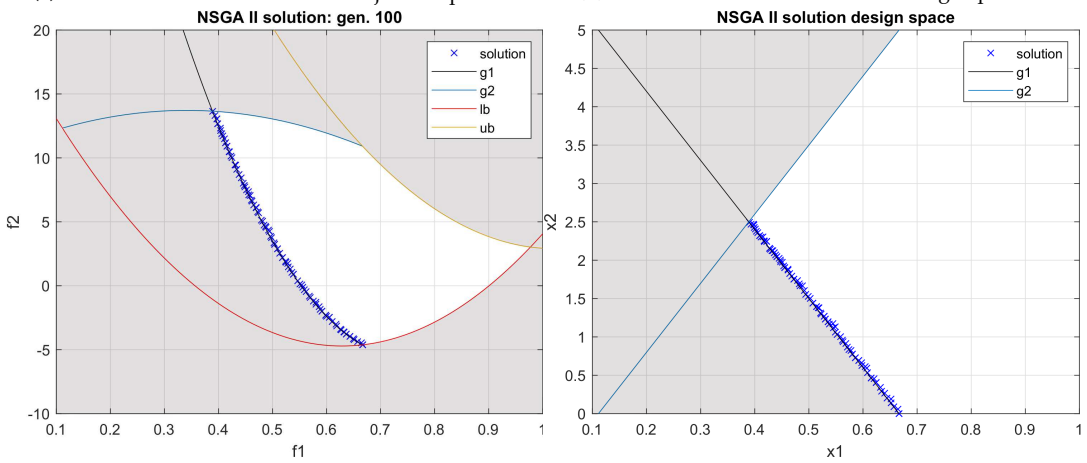
6.3 Zitzler-Deb-Thiele functions

To test Response Surface Modelling (RSM) on other functions, two test functions from Zitzler et al. (2000) are applied, ZDT1, and ZDT2. These are generated to make it difficult for genetic algorithms to converge. The general function description is shown in table 6.2, where the difference in the two functions depends on



(a) Solution for direct function in objective space

(b) Solution for direct function in design space



(c) Solution for response surface function in objective space

(d) Solution for response surface function in design space

Figure 6.2: Solutions to the CONSTR function for a quadratic RSM and direct solver. Infeasible regions are indicated by gray.

the description of f_2. In the equations, m is the number of variables, of which there should be 30. The analytical solutions are at $g(x) = 1$.

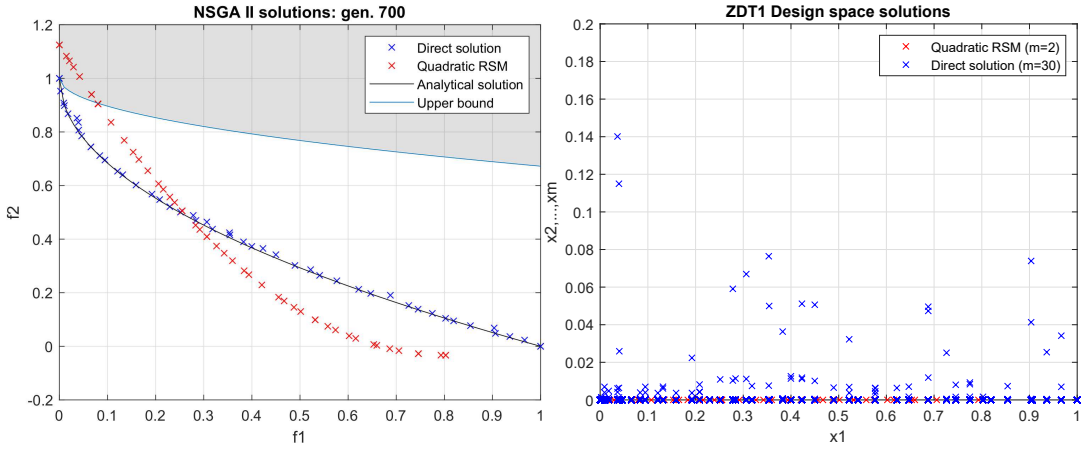
Table 6.2: Zitzler-Deb-Thiele test problems 1 and 2 for unconstrained multi-objective optimisation.

Range	Objective function	Subfunctions
$x_i \in [0,1]$	$f_1(x_1) = x_1$	$g(x_2,\ldots,x_m) = 1 + 9 \cdot \sum_{i=2}^{m} x_i/(m-1)$
(ZDT1)	$f_2(\mathbf{x}) = g \cdot h_2$	$h_2(f_1,g) = 1 - \sqrt{f_1/g}$
(ZDT2)	$f_2(\mathbf{x}) = g \cdot h_2$	$h_2(f_1,g) = 1 - (f_1/g)^2$

As a response surface of $m = 30$ variables has 931 coefficients, it is instead generated for $m = 2$. As the problem is only constrained by the design variable bounds, constraints cannot be shown. Furthermore, an analytical solution of $g(x) = 1$ means that the optimal solution exists for all x_1 with $x_2,\ldots,x_m = 0$. This means

that the solution is the lower design variable bound. Comparing solutions in the design space will consequently not give much information, as $x_2, \ldots, x_m = 0$ is true for both solutions. Nonetheless, the design space is compared in figures 6.3b and 6.4b. As there are many more variables in the direct solution, views may be skewed.

The results are shown in figures 6.3 and 6.4. As the figures show, NSGA II converges at the Pareto optimal solution in both cases. The RSM, however, cannot capture the functions on a quadratic form, so those solutions will not coincide with the Pareto-frontiers. Furthermore, the quadratic approximations work as in section 6.2 in that they will have their own optimal and boundaries. This is apparent from figure 6.3a and 6.4a.



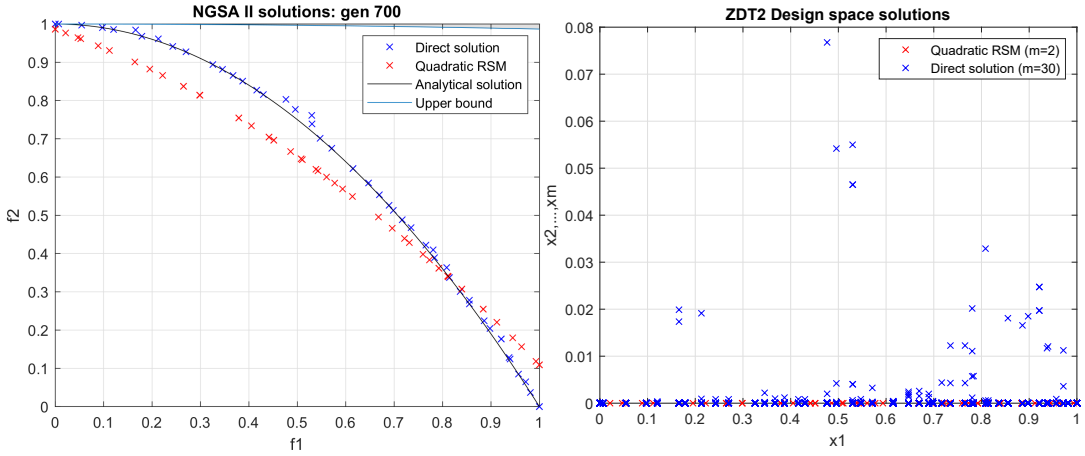
(a) Solution of ZDT1 in objective space

(b) Solution of ZDT1 in the design space

Figure 6.3: Solutions to the Zitzler-Deb-Thiele functions 1 for a quadratic RSM and direct solver. Infeasible regions are indicated by gray.

The comparisons between the solutions in the design space show that the direct solutions are further from the optimal of 0 after the given 700 generations. This may be explained by the direct solution having to optimise 30 variables compared to the response surfaces two. For ZDT in figure 6.3b, the mean of this specific set of solutions $x_2, \ldots, x_m$ is 934.12×10^{-6}, with median value 0. The solution to ZDT2 in figure 6.4b has a mean of this specific set of solutions $x_2, \ldots, x_m$ of 470.75×10^{-6}, also with median value 0. So while some variables are non-zero, the majority is zero at this point.

From these test functions, it can be seen that RSM does not give an exact model of the given function. Solving for a solution can give model solutions similar to the direct solution method, but does not guarantee that the entire Pareto frontier is found. As a direct optimisation is not feasible, with CFD as the objective function, the figures show that RSM can give some solutions close to the Pareto-frontier. CFD can then be utilized to check the validity of the RSM results.



(a) Solution of ZDT2 in objective space

(b) Solution of ZDT2 in the design space

Figure 6.4: Solution to the Zitzler-Deb-Thiele function 2 for a quadratic RSM and direct solver. Infeasible regions are indicated by gray.

Chapter 7

Results and discussion

With the validation of the models done, the CFD model is run through the 41 different configurations, as indicated in table A.1, as well as with a reference case of plain fin geometry, which has a mesh with similar characteristics to the VG-cases. The simulation runs on a work station with a processor frequency of 3.5 GHz and 16 cores.
Relevant data from these runs can be seen in appendix A.2, with the given design variables and the resulting performance evaluation ratios.

7.1 CFD model

The modelled fields of interest are the velocity, temperature, and pressure difference field. These can be investigated to see if the models give relevant results and illuminate flow phenomena. As a baseline, the configuration with mean design variables, which is number 21 in table A.1, can be investigated. Four graphical representations are chosen to highlight the solutions; two slice- representations of the pressure and temperature field, a velocity vector field, and the surface temperature of the fin. The selected representations can be seen in figure 7.1 on page 56.

The pressure difference field does not give much information; pressure is high where the velocity is low, and vice versa. The VG's do not seem to generate a significant change to the pressure difference field. From the velocity vectors, there is a slight indication that the VG slows down the flow and directs it a bit more behind the tubes. Slow recirculation zones appear behind the tubes, which is to be expected. In the temperature field, the cooling from the VG's is visible for the first VG, where the temperature field shifts near it. When looking at the temperature of the fins, the temperature is higher far away from the VG's, indicating that the VG's may transport the exchanged heat efficiently into the fin.

Other configurations of interest are the best and worst case scenarios. These can be found by plotting the resulting data and looking for solutions of high Nu and low f, and low Nu and high f. For the given points, this can be seen in figure 7.2. In the figure, some of the better solutions are configuration 5, which seems to improve a

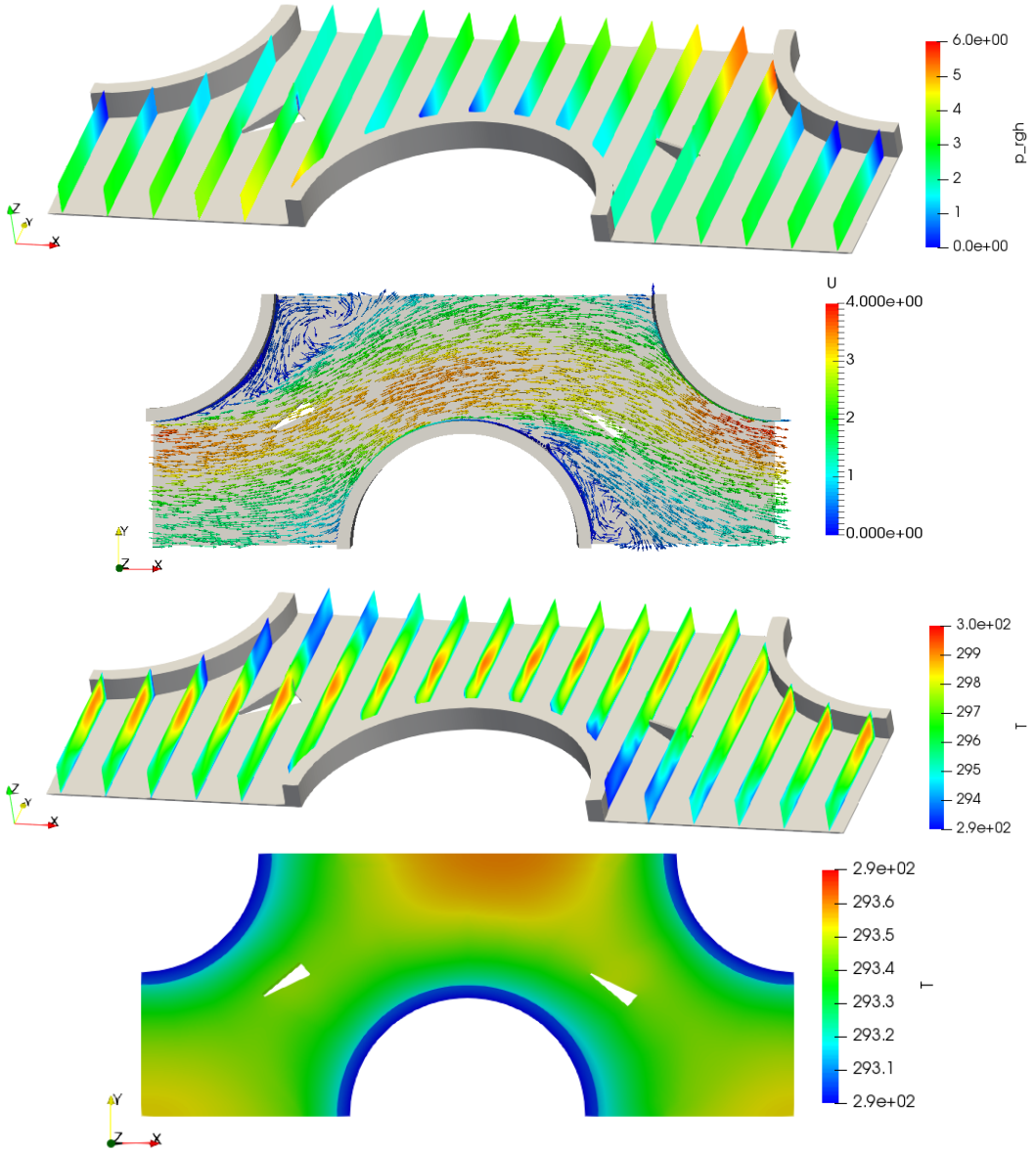


Figure 7.1: Graphical representations of VG configuration 21. Shown in order are; pressure difference field (Unit [m^2/s^2]), velocity vectors (Unit [m/s]), temperature field (Unit [K]), and fin temperature (Unit [K]).

lot in both parameters, 1, which has the highest heat transfer, and 26, which has the lowest loss coefficient. Some of the worst solutions are 17, which has the lowest heat transfer, and 16, which has the highest loss coefficient. The difference in fields can be highlighted to show potential trends to improve heat transfer and reduce pressure loss, respectively.

The temperature fields of configuration 5, 17, and 1 are shown in figure 7.3 on page 58. Solution 5 has the warm part of the stream well distributed across the flow channel. The wakes of the VG's are also visible and shows how they contribute to cooling the fluid. Solution 17 has a higher temperature in the middle of

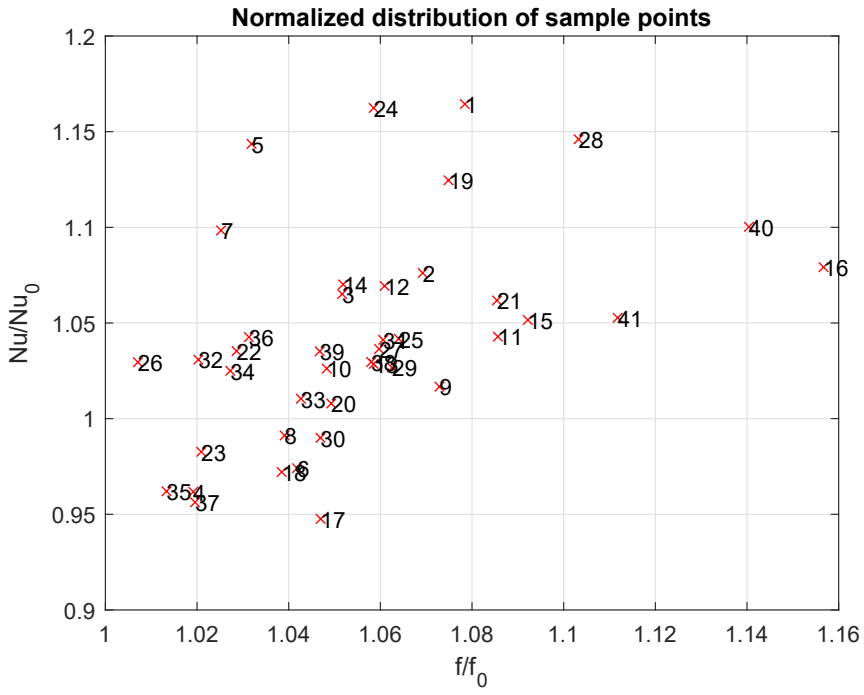


Figure 7.2: Comparison of the tested design configurations, normalized to the reference case.

the flow channel but also has the hot region close to the middle part of both tubes. Solution 1, the one with the highest Nusselt number, also has the hot region well distributed across the flow channel. The VG's here are larger than for configuration 5, so heat may be better exchanged at the cost of extra surface friction. The increase in heat transfer is also apparent in the temperature of the fins, shown in figure 7.4. Here configuration 5 and 17 are compared, and it can be seen how the placement of the VG change how well heat is transferred to the fin. This information could be an indicator of a good VG location.

Comparison of the velocity streamlines of configuration 5, 16, and 26 can be seen in figure 7.5 on page 59. Configuration 5 has the smaller VG's of the compared group. They are located near the end of the high-velocity regions next to the side of the tubes. The angle leads the flow behind the tubes to some degree. Configuration 16 has larger VG's at a steeper angle to the flow direction. The VG's are also placed behind the tubes, with only the tip in the high-velocity region. The model would suggest that this creates more heat transfer, but also comparatively more friction. Configuration 26 has the VG's inside the high-velocity region as well, but at a less steep angle compared to the other two configurations. If the flow is affected here, it is mostly directed toward the side of the tube. This could mean that the VG's, in this case, work more as a surface for heat transport than a way to direct the flow.

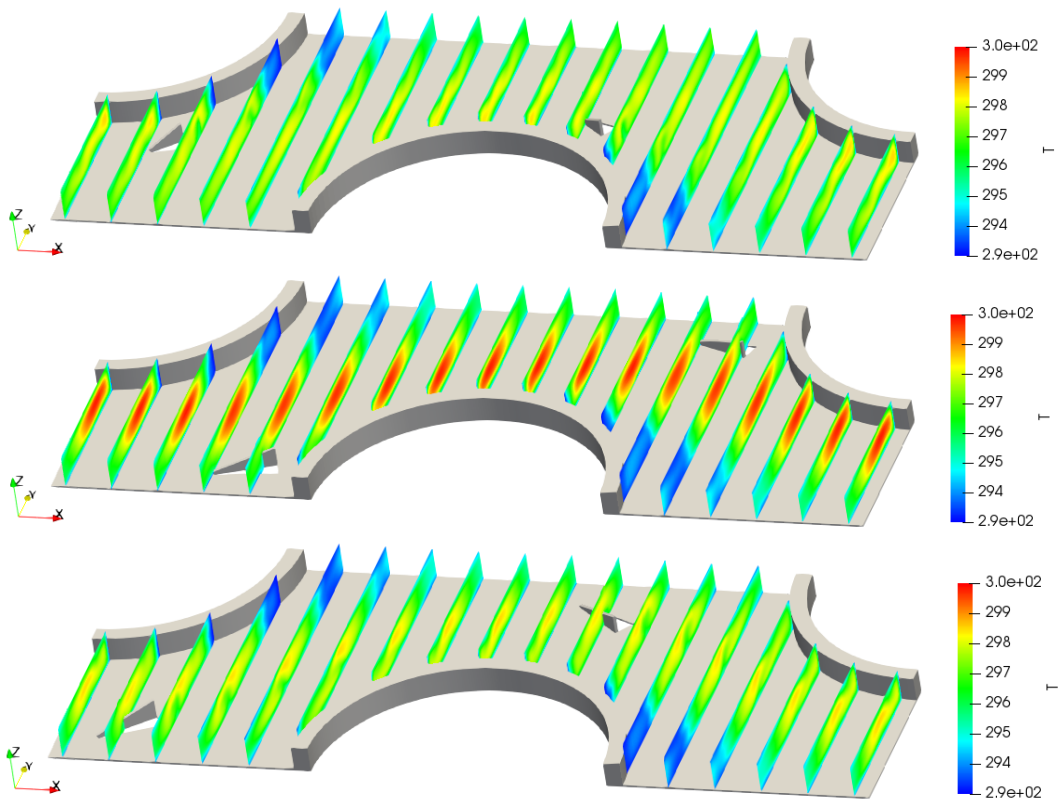


Figure 7.3: Comparison of the temperature field in the middle of the geometry for configuration 5, 17, and 1. Temperature is in [K].

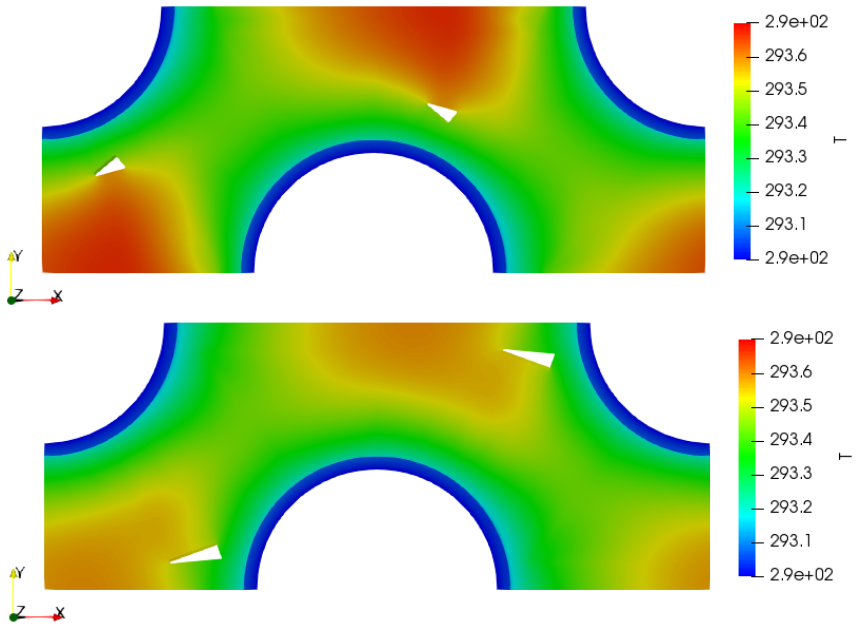


Figure 7.4: Comparison of the fin temperature for configuration 5 and 17. Temperature is in [K].

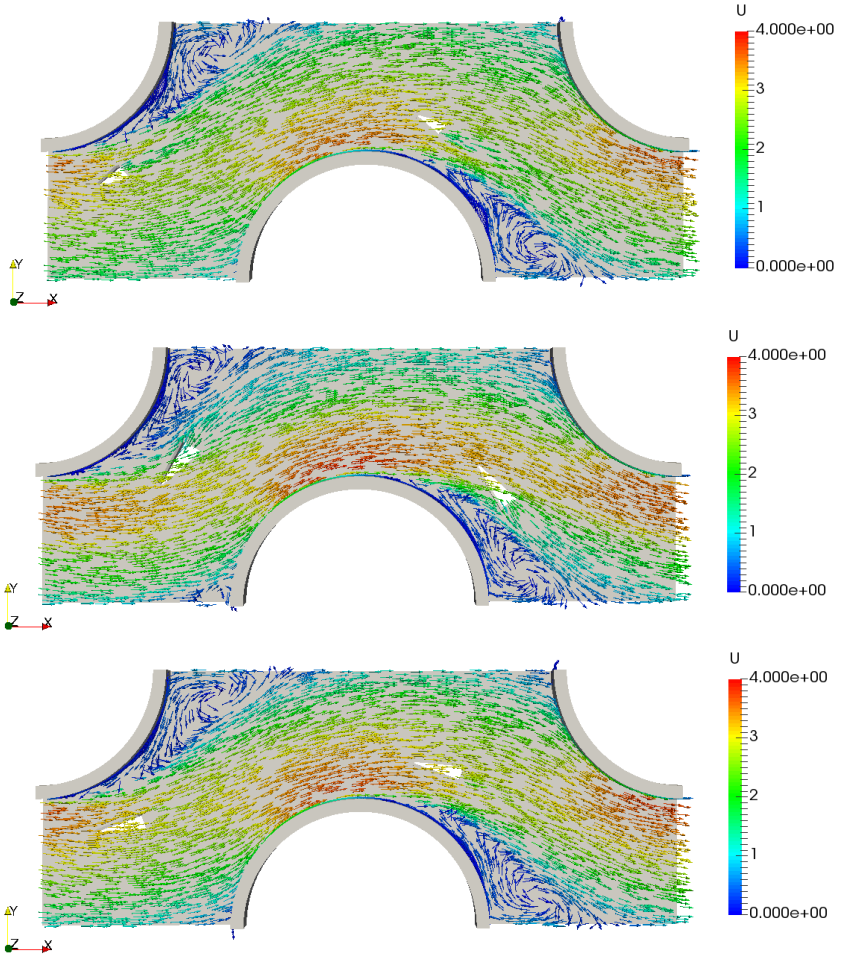


Figure 7.5: Comparison of the velocity field for configuration 5, 16, and 26. Velocity is in [m/s].

7.2 Optimisation

The chosen design of experiments of a modified Box-Behnken design, combined with the upper and lower limits of the design variables, can be seen in appendix A.2, table A.1. To also help identify the modified part of the design, levels are shown in parenthesis for each design variable.
The data is applied in a Response Surface Model (RSM) to give the quadratic functions in equation 7.1 and 7.2.

$$
\begin{aligned}
f/f_0 = {} & 0.8396 + 0.0165 \cdot d_x + 2.6\text{e-}3 \cdot d_y + 0.0218 \cdot l + 0.2132 \cdot h - 1.5\text{e-}3 \cdot \alpha \\
& - 1.5\text{e-}3 \cdot d_x^2 - 1.7\text{e-}3 \cdot d_y^2 - 2.9\text{e-}3 \cdot l^2 - 0.1428 \cdot h^2 + 2.4\text{e-}5 \cdot \alpha^2 \\
& + 2.3\text{e-}3 \cdot d_x \cdot d_y + 1.4\text{e-}4 \cdot d_x \cdot l + 1.7\text{e-}3 \cdot d_x \cdot h - 2.1\text{e-}4 \cdot d_x \cdot \alpha \\
& + 1.1\text{e-}4 \cdot d_y \cdot l + 8.5\text{e-}5 \cdot d_y \cdot h + 1.5\text{e-}5 \cdot d_y \cdot \alpha \\
& - 5.1\text{e-}5 \cdot l \cdot h - 1.2\text{e-}4 \cdot l \cdot \alpha + 3.0\text{e-}3 \cdot h \cdot \alpha
\end{aligned}
\tag{7.1}
$$

$$\begin{aligned} Nu/Nu_0 = {} & 0.2500 - 3.3\text{e-}3 \cdot d_x + 0.0315 \cdot d_y + 0.0513 \cdot l + 0.7150 \cdot h + 9.9\text{e-}3 \cdot \alpha \\ & - 3.7\text{e-}4 \cdot d_x^2 - 1.7\text{e-}3 \cdot d_y^2 - 5.5\text{e-}35 \cdot l^2 - 0.2522 \cdot h^2 - 2.6\text{e-}5 \cdot \alpha^2 \\ & + 1.6\text{e-}3 \cdot d_x \cdot d_y + 1.6\text{e-}3 \cdot d_x \cdot l - 8.0\text{e-}3 \cdot d_x \cdot h - 1.8\text{e-}4 \cdot d_x \cdot \alpha \\ & - 1.7\text{e-}3 \cdot d_y \cdot l - 2.8\text{e-}3 \cdot d_y \cdot h - 2.9\text{e-}4 \cdot d_y \cdot \alpha \\ & + 7.9\text{e-}3 \cdot l \cdot h - 3.7\text{e-}4 \cdot l \cdot \alpha - 5.8\text{e-}4 \cdot h \cdot \alpha \end{aligned} \tag{7.2}$$

Numbers in power notation are truncated at two digits. A longer format of the numbers can be found in Appendix A.3, table A.2, and A.3.
To check whether all of the parameters are significant, the RSM is put through multiple regression in the open source software R. This will indicate how much of the data set can be explained by the model, and how significant each of the coefficients are. If some parameters are found to be very insignificant, they can be removed to reduce the model's complexity without the loss of significant information. This will be monitored by looking at each new models R-square and adjusted R-square. The regular R-square will decrease as parameters are removed, but as the adjusted R-square is dependent on parameter count, and may increase, in the beginning. The reduction will be applied to the most insignificant variable until the adjusted R-square does not increase anymore. To check the predictive power of the models, they are compared to the sampled data in figure 7.6, with model data in Appendix A.3. The full line indicates values that are precisely estimated, while the dashed lines indicate a 10% error margin. As all values fall within the margin, the models are determined to be applicable. The loss coefficient model has a multiple R-squared of 0.8747, an adjusted R-squared of 0.7494, and a p-value of 2.98e-05. The Nusselt number mode has a multiple R-squared of 0.8315, an adjusted R-square of 0.6631, and a p-value of 3.851e-04.

The models are then reduced through multiple regression to remove insignificant variables. For equation 7.1, seven interaction terms are removed, resulting in a multiple R-square of 0.8710, an adjusted R-square of 0.8089, and a p-value of 8.897e-09. This is shown in equation 7.3.

$$\begin{aligned} f/f_0 = {} & 0.8280 + 0.0188 \cdot d_x + 3.7\text{e-}3 \cdot d_y + 0.0191 \cdot l + 0.2301 \cdot h - 1.9\text{e-}3 \cdot \alpha \\ & - 1.5\text{e-}3 \cdot d_x^2 - 1.7\text{e-}3 \cdot d_y^2 - 2.9\text{e-}3 \cdot l^2 - 0.1428 \cdot h^2 + 2.4\text{e-}5 \cdot \alpha^2 \\ & + 2.3\text{e-}3 \cdot d_x \cdot d_y - 2.1\text{e-}4 \cdot d_x \cdot \alpha + 3.0\text{e-}3 \cdot h \cdot \alpha \end{aligned} \tag{7.3}$$

For equation 7.2, four interaction terms are removed along with two quadratic terms. This results in a multiple R-square of 0.8182, an adjusted R-square of 0.7203, and a p-value of 2.125e-06. This is shown in equation 7.4.

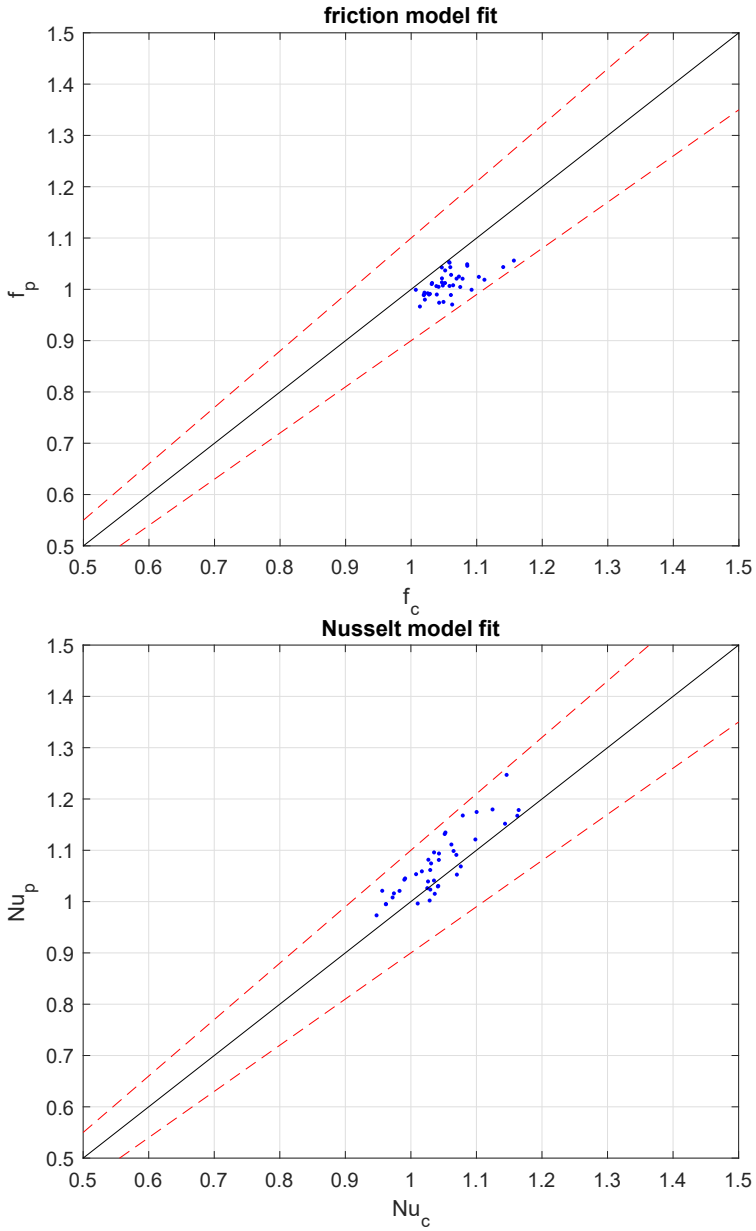


Figure 7.6: Predicted response surface model values compared to calculated data. The dashed red lines indicate a 10% margin of error.

$$
\begin{aligned}
Nu/Nu_0 = {} & 0.4995 - 6.2\text{e-}3 \cdot d_x + 0.0274 \cdot d_y + 0.0271 \cdot l + 0.5364 \cdot h + 5.6\text{e-}3 \cdot \alpha \\
& - 1.3\text{e-}3 \cdot d_y^2 - 3.3\text{e-}3 \cdot l^2 - 0.1743 \cdot h^2 \\
& + 1.1\text{e-}3 \cdot d_x \cdot d_y + 1.6\text{e-}3 \cdot d_x \cdot l - 8.8\text{e-}3 \cdot d_x \cdot h - 1.8\text{e-}4 \cdot d_x \cdot \alpha \\
& - 1.7\text{e-}3 \cdot d_y \cdot l - 2.9\text{e-}4 \cdot d_y \cdot \alpha
\end{aligned}
\tag{7.4}
$$

To again check the predictive power of the models, they are also compared to

the sampled data, seen in figure 7.7. The data for the models can be seen in appendix A.3.

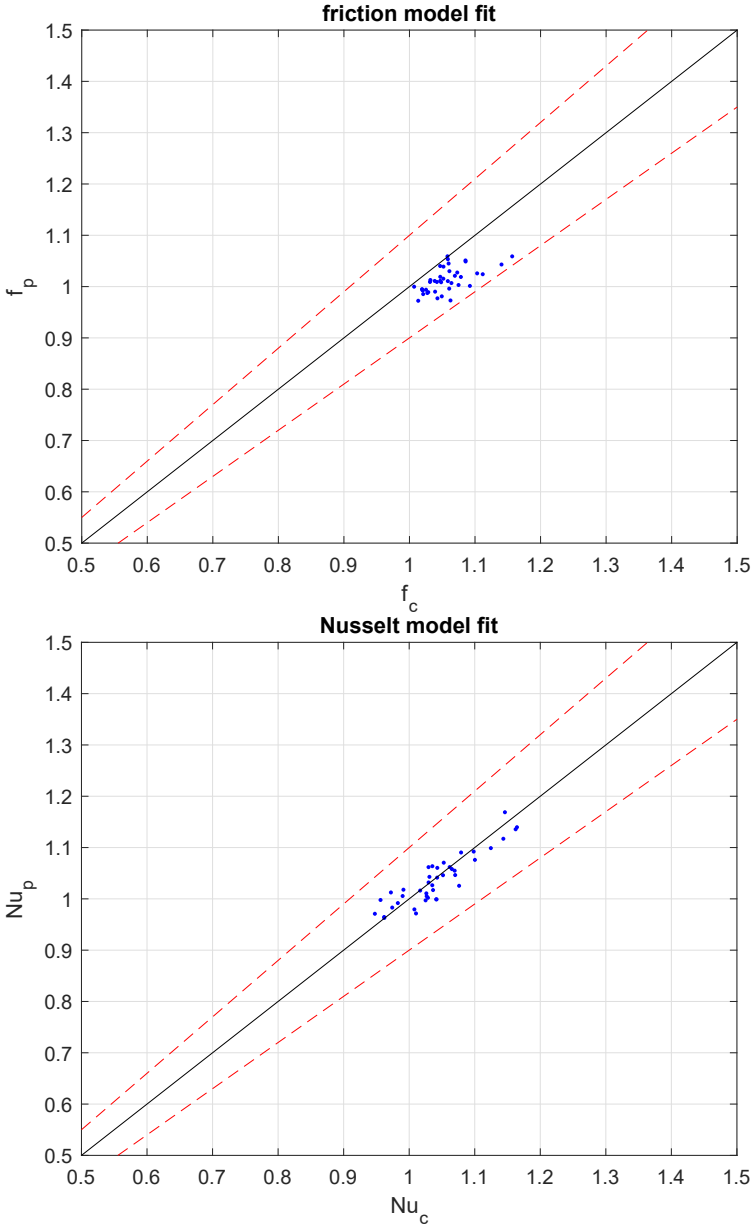


Figure 7.7: Predicted reduced response surface model values compared to calculated data. The dashed red lines indicate a 10% margin of error.

The friction does not seem to be significantly changed, but the reduced model for the Nusselt number seems to be closer to the fitted line. This gives two sets of models to optimise around; the full quadratic model with the higher R-square value, and the reduced models, that better fit the utilised data and have a higher adjusted R-square.

These two sets of two equations are then implemented in the NSGA-II algorithm as

the objective functions. Constraints are also implemented similar to the procedure from 3.1. This is done with a 10% margin from the outer tube walls, resulting in six overall constraints; three corner points for the lanced hole constrained for two tube walls in the design space.

After some initial testing, the algorithm is set to run with a population of 200 for 1000 generations. The resulting Pareto frontiers can be seen in figure 7.8 and 7.9.

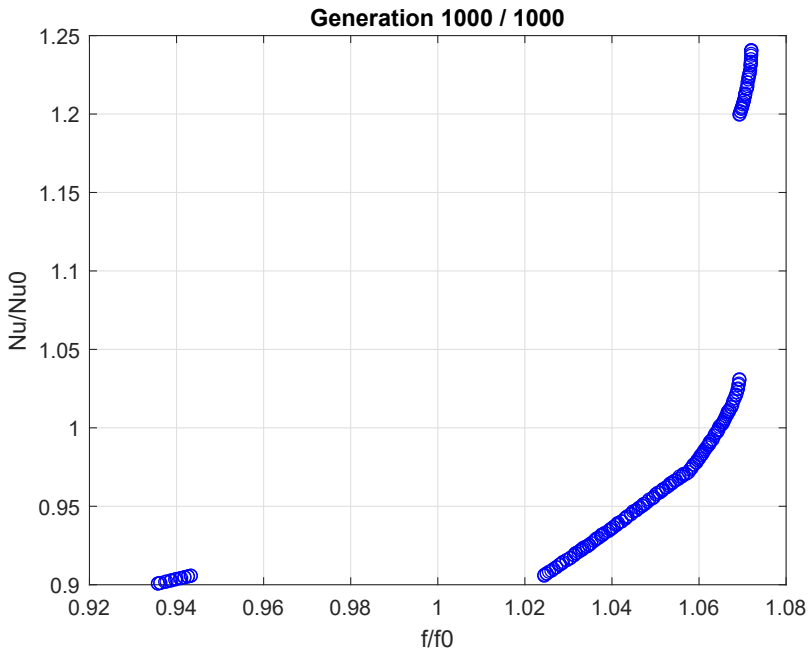


Figure 7.8: Solution set from the NSGA-II optimisation of the full quadratic model.

From figure 7.8 it can be seen that the algorithm for the full model has a small set of unlikely solutions, as the implementation of a VG decreases loss coefficient and heat transfer, something not apparent from any of the modelled points. The best solution appears at the top end of the Pareto-front. This point indicates an improvement in Nu of $\approx$ 25%with an increase in f of $\approx$ 7%. The sectioning of the front is likely caused by different parts of it being subject to different boundaries on the design variables. The given Pareto set can be sectioned into four parts, with the middle block divided at the bend around f/f0 = 1.06. Each part can then be analysed to show which variable bounds are met if any. This is indicated in table 7.1. The best part of the front seems to be the solutions which use the highest allowed h and α. This could indicate that better solutions may exist beyond these bounds.
In figure 7.9, most of the Pareto-front is spread out over the unlikely results of the reduction in heat transfer and loss coefficient. The best solutions only seem to indicate an improvement in Nu of $\approx$ 2% with an increase in f of $\approx$ 7% as well. This Pareto set can be sectioned into six parts, with the bottom and top blocks block divided at their bends around f/f0 = 0.9 and f/f0 = 1.07 respectively. Each part can again be analysed to show which variable bounds are met if any. This is indicated

Table 7.1: Boundaries met for the full RSM Pareto set.

part #	d_x	d_y	l	h	α
1	-	16 (1)	6 (1)	0.8 (-1)	20 (-1)
2	-	-	2.4 (-1)	0.8 (-1)	20 (-1)
3	-	-	-	-	20 (-1)
4	-	-	-	1.4 (1)	60 (1)

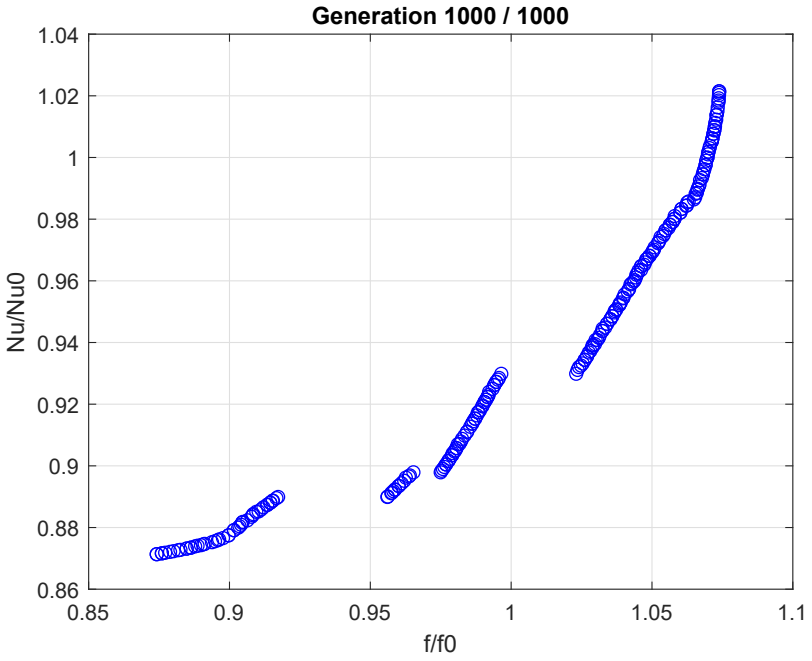


Figure 7.9: Solution set from the NSGA-II optimisation of the reduced model.

in table 7.2. The indication here is that both solution-sets share boundary sets in their parts. Boundaries for part one, two, and three from the full model are the same as part one, five, and six from the reduced model. What the reduced model lack is the high Nu/Nu0 part four from the full model, while it has more solutions in-between that are not so optimal.

Table 7.2: Boundaries met for the reduced RSM Pareto set.

part #	d_x	d_y	l	h	α
1	-	16 (1)	6 (1)	0.8 (-1)	60 (1)
2	-	16 (1)	6 (1)	-	60 (1)
3	16 (1)	16 (1)	-	1.4 (1)	60 (1)
4	16 (1)	-	2.4 (-1)	1.4 (1)	60 (1)
5	-	-	2.4 (-1)	0.8 (-1)	20 (-1)
6	-	-	-	-	20 (-1)

As the optimisation of the full model predicts better improvements, the full quadratic model will be used continuing forward. Select data from the Pareto solution set can be seen in appendix A.4, table A.6.
To evaluate the solution from the optimisation algorithm, the best-predicted configuration is taken from the solution set and used for a CFD model. The configuration and solution can be seen in table 7.3.

Table 7.3: Sample result from the optimisation algorithm, tested in CFD. Parameter value is indicated in parenthesis.

Parameter	Description	Value	Unit
d_x	Transversal position of VG	6.3857 (-0.3735)	mm
d_y	Longitudinal position of VG	5.2091 (-0.5416)	mm
l	Length of VG	2.9018 (-0.7212)	mm
h	Height of VG	1.4 (1)	mm
α	Angle of attack	60 (1)	°
f_{est}	Estimated f/f_0	1.0720	-
f_{mod}	CFD modelled f/f_0	1.0929	-
ϵ_f	Relative Error	1.91	%
Nu_{est}	Estimated Nu/Nu_0	1.2407	-
Nu_{mod}	CFD modelled Nu/Nu_0	1.2414	-
ϵ_{Nu}	Relative Error	0.06	%

The selected representations of the chosen solution can be seen in figure 7.10 on page 66. The pressure difference field is again not that indicative of the influence of the VG. It is, however, possible to see a slight increase in pressure on the front side of the second VG. On the velocity vectors, it can be seen how the VG's slow down the flow, and change the direction of it visibly. This might also cause a lower velocity near the tube walls, preventing pressure loss due to friction. The temperature field shows how the wake of the VG's helps distribute the temperature across the flow region. This can be seen as the central temperature switches from yellow to green. Looking at the temperature field in the fins, it appears that the VG's are placed near the end of the hot region, making the upstream side colder than the downstream one. This could also be an indication of a good VG location.

7.3 Further discussion

The solution given in figure 7.10, along with the data presented in table 7.3 indicates that the optimal solution, that the algorithm found on the RSM, is very close to the values found in the CFD model. This is a good indicator that this approach will be able to yield better solutions to similar optimisation problems.

The validity of the CFD model was also shown to be good, as the modelled reference results were within the accuracy of applied literature. To better predict the

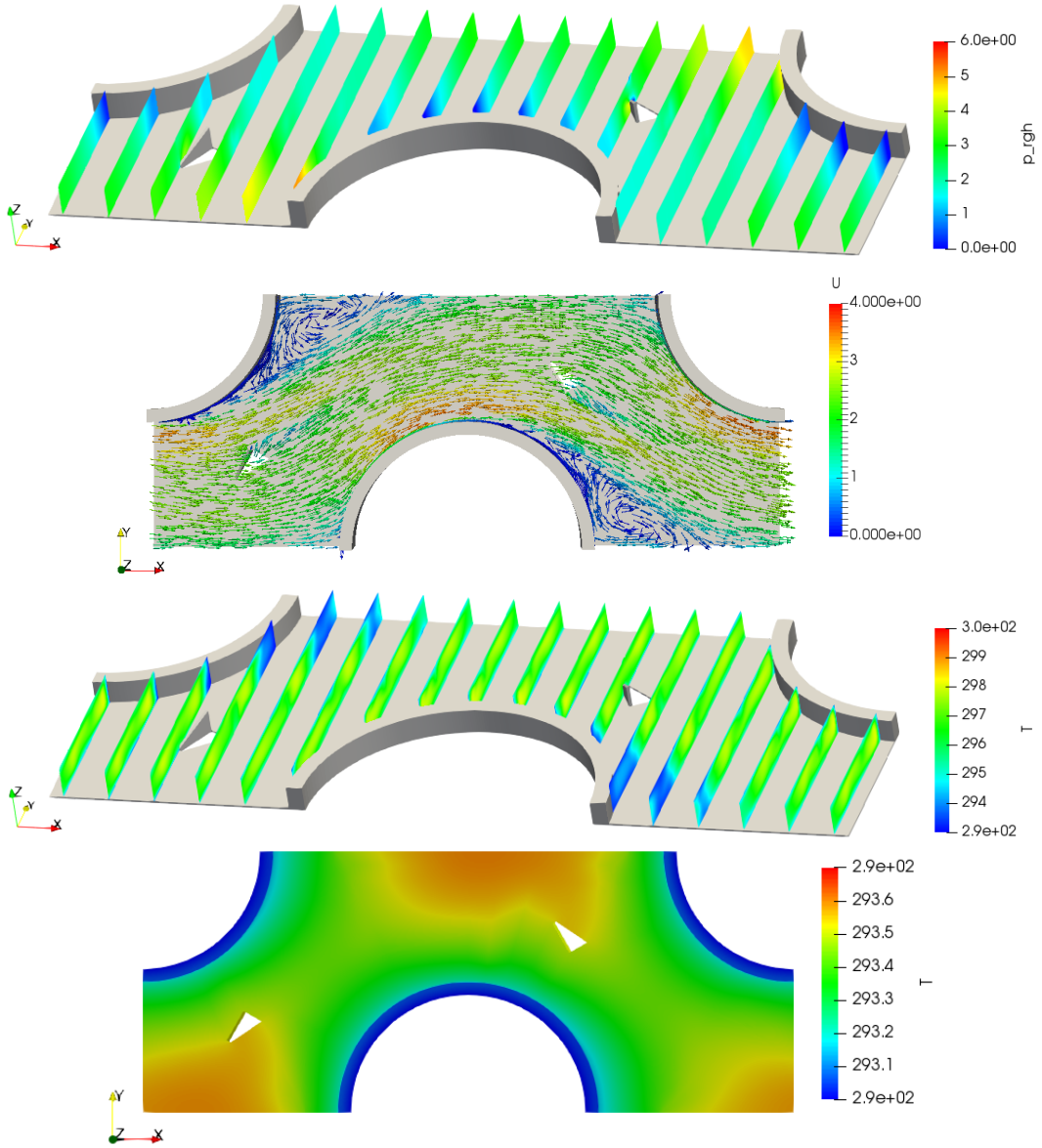


Figure 7.10: Graphical representations of VG configuration in the chosen Pareto solution. Shown in order are; pressure difference field (Unit [m^2/s^2]), velocity vectors (Unit [m/s]), temperature field (Unit [K]), and fin temperature (Unit [K]).

results, a Large Eddy Simulation could have been used to resolve potential smaller scale eddies. This would also mean another closure model, but LES would most likely only be necessary if very precise solutions would have to be modelled, and could also include experimental validation at that point. This is also only a possibility if more computational resources are available, as this thesis found RANS to be a suitable compromise between accuracy and computational power.
Furthermore, for meshing purposes 10% of the VG tip was removed from the geometry. Even if the impact has been insignificant, it does mean that the mesh does not completely resemble the geometry. A different meshing strategy could allow

for the complete geometry to be modelled.

The CFD model was run using FOAMExtend 4, as the implementation of conjugate heat transfer under the desired conditions was not available in the other versions of OpenFOAM. This meant that some of the normally available schemes and solvers were not compatible with the model. If newer versions of OpenFOAM include conjugate heat transfer in their software, other settings could be an area of investigation, so that this endeavour might be more easily accomplished in the future.

The model was done under periodic assumptions, but to also reduce the amount of variables, the two VG's were assumed to be identical relative to their respective tube wall. If results from Salviano et al. (2016) are to be any indicator, the VG's may have optimal settings that are asymmetrical. This could again be done by using more computational power. In this case, the reduction of the response surface may be more advantageous, as the number of coefficients could increase above 100.

Roll angle is also a parameter that could have expanded the model. This would also be in line with the modelling of the holes, as the roll angle may induce more flow in the z-direction.

As the final modelled Pareto solution has its angle of attack and VG height at the upper variable bound, further investigations could try to move the search area more in this direction. This may uncover even better optimisation points, that were outside of the scope of this thesis.

The GCI for the chosen mesh refinement was only low for the Nusselt number, while the value for the loss coefficient was further from the Richardson extrapolated one. Either a new meshing strategy or a higher rate of refinement could be applied to make sure the mesh has the correct refinement. This would also require higher computational power, as the cell count could increase significantly.

For the choice of optimiser, DACE was rejected due to its lack of availability and applicability in multi-objective optimisation. Software is available for applying DACE to data, but the possibility of implementation in genetic algorithms was not apparent. This could have proven to be a better alternative, but as the RSM already were at a high R-square and through the optimiser gave a result very close to the modelled values, this might not be worth investigating in this case.

The validation of the optimisation methods shows that the NSGA-II adequately works on the test problems. The RSM was shown not completely to represent the functions in the objective space, but gives optimal solutions in the design space. This trend was also more apparent in the case of a full quadratic model, compared to a reduced one.

7.4 Further studies

As the validation through GCI indicated, the mesh was only within the error bands for the loss coefficient. To get the Nusselt number within these, a finer mesh was required, which was not feasible to model with the given time frame. This could be improved by either refining the mesh several times and invest more computational power and time, or by investigating if other factors could prevent the solution from converging at a reasonable rate. The investigation could look at aspect ratio or skewness of the cells. The meshing strategy could also be reworked to include triangular polyhedrals, allowing the VG to be modelled with the front tip as well. When applying the DoE for the Box-Behnken design, some of those designs were infeasible due to their placement inside the tubes. If a method of mathematical surface transformation could be applied to properly place all the VG's within the feasible area while still keeping the correct spread, this could improve eventual surrogate models in future investigations.

Chapter 8

Conclusion

This thesis sought to apply parametric optimisation and CFD modelling on the implementation of VG's in a fin and tube heat exchanger. This led to the problem statement to be answered:

Which configurations of VG's in staggered fin-tube heat exchangers give an optimum between maximal heat transfer and minimal pressure drop?

A CFD model using conjugate heat transfer was implemented for a representative geometry under the assumption of periodic boundary conditions. The model was validated with reference work to be within $\approx$ 4% of the predicted value of the Nusselt number, with the reference work having an accuracy of $\pm$10%.
The NSGA-II optimisation algorithm was tested for constraint handling and optimisation of the Zitzler-Deb-Thiele optimisation test problems 1 and 2. Both the constraint handling and test functions gave valid solutions within the design space.
For the given geometry, 41 different configurations were modelled, and from the given performance evaluation parameters, response surfaces were calculated. The models showed a multiple R-square of 0.8747 and 0.8315, with p-values of 2.98e-05 and 3.851e-04 respectively. The implementation in NSGA-II yielded a set of Pareto optimal solutions, with a portion within optimising range.
The optimal solution from the Pareto set was implemented in the CFD model to validate the optimisation result. Fields of interest can be seen in figure 7.10, with configuration data in table 7.3. The CFD model showed that the optimisation had a deviation of 1.91% on the loss coefficient and 0.06% of the Nusselt number, for a loss coefficient increase of 1.0720 and a Nusselt number increase of 1.2407.
As a heat transfer increase of $\approx$ 24% at a pressure drop increase of $\approx$ 7% is seen as a valuable increase in performance, this thesis can conclude that optimal configurations which compromise between high heat transfer and low pressure drop can be determined.

Bibliography

Lei, Yong-Gang et al. (2010). „Hydrodynamics and heat transfer characteristics of a novel heat exchanger with delta-winglet vortex generators“. In: *Chemical Engineering Science* 65, pp. 1551 – 1562. DOI: 10.1016/j.ces.2009.10.017.

Bacellar, Daniel et al. (June 2016). „Novel Airside Heat Transfer Surface Designs Using an Integrated Multi-Scale Analysis with Topology and Shape Optimization“. In: *International Refrigeration and Air Conditioning Conference*. Purdue University.

Fiebig, M. (1998). „Vortices, Generators and Heat Transfer“. In: *Chemical Engineering Research and Design* 76.2. 5th UK National Heat Transfer Conference, pp. 108 –123.

He, Y.L. et al. (2012). „Numerical study of heat-transfer enhancement by punched winglet-type vortex generator arrays in fin-and-tube heat exchangers“. In: *International Journal of Heat and Mass Transfer* 55, pp. 5449 –5458. DOI: 10.1016/j.ijheatmasstransfer.2012.04.059.

Saha, Pankaj, Gautam Biswas, and Subrata Sarkar (2014). „Comparison of winglet-type vortex generators periodically deployed in a plate-fin heat exchanger – A synergy based analysis“. In: *International Journal of Heat and Mass Transfer* 74, pp. 292 –305. DOI: 10.1016/j.ijheatmasstransfer.2014.03.015.

Khanjian, Assadour et al. (2017). „Effect of rectangular winglet pair roll angle on the heat transfer enhancement in laminar channel flow“. In: *International Journal of Thermal Sciences* 114, pp. 1 –14. DOI: 10.1016/j.ijthermalsci.2016.12.010.

Välikangas, Turo et al. (2018). „Fin-and-tube heat exchanger enhancement with a combined herringbone and vortex generator design“. In: *International Journal of Heat and Mass Transfer* 118, pp. 602 –616. DOI: 10.1016/j.ijheatmasstransfer.2017.11.006.

Qian, Zuoqin, Qiang Wang, and Junlin Cheng (2018). „Analysis of heat and resistance performance of plate fin-and-tube heat exchanger with rectangle-winglet vortex generator“. In: *International Journal of Heat and Mass Transfer* 124, pp. 1198 –1211. DOI: 10.1016/j.ijheatmasstransfer.2018.04.037.

Arora, Amit, P.M.V. Subbarao, and R.S. Agarwal (2016). „Development of parametric space for the vortex generator location for improving thermal compactness of an existing inline fin and tube heat exchanger“. In: *Applied Thermal Engineering* 98, pp. 727 –742. DOI: 10.1016/j.applthermaleng.2015.12.117.

Zeng, M. et al. (2010). „Optimization of heat exchangers with vortex-generator fin by Taguchi method“. In: *Applied Thermal Engineering* 30, pp. 1775 –1783. DOI: 10.1016/j.applthermaleng.2010.04.009.

Salviano, Leandro O., Daniel J. Dezan, and Jurandir I. Yanagihara (2015). „Optimization of winglet-type vortex generator positions and angles in plate-fin compact heat exchanger: Response Surface Methodology and Direct Optimization“. In: *International Journal of Heat and Mass Transfer* 82, pp. 373 –387. DOI: 10.1016/j.ijheatmasstransfer.2014.10.072.

Salviano, Leandro O., Daniel J. Dezan, and Jurandir I. Yanagihara (2016). „Thermal-hydraulic performance optimization of inline and staggered fin-tube compact heat exchangers applying longitudinal vortex generators“. In: *Applied Thermal Engineering* 95, pp. 311 –329. DOI: 10.1016/j.applthermaleng.2015.11.069.

Tang, Song-Zhen et al. (2019). „Parametric optimization of H-type finned tube with longitudinal vortex generators by response surface model and genetic algorithm“. In: *Applied Energy* 239, pp. 908 –918. DOI: 10.1016/j.apenergy.2019.01.122.

Lemouedda, A. et al. (2010). „Optimization of the angle of attack of delta-winglet vortex generators in a plate-fin-and-tube heat exchanger“. In: *International Journal of Heat and Mass Transfer* 53, pp. 5386 –5399. DOI: 10.1016/j.ijheatmasstransfer.2010.07.017.

Esmaeilzadeh, A, Nima Amanifard, and H.M. Deylami (July 2017). „Comparison of simple and curved trapezoidal longitudinal vortex generators for optimum flow characteristics and heat transfer augmentation in a heat exchanger“. In: *Applied Thermal Engineering* 125.

Moulinec, C., J.C.R. Hunt, and F.T.M. Nieuwstadt (Sept. 2004). „Disappearing Wakes and Dispersion in Numerically Simulated Flows Through Tube Bundles“. In: *Flow, Turbulence and Combustion* 73, pp. 95–116. DOI: 10.1023/B:APPL.0000049297.28738.71.

Balabani, S. and M. Ylannekis (1997). „Vortex shedding and turbulence scales in staggered tube bundle flows“. In: *The Canadian Journal of Chemical Engineering* 75, pp. 823–831. DOI: 10.1002/cjce.5450750502.

Bhuiyan, Arafat A., M Ruhul Amin, and A.K.M. Sadrul Islam (2012). „Numerical study of 3d thermal and hydraulic Characteristics of wavy fin and tube heat exchanger“. In: *Frontiers in Heat and Mass Transfer* 3. DOI: 10.5098/hmt.v3.3.3006.

Välikangas, Turo (2016). *Conjugate heat transfer in OpenFOAM*. Student report. Chalmers University of Technology. DOI: 10.17196/OS_CFD#YEAR_2016.

Patankar, S.V., C.H. Liu, and E.M. Sparrow (1978). „The periodic thermally developed regime in ducts with streamwise periodic wall temperature or heat flux“. In: *International Journal of Heat and Mass Transfer* 21, pp. 557 –566. DOI: https://doi.org/10.1016/0017-9310(78)90052-2.

Jasak, Hrvoje (1996). „Error Analysis and Estimation for the Finite Volume Method With Applications to Fluid Flows“. PhD. Imperial College of Science, Technology and Medicine.

Roache, Patrick J (1997). „Quantification of uncertainty in computational fluid dynamics“. In: *Annual review of fluid Mechanics* 29.1, pp. 123–160.

Roache, P. J. (1994). „Perspective: A Method for Uniform Reporting of Grid Refinement Studies“. In: *Journal of Fluids Engineering* 116, pp. 405–413. DOI: 10.1115/1.2910291.

Arora, Jasbir Singh (2017). *Introduction to optimum design*. 4th ed. Elsevier.

Eckhardt, Roger (1987). „Stan Ulam, John von Neumann, and the Monte Carlo method“. In: 15.

Jones, Donald R. (2001). „A Taxonomy of Global Optimization Methods Based on Response Surfaces“. In: *Journal of Global Optimization* 21, pp. 345–383. DOI: 10.1023/A:1012771025575.

Simpson, T.W. et al. (2001). „Metamodels for Computer-based Engineering Design: Survey and recommendations“. In: *Engineering with Computers* 17, pp. 129–150. DOI: 10.1007/PL00007198.

Simpson, Timothy W. et al. (1998). „Comparison of Response Surface and Kriging Models for Multidisciplinary Design Optimization“. In:

Giunta, Anthony A. (May 1997). „Aircraft Multidisciplinary design optimization using design of experiments theory and response surface modeling methods“. Ph.D. Virginia Polytechnic Institute & State University.

Cavazzuti, Marco (2013). *Optimization Methods: From Theory to Design Scientific and Technological Aspects in Mechanics*. Springer Berlin Heidelberg.

Khan, Rehman M. (2013). *Problem Solving and Data Analysis Using Minitab. A Clear and Easy Guide to Six Sigma Methodology*. John Wiley & Sons.

Deb, K. et al. (2002). „A fast and elitist multiobjective genetic algorithm: NSGA-II“. In: *IEEE Transactions on Evolutionary Computation* 6, pp. 182–197. DOI: 10.1109/4235.996017.

Srinivas, N. and K. Deb (1994). „Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms“. In: *Evolutionary Computation* 2, pp. 221–248. DOI: 10.1162/evco.1994.2.3.221.

Lin, Song (2011). *NGPM – A NSGA-II Program in Matlab v1.4*. Version 1.6.0.0. URL: https://se.mathworks.com/matlabcentral/fileexchange/31166-ngpm-a-nsga-ii-program-in-matlab-v1-4 (visited on 04/10/2019).

VDI, Gesellschaft Verfahrenstechnik und Chemieingenieurwesen (2010). *VDI Heat Atlas*. 2nd ed. Springer. DOI: 10.1007/978-3-540-77877-6.

Zitzler, Eckart, Kalyanmoy Deb, and Lothar Thiele (2000). „Comparison of Multiobjective Evolutionary Algorithms: Empirical Results“. In: *Evolutionary Computation* 8, pp. 173 –195. DOI: 10.1162/106365600568202.

Appendix A

Model data

A.1 Figures of temperature and velocity

This section contains the velocity and temperature fields, as well as the velocity streamlines, for select configurations. Fluid temperature fields are shown in figure A.1 on page 74. The temperature distribution in the fins can be seen in figure A.2 on page 75. Velocity field of the configurations is shown in figure A.3 on page 76. Finally, the velocity streamlines can be seen on figure A.4 on page 77

A.2 CFD Data

Data from the CFD model used for the response surface model generation. As the table is too large to be put on the same page as this text, it can be seen on page 78. The numbers in parenthesis indicate the design level. This can be helpful as not all configurations fall under high (1), low (-1), or middle (0).

A.3 Multiple regression

The data from the multiple regression is shown in table A.2 and A.3.

A.3.1 Multiple regression models

A.4 Pareto Solutions

Solutions from the NSGA-II algorithm on the RSM from the DoE. (As the set is extensive, only 20 are shown here.

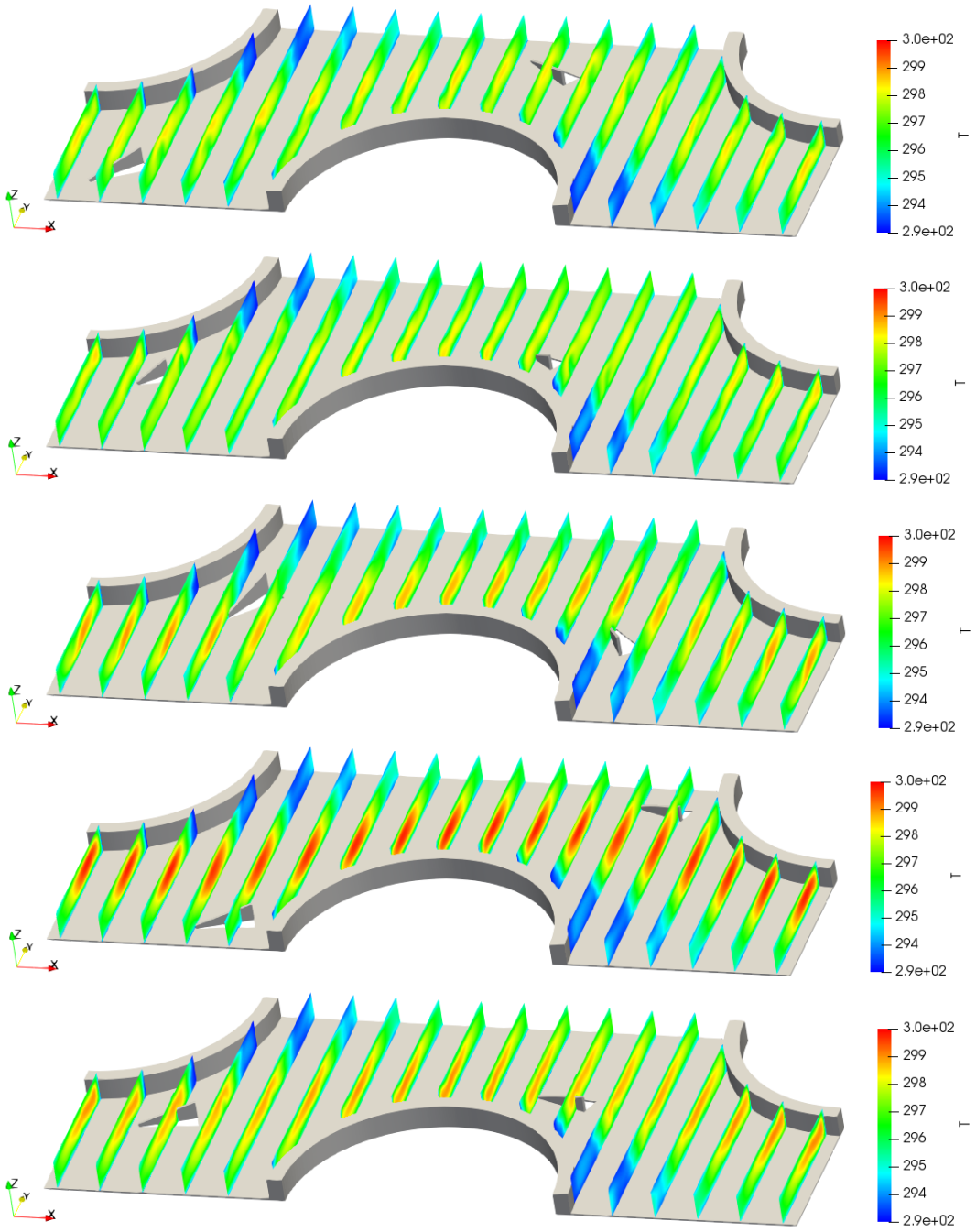


Figure A.1: Comparison of the temperature fields of configurations 1, 5, 16, 17, and 26. Temperature is in [K]

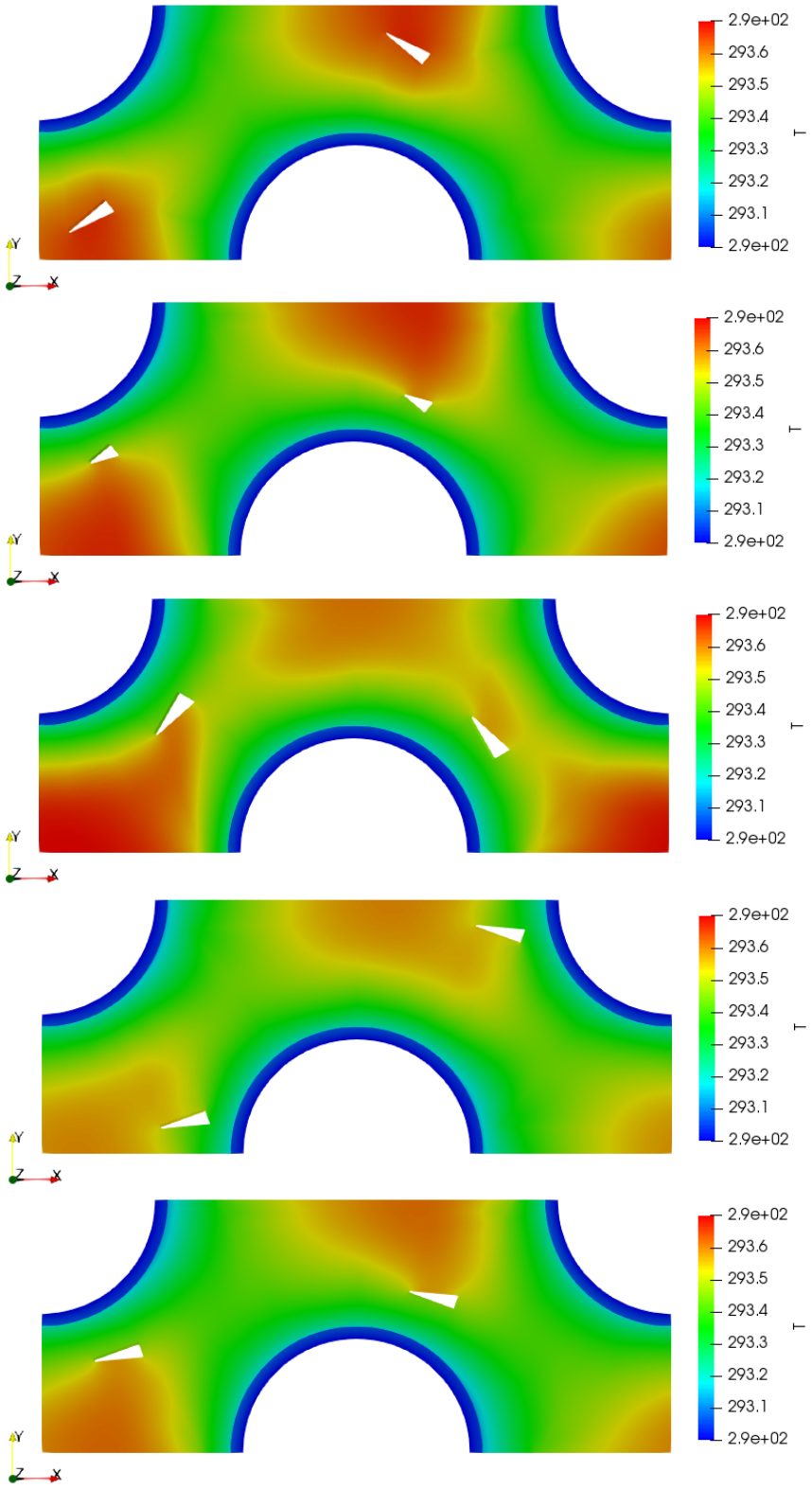


Figure A.2: Comparison of the fin temperature fields of configurations 1, 5, 16, 17, and 26. Temperature is in [K]

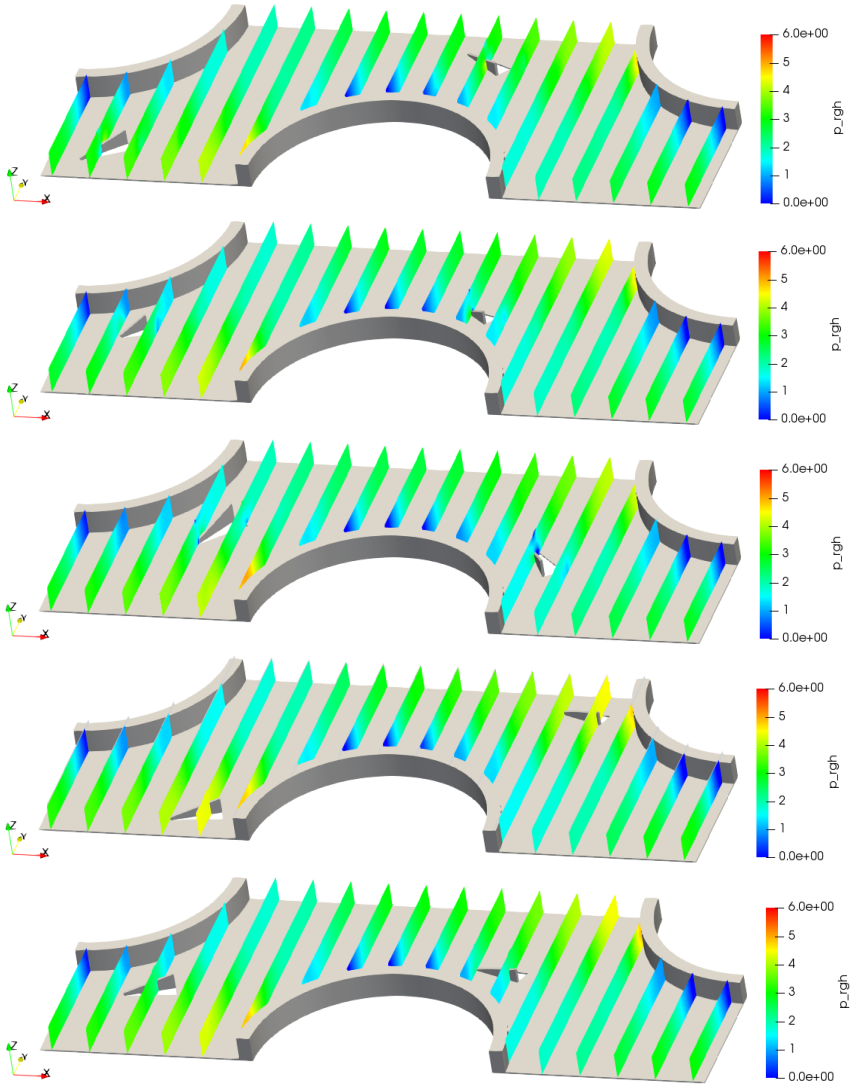


Figure A.3: Comparison of the pressure difference fields of configurations 1, 5, 16, 17, and 26. Pressure is kinematic pressure in $[m^2/s^2]$

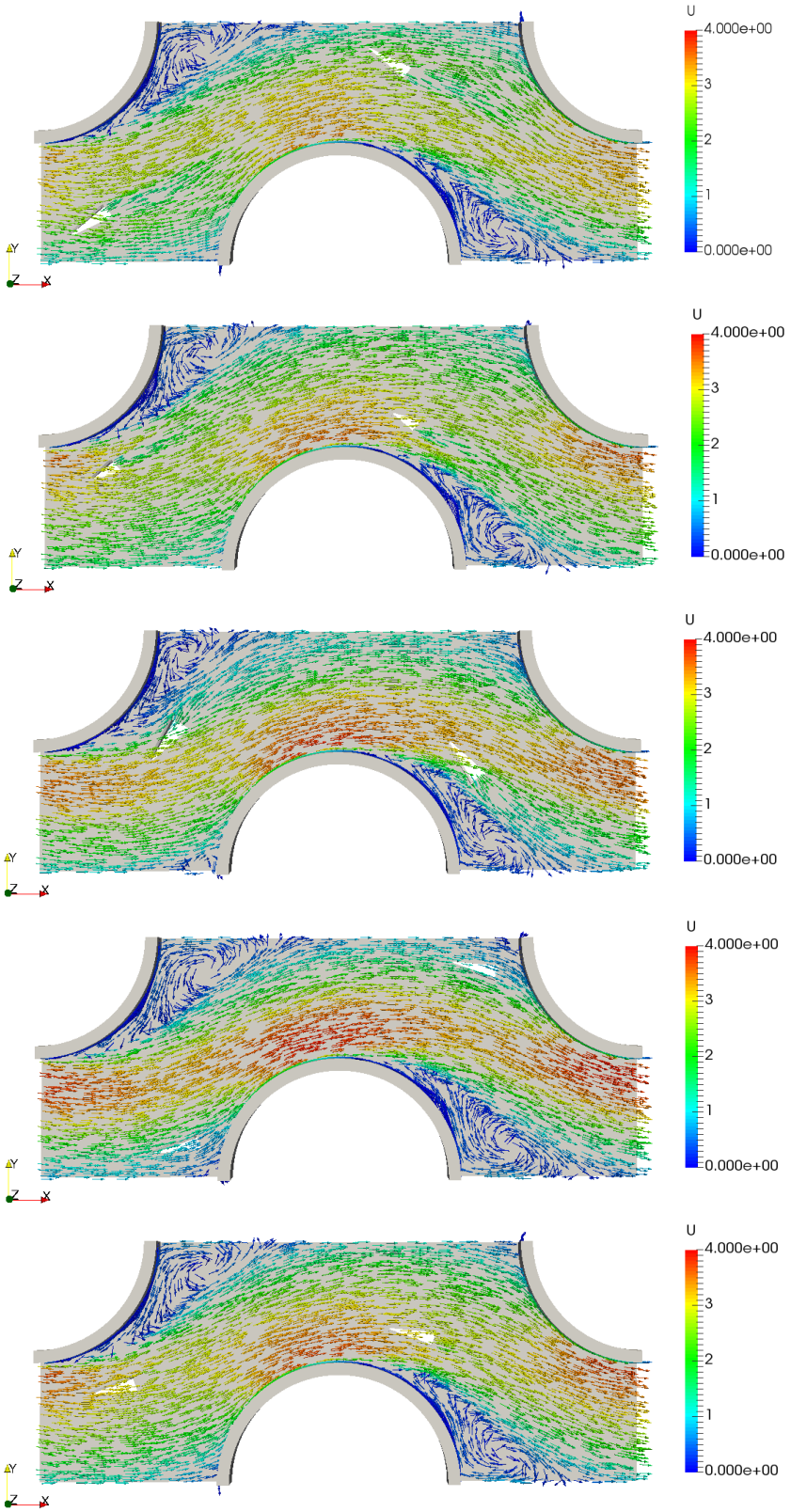


Figure A.4: Comparison of the velocity vectors of configurations 1, 5, 16, 17, and 26. Velocity is in [m/s]

Table A.1: Box-Behnken data from the CFD Model

# [-]	X-pos [mm]	Y-pos [mm]	Length [mm]	Height [mm]	Angle [°]	f/f_0 [-]	Nu/Nu_0 [-]
1	2 (-1)	2 (-1)	4.2 (0)	1.1 (0)	40 (0)	1.0784	1.1644
2	12.5 (+0.5)	5.5 (-0.5)	4.2 (0)	1.1 (0)	40 (0)	1.0692	1.0761
3	7.25 (-0.25)	10.75 (+0.25)	4.2 (0)	1.1 (0)	40 (0)	1. 0517	1.0651
4	16 (+1)	16 (+1)	4.2 (0)	1.1 (0)	40 (0)	1.0192	0.9618
5	3.75 (-0.75)	7.25 (-0.25)	2.4 (-1)	1.1 (0)	40 (0)	1.0319	1.1436
6	16 (+1)	9 (0)	2.4 (-1)	1.1 (0)	40 (0)	1.0418	0.9741
7	3.75 (-0.75)	7.25 (-0.25)	6 (+1)	1.1 (0)	40 (0)	1.0252	1.0984
8	16 (+1)	9 (0)	6 (+1)	1.1 (0)	40 (0)	1.0391	0.9912
9	9 (0)	9 (0)	2.4 (-1)	0.8 (-1)	40 (0)	1.0729	1.0167
10	9 (0)	9 (0)	6 (+1)	0.8 (-1)	40 (0)	1.1155	0.9469
11	9 (0)	9 (0)	2.4 (-1)	1.4 (+1)	40 (0)	1.0856	1.0429
12	9 (0)	9 (0)	6 (+1)	1.4 (+1)	40 (0)	1.0701	0.0707
13	9 (0)	9 (0)	4.2 (0)	0.8 (-1)	20 (-1)	1.0586	1.0286
14	9 (0)	9 (0)	4.2 (0)	1.4 (+1)	20 (-1)	1.0519	1.0702
15	9 (0)	9 (0)	4.2 (0)	0.8 (-1)	60 (+1)	1.0922	1.0516
16	9 (0)	9 (0)	4.2 (0)	1.4 (+1)	60 (+1)	1.1567	1.0792
17	9 (0)	2 (-1)	4.2 (0)	1.1 (0)	20 (-1)	1.0470	0.9476
18	10.75 (+0.25)	14.25 (+0.75)	4.2 (0)	1.1 (0)	20 (-1)	1.0385	0.9721
19	9 (0)	2 (-1)	4.2 (0)	1.1 (0)	60 (+1)	1.0749	1.1246
20	10.75 (+0.25)	14.25 (+0.75)	4.2 (0)	1.1 (0)	60 (+1)	1.0493	1.0079
21	9 (0)	9 (0)	4.2 (0)	1.1 (0)	40 (0)	1.0855	1.0618
22	3.75 (-0.75)	7.25 (-0.25)	4.2 (0)	0.8 (-1)	40 (0)	1.0286	1.0353
23	16 (+1)	9 (0)	4.2 (0)	0.8 (-1)	40 (0)	1.0209	0.9827
24	3.75 (-0.75)	7.25 (-0.25)	4.2 (0)	1.4 (+1)	40 (0)	1.0585	1.1624
25	16 (+1)	9 (0)	4.2 (0)	1.4 (+1)	40 (0)	1.0640	1.0417
26	3.75 (-0.75)	7.25 (-0.25)	4.2 (0)	1.1 (0)	20 (-1)	1.0071	1.0295
27	16 (+1)	9 (0)	4.2 (0)	1.1 (0)	20 (-1)	1.0597	1.0364
28	3.75 (-0.75)	7.25 (-0.25)	4.2 (0)	1.1 (0)	60 (+1)	1.1033	1.1460
29	16 (+1)	9 (0)	4.2 (0)	1.1 (0)	60 (+1)	1.0626	1.0267
30	9 (0)	2 (-1)	2.4 (-1)	1.1 (0)	40 (0)	1.0469	0.9900
31	10.75 (+0.25)	14.25 (+0.75)	2.4 (-1)	1.1 (0)	40 (0)	1.0606	1.0412
32	9 (0)	2 (-1)	6 (+1)	1.1 (0)	40 (0)	1.0203	1.0308
33	10.75 (+0.25)	14.25 (+0.75)	6 (+1)	1.1 (0)	40 (0)	1.0427	1.0104
34	9 (0)	2 (-1)	4.2 (0)	0.8 (-1)	40 (0)	1.0273	1.0250
35	10.75 (+0.25)	14.25 (+0.75)	4.2 (0)	0.8 (-1)	40 (0)	1.0134	0.9620
36	9 (0)	2 (-1)	4.2 (0)	1.4 (+1)	40 (0)	1.0313	1.0426
37	10.75 (+0.25)	14.25 (+0.75)	4.2 (0)	1.4 (+1)	40 (0)	1.0196	0.9563
38	9 (0)	9 (0)	2.4 (-1)	1.1 (0)	20 (-1)	1.0580	1.0296
39	9 (0)	9 (0)	6 (+1)	1.1 (0)	20 (-1)	1.0468	1.0352
40	9 (0)	9 (0)	2.4 (-1)	1.1 (0)	60 (+1)	1.1404	1.1003
41	9 (0)	9 (0)	6 (+1)	1.1 (0)	60 (+1)	1.1118	1.0527

Table A.2: Quadratic model of Nu/Nu_0

Coef.	Estimate	Std. Error	T-value	P-value
Intercept	2.500e-01	3.756e-01	0.666	0.51329
d_x	-3.274e-03	1.445e-02	-0.227	0.82305
d_y	3.148e-02	1.414e-02	2.226	0.03769
l	5.133e-02	5.440e-02	0.944	0.35664
h	7.150e-01	3.912e-01	1.828	0.08252
α	9.930e-03	4.699e-03	2.113	0.04734
d_x:d_y	1.570e-03	1.018e-03	1.543	0.13851
d_x:l	1.561e-03	1.467e-03	1.064	0.29994
d_x:h	-8.048e-03	8.804e-03	-0.914	0.37156
d_x:α	-1.845e-04	1.321e-04	-1.397	0.17766
d_y:l	-1.746e-03	1.467e-03	-1.190	0.24808
d_y:h	-2.829e-03	8.804e-03	-0.321	0.75130
d_y:α	-2.930e-04	1.321e-04	-2.219	0.03823
l:h	7.908e-03	2.955e-02	0.268	0.79176
l:α	-3.693e-04	4.433e-04	-0.833	0.41470
h:α	-5.815e-04	2.660e-03	-0.219	0.82917
d_x^2	-3.685e-04	5.790e-04	-0.636	0.53173
d_y^2	-1.680e-03	5.790e-04	-2.901	0.00883
l^2	-5.462e-03	4.304e-03	-1.269	0.21897
h^2	-2.522e-01	1.549e-01	-1.627	0.11931
α^2	-2.644e-05	3.486e-05	-0.758	0.45708

Table A.3: Quadratic model of f/f_0

Coef.	Estimate	Std. Error	T-value	P-value
Intercept	8.396e-01	1.900e-01	4.418	0.000265
d_x	1.647e-02	7.310e-03	2.253	0.035661
d_y	2.579e-03	7.157e-03	0.360	0.722322
l	2.177e-02	2.752e-02	0.791	0.438336
h	2.132e-01	1.979e-01	1.077	0.294322
α	-1.452e-03	2.378e-03	-0.611	0.548300
d_x:d_y	2.333e-03	5.149e-04	4.531	0.000204
d_x:l	1.397e-04	7.425e-04	0.188	0.852663
d_x:h	1.744e-03	4.455e-03	0.391	0.699654
d_x:α	-2.124e-04	6.682e-05	-3.178	0.004728
d_y:l	1.110e-04	7.425e-04	0.150	0.882644
d_y:h	8.456e-05	4.455e-03	0.019	0.985044
d_y:α	1.526e-05	6.682e-05	0.228	0.821637
l:h	-5.114e-05	1.495e-02	-0.003	0.997305
l:α	-1.211e-04	2.243e-04	-0.540	0.595331
h:α	2.967e-03	1.346e-03	2.204	0.039380
d_x^2	-1.485e-03	2.930e-04	-5.067	5.89e-05
d_y^2	-1.709e-03	2.930e-04	-5.833	1.05e-05
l^2	-2.870e-03	2.178e-03	-1.318	0.202498
h^2	-1.428e-01	7.840e-02	-1.821	0.083576
α^2	2.368e-05	1.764e-05	1.342	0.194575

Table A.4: Multiple regression on Nu/Nu_0

Coef.	Estimate	Std. Error	T-value	P-value
Intercept	2.500e-01	3.756e-01	0.666	0.51329
d_x	-3.274e-03	1.445e-02	-0.227	0.82305
d_y	3.148e-02	1.414e-02	2.226	0.03769
l	5.133e-02	5.440e-02	0.944	0.35664
h	7.150e-01	3.912e-01	1.828	0.08252
α	9.930e-03	4.699e-03	2.113	0.04734
d_x:d_y	1.570e-03	1.018e-03	1.543	0.13851
d_x:l	1.561e-03	1.467e-03	1.064	0.29994
d_x:h	-8.048e-03	8.804e-03	-0.914	0.37156
d_x:α	-1.845e-04	1.321e-04	-1.397	0.17766
d_y:l	-1.746e-03	1.467e-03	-1.190	0.24808
d_y:h	-2.829e-03	8.804e-03	-0.321	0.75130
d_y:α	-2.930e-04	1.321e-04	-2.219	0.03823
l:h	7.908e-03	2.955e-02	0.268	0.79176
l:α	-3.693e-04	4.433e-04	-0.833	0.41470
h:α	-5.815e-04	2.660e-03	-0.219	0.82917
d_x^2	-3.685e-04	5.790e-04	-0.636	0.53173
d_y^2	-1.680e-03	5.790e-04	-2.901	0.00883
l^2	-5.462e-03	4.304e-03	-1.269	0.21897
h^2	-2.522e-01	1.549e-01	-1.627	0.11931
α^2	-2.644e-05	3.486e-05	-0.758	0.45708

Table A.5: Multiple regression on f/f_0

Coef.	Estimate	Std. Error	T-value	P-value
Intercept	8.280e-01	1.396e-01	5.930	2.54e-06
d_x	1.882e-02	3.830e-03	4.913	3.85e-05
d_y	3.749e-03	2.808e-03	1.335	0.192953
l	1.914e-02	1.609e-02	1.189	0.244796
h	2.301e-01	1.582e-01	1.454	0.157337
α	-1.866e-03	1.872e-03	-0.997	0.327626
d_x:d_y	2.333e-03	4.496e-04	5.189	1.84e-05
d_x:α	-2.085e-04	5.643e-05	-3.695	0.000987
h:α	2.967e-03	1.175e-03	2.524	0.017779
d_x^2	-1.485e-03	2.559e-04	-5.803	3.56e-06
d_y^2	-1.709e-03	2.559e-04	-6.679	3.61e-07
l^2	-2.870e-03	1.902e-03	-1.509	0.142918
h^2	-1.428e-01	6.846e-02	-2.086	0.046604
α^2	2.368e-05	1.540e-05	1.537	0.135927

Table A.6: Pareto solutions from NSGA-II

X-pos [mm]	Y-pos [mm]	Length [mm]	Height [mm]	Angle [°]	f/f_0 [-]	Nu/Nu_0 [-]
10.2470	16.0000	6.0000	0.8000	20	0.9357	0.9008
6.3857	5.2091	2.9018	1.4000	60	1.0720	1.2407
6.6675	2.0000	2.4000	0.8000	20	1.0244	0.9058
10.6252	16.0000	6.0000	0.8000	20	0.9433	0.9058
8.2830	6.6075	2.7658	1.4000	60	1.0693	1.1996
12.0830	9.9038	3.7588	1.0178	20	1.0693	1.0310
11.4197	9.3325	2.6954	0.8816	20	1.0635	0.9951
6.9816	5.6716	2.8863	1.4000	60	1.0717	1.2281
6.9247	3.1053	2.4000	0.8000	20	1.0352	0.9265
7.7496	5.1094	2.4000	0.8000	20	1.0482	0.9529
7.3856	4.4879	2.4000	0.8005	20	1.0448	0.9468
11.9845	9.7731	3.2348	0.9451	20	1.0677	1.0149
11.3262	9.0550	2.4000	0.8000	20	1.0576	0.9718
7.3203	4.1702	2.4000	0.8000	20	1.0431	0.9421
10.3284	16.000	6.0000	0.8000	20	0.9374	0.9019
8.5268	6.3540	2.4000	0.8000	20	1.0531	0.9632
7.4619	6.3020	2.7800	1.4000	60	1.0709	1.2159
6.6402	2.0947	2.4000	0.8000	20	1.0255	0.9083
7.7526	6.3707	2.8766	1.3999	60	1.0705	1.2112
7.1640	3.9304	2.4000	0.8000	20	1.0414	0.9394

Appendix B

OpenFOAM models

Here the applied methods indicated in chapter 4.3 are explained briefly.

B.1 Discretization schemes

Interpolation These are used to get face-centred quantities from cell-centred ones. By default, it is unused but applied specifically to the gradient, divergence, and Laplacian terms.

Gradient: Gauss The scheme calculates cell gradient using Gauss' theorem:

$$\int_V (\nabla \cdot \boldsymbol{u})\, dV = \oint_S (\boldsymbol{n} \cdot \boldsymbol{u})\, dS$$

Divergence: Gauss upwind & Gauss linear Upwind divergence is first-order bounded.

$$\varphi_f = \varphi_C$$

Linear divergence is also known as midpoint. It is first/second order unbounded. Equal to linear for isotropic meshes. Also exists as Bounded central difference.

$$\varphi_f = 0,5 \cdot (\varphi_C + \varphi_D)$$

Laplacian Resolution of the Laplacian is based on the Gauss theorem and needs an interpolation scheme and a surface-normal scheme.

Surface-normal gradient: Corrected Central-difference surface-normal gradient scheme with non-orthogonal correction. Explicit non-orthogonal correction.

$$\nabla_{\perp f} Q = \frac{1}{\cos\theta} \frac{Q_P - Q_N}{|\boldsymbol{d}|} + \left(\hat{\boldsymbol{n}} - \frac{1}{\cos\theta} \hat{\boldsymbol{d}} \right) \cdot (\nabla Q)_f$$

B.2 Solvers

BiCGStab More stable version of Bi-Conjugate Gradient, which should not be used.

PCG Preconditioned Conjugate Gradient for symmetric matrices with good parallel scaling.

B.3 Preconditioners

Cholesky Incomplete Cholesky preconditioning with no fill-in.

DIC Simplified Diagonal-based Incomplete Cholesky preconditioner for symmetric matrices. Parallel inconsistent.

DILU Simplified Diagonal-based Incomplete LU preconditioner for asymmetric matrices. Parallel inconsistent.

B.4 Boundary conditions for the computational domain

Boundary conditions for the CFD model, defined for every parameter. The solid and fluid are shown in their separate tables.

Boundary conditions for solid:

Parameter	Boundary	Type	Value	
			Initial	Fixed
k	internalField		100	
	walls	chtRcThermalDiffusivity		10
	Top/Bottom/Front/Back	symmetryPlane		
	Tube	zeroGradient		
	In/Out	cyclic		
Temperature	internalField		293	
	walls	chtRcTemperature		293
	Top/Bottom/Front/Back	symmetryPlane		
	Tube	fixedValue		293
	In/Out	cyclic		

Boundary conditions for fluid:				
Parameter	Boundary	Type	Value	
			Initial	Fixed
Pressure	internalField		0	
	Top/Bottom	symmetryPlane		
	fluidWalls	buoyantPressure		
	In/Out, Hole Front/Back	cyclic		
Temperature	internalField		303	
	Top/Bottom	symmetryPlane		
	fluidWalls	chtRcTemperature		303
	In/Out, Hole Front/Back	cyclic		
Velocity	internalField		(2 0 0)	
	Top/Bottom	symmetryPlane		
	fluidWalls	fixedValue		(0 0 0)
	In/Out, Hole Front/Back	cyclic		
k	internalField		$0,24$	
	Top/Bottom	symmetryPlane		
	fluidWalls	kqRWallFunction		$0,24$
	In/Out, Hole Front/Back	cyclic		
κ_{Eff}	internalField		$0,001$	
	Top/Bottom	symmetryPlane		
	fluidWalls	chtRcThermalDiffusivity		$0,001$
	In/Out, Hole Front/Back	cyclic		
ν_t	internalField		0	
	Top/Bottom	symmetryPlane		
	fluidWalls	nutWallFunction		0
	In/Out, Hole Front/Back	cyclic		
ω	internalField		$1,78$	
	Top/Bottom	symmetryPlane		
	fluidWalls	omegaWallFunction		$1,78$
	In/Out, Hole Front/Back	cyclic		

Supplements III

MATLAB Code

III.1 License for NGPM – A NSGA_II Program in Matlab v1.4

```
Copyright (c) 2011, Song Lin
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the distribution

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

III.2 Nondominated Sorting Approach

```
function [opt, pop] = ndsort(opt, pop)
% Function: [opt, pop] = ndsort(pop)
% Description: Fast non-dominated sort.
```

```
%**************************************************************************
% 1. Initialize variables
%   indi.np: number of individuals which dominate this individual
%   indi.sp(:): a set of individuals that this individual dominate
%**************************************************************************
N = length(pop);    %popsize
ind = repmat(struct('np',0, 'sp', []),[1,N]);   %individual structure

for i = 1:N
    pop(i).rank = 0;
    pop(i).distance = 0;
    pop(i).prefDistance = 0;
end

%**************************************************************************
% 2. fast non-dominated sort
%**************************************************************************
% Calculate the domination matrix for improving the efficiency.
nViol   = zeros(N, 1);
violSum = zeros(N, 1);
for i = 1:N
    nViol(i)    = pop(i).nViol;
    violSum(i)  = pop(i).violSum;
end

obj     = vertcat(pop(:).obj);
domMat  = calcDominationMatrix(nViol, violSum, obj);

% Compute np and sp of each indivudal
for p = 1:N-1
    for q = p+1:N
        if(domMat(p, q) == 1)           % p dominate q
            ind(q).np = ind(q).np + 1;
            ind(p).sp = [ind(p).sp , q];
        elseif(domMat(p, q) == -1)      % q dominate p
            ind(p).np = ind(p).np + 1;
            ind(q).sp = [ind(q).sp , p];
        end
    end
end

% The first front(rank = 1)
front(1).f = [];    % There are only one field 'f' in structure 'front'.
                    % This is intentional because the number of individuals
                    % in the front is different.
for i = 1:N
    if( ind(i).np == 0 )
        pop(i).rank = 1;
        front(1).f = [front(1).f, i];
    end
end

% Calculate pareto rank of each individuals, viz., pop(:).rank
fid = 1;        %pareto front ID
while( ~isempty(front(fid).f) )
    Q = [];
```

```
    for p = front(fid).f
        for q = ind(p).sp
            ind(q).np = ind(q).np -1;
            if( ind(q).np == 0 )
                pop(q).rank = fid+1;
                Q = [Q, q];
            end
        end
    end
    fid = fid + 1;

    front(fid).f = Q;
end
front(fid) = [];    % delete the last empty front set

%**************************************************************************
% 3. Calculate the distance
%**************************************************************************
if(isempty(opt.refPoints))
    pop = calcCrowdingDistance(opt, pop, front);
else
    [opt, pop] = calcPreferenceDistance(opt, pop, front);
end
```

III.2.1 Domination Matrix

```
function domMat = calcDominationMatrix(nViol, violSum, obj)
% Function: domMat = calcDominationMatrix(nViol, violSum, obj)
% Description: Calculate the domination maxtir which specified the domination
%   releation between two individual using constrained-domination.
%
% Return:
%   domMat(N,N) : domination matrix
%       domMat(p,q)=1  : p dominates q
%       domMat(p,q)=-1 : q dominates p
%       domMat(p,q)=0  : non dominate
%**************************************************************************
N       = size(obj, 1);
numObj  = size(obj, 2);
domMat  = zeros(N, N);

for p = 1:N-1
    for q = p+1:N
        %******************************************************************
        % 1. p and q are both feasible
        %******************************************************************
        if(nViol(p) == 0 && nViol(q)==0)
            pdomq = false;
            qdomp = false;
            for i = 1:numObj
                if( obj(p, i) < obj(q, i) )
                    pdomq = true;
                elseif(obj(p, i) > obj(q, i))
                    qdomp = true;
```

```
                end
            end

            if( pdomq && ~qdomp )
                domMat(p, q) = 1;
            elseif(~pdomq && qdomp )
                domMat(p, q) = -1;
            end
        %**************************************************************
        % 2. p is feasible, and q is infeasible
        %**************************************************************
        elseif(nViol(p) == 0 && nViol(q)~=0)
            domMat(p, q) = 1;
        %**************************************************************
        % 3. q is feasible, and p is infeasible
        %**************************************************************
        elseif(nViol(p) ~= 0 && nViol(q)==0)
            domMat(p, q) = -1;
        %**************************************************************
        % 4. p and q are both infeasible
        %**************************************************************
        else
            if(violSum(p) < violSum(q))
                domMat(p, q) = 1;
            elseif(violSum(p) > violSum(q))
                domMat(p, q) = -1;
            end
        end
    end
end
domMat = domMat - domMat';
```

III.2.2 Crowding Distance

```
function pop = calcCrowdingDistance(opt, pop, front)
% Function: pop = calcCrowdingDistance(opt, pop, front)
% Description: Calculate the 'crowding distance' used in the original NSGA-II.
%**************************************************************************
numObj = length( pop(1).obj );  % number of objectives
for fid = 1:length(front)
    idx = front(fid).f;
    frontPop = pop(idx);        % Individuals in front fid

    numInd = length(idx);       % Number of individuals in current front

    obj = vertcat(frontPop.obj);
    obj = [obj, idx'];          % objctive values are sorted with ID
    for m = 1:numObj
        obj = sortrows(obj, m);

        colIdx = numObj+1;
        pop( obj(1, colIdx) ).distance = Inf;         % the first one
        pop( obj(numInd, colIdx) ).distance = Inf;    % the last one
```

```
        minobj = obj(1, m);          % the maximum of objective m
        maxobj = obj(numInd, m);     % the minimum of objective m

        for i = 2:(numInd-1)
            id = obj(i, colIdx);
            pop(id).distance = pop(id).distance ...
                + (obj(i+1, m) - obj(i-1, m)) / (maxobj - minobj);
        end
    end
end
```

III.3 Binary Tournament Selection

```
function newpop = selectOp(opt, pop)
% Function: newpop = selectOp(opt, pop)
% Description: Selection operator, use binary tournament selection.
%*********************************************************************
popsize = length(pop);      % Size of population
pool = zeros(1, popsize);   % pool : the individual index selected
% 2 popsize integers between 0 and popsize
randnum = randi(popsize, [1, 2 * popsize]);

j = 1;
for i = 1:2:(2*popsize)
    p1 = randnum(i);
    p2 = randnum(i+1);

    if(isempty(opt.refPoints))
        % Crowded-comparison operator (NSGA-II)
        result = crowdingComp( pop(p1), pop(p2) );
    end

    if(result == 1)
        pool(j) = p1;
    else
        pool(j) = p2;
    end
    j = j + 1;
end
newpop = pop( pool );

function result = crowdingComp( guy1, guy2)
% Function: result = crowdingComp( guy1, guy2)
% Description: Crowding comparison operator.
% Return:
%   1 = guy1 is better than guy2
%   0 = other cases
%*********************************************************************
if(  (guy1.rank  < guy2.rank) || ...
    ((guy1.rank == guy2.rank) && (guy1.distance > guy2.distance) ))
    result = 1;
else
    result = 0;
end
```

III.4 Crossover operation

```
function pop = crossoverOp(opt, pop, state)
% Function: pop = crossoverOp(opt, pop, state)
% Description: Crossover operator. All of the individuals will do crossover, but
%   only "crossoverFraction" of design variables of an individual will change.
%**************************************************************************

%**************************************************************************
% 1. Check for the parameters
%**************************************************************************
% determine the crossover method
strfun = lower(opt.crossover{1});
numOptions = length(opt.crossover) - 1;
[crossoverOpt{1:numOptions}] = opt.crossover{2:end};

switch( strfun )
    case 'intermediate'
        fun = @crsIntermediate;
    case 'sbx'
        fun = @SBX;
    otherwise
        error('NSGA2:CrossoverOpError', 'No support crossover operator!');
end

nVar = opt.numVar;

% "auto" crossover fraction in case none given
if( ischar(opt.crossoverFraction) )
    if( strcmpi(opt.crossoverFraction, 'auto') )
        fraction = 2.0 / nVar;
    else
        error('NSGA2:CrossoverOpError', ...
            'The "crossoverFraction" parameter should be scalar or "auto" string.');
    end
else
    fraction = opt.crossoverFraction;
end


for ind = 1:2:length(pop)     % Popsize should be even number
        % Create children
        [child1, child2] = fun( pop(ind), pop(ind+1), fraction, crossoverOpt );

        % Round
        for v = 1:nVar
            if( opt.vartype(v) == 2)
                child1.var(v) = round( child1.var(v) );
                child2.var(v) = round( child2.var(v) );
            end
        end

        % Bounding limit
```

```
        child1.var = varlimit(child1.var, opt.lb, opt.ub);
        child2.var = varlimit(child2.var, opt.lb, opt.ub);

        pop(ind)    = child1;
        pop(ind+1)  = child2;

end
```

III.4.1 Simulated Binary Crossover

```
function [child1, child2] = SBX(parent1, parent2, fraction, options)
% Function: [child1, child2] = SBX(parent1, parent2, fraction, options)
% Description: Simulated Binary crossover.
% child = 1/2 * (parent1 * (1 + rand) + parent2 * (1 - rand))
% Parameters:
%   fraction : crossover fraction of variables of an individual
%   options = distribution parameter
% Simulated Binary Crossover (SBX)
if( length(options)~=1 || ~isnumeric(options{1}))
    error('NSGA2:CrossoverOpError', 'Crossover operator parameter error!');
end

mu = options{1};
% Copy structure from parents
child1 = parent1;
child2 = parent2;

nVar = length(parent1.var);
crsFlag = rand(1, nVar) < fraction; % Probability of crossover
for j = 1:nVar
    % Generate a random number
    u = rand(1);
    % Find the corresponding spread in a mu dependent distribution
    if u <= 0.5
        bq = (2*u)^(1/(mu+1));
    else
        bq = (1/(2*(1 - u)))^(1/(mu+1));
    end
    % Generate the elements of the children
    if crsFlag(j)
        child1.var(j) = 0.5 * ((1 + bq) * parent1.var(j) + ...
                        (1 - bq) * parent2.var(j));
        child2.var(j) = 0.5 * ((1 - bq) * parent2.var(j) + ...
                        (1 + bq) * parent2.var(j));
    end
end
```

III.5 Mutation operation

```
function pop = mutationOp(opt, pop, state)
% Function: pop = mutationOp(opt, pop, state)
% Description: Mutation Operator. All of the individuals will do mutation, but
```

```
%   only "mutationFraction" of design variables of an individual will change.
%*********************************************************************

%*********************************************************************
% 1. Check for the parameters
%*********************************************************************
% mutation method
strfun = lower(opt.mutation{1});
numOptions = length(opt.mutation) - 1;
[mutationopt{1:numOptions}] = opt.mutation{2:end};

switch (strfun)
    case 'gaussian'
        fun = @mutationGaussian;
    case 'poly'
        fun = @mutationPolynomial;
    otherwise
        error('NSGA2:MutationOpError', 'No support mutation operator!');
end

nVar = opt.numVar;

% "auto" mutation fraction
if( ischar(opt.mutationFraction) )
    if( strcmpi(opt.mutationFraction, 'auto') )
        fraction = 2.0 / nVar;
    else
        error('NSGA2:MutationOpError',...
            'The "mutationsFraction" parameter should be scalar or "auto" string.');
    end
else
    fraction = opt.mutationFraction;
end

% All of the individual would be modified, but only 'mutationFraction' of design
% variables for an individual would be changed.
for ind = 1:length(pop)
        child = fun( pop(ind), opt, state, fraction, mutationopt);

        % Rounding for integer variables
        for v = 1:nVar
            if( opt.vartype(v) == 2)
                child.var(v) = round( child.var(v) );
            end
        end

        child.var = varlimit(child.var, opt.lb, opt.ub);

        pop(ind) = child;
end
```

III.5.1 Polynomial mutation

```
function child = mutationPolynomial(parent,opt,state,fraction,options)
```

```
% Simulated Binary Crossover mutation
if( length(options)~=1)
    error('NSGA2:MutationOpError', 'Mutation operator parameter error!');
end

nVar = length(parent.var);
mutFlag = rand(1, nVar) < fraction; % Does mutation happen?

mu = options{1};
lb = opt.lb;
ub = opt.ub;

child = parent;
% Addition of shrinkage: mutation means less as the program progresses
% shrink = 1 - 0.25 * 1 * state.currentGen / opt.maxGen;

for j = 1:nVar
    r = rand(1);
    if r < 0.5
        delta = (2*r)^(1/(mu+1)) - 1;
    else
        delta = 1 - (2*(1 - r))^(1/(mu+1));
    end
    child.var(j) = child.var(j) + (ub(j)-lb(j))*delta*mutFlag(j);
end
```

Supplements IV

Conjugate heat transfer solver

This section serves to give a grief overview of how the modified conjugate heat transfer solver in FOAM-Extend operates. Additional files that were implemented, will also be included in this section. The SIMPLE-algorithm applied by this solver, is shown in figure IV.1.

IV.1 Modified momentum equation:

The calculation of pressure gradient needed to correct the pressure in the domain is done with a pressure gradient source, which can be seen as "momentumSource.Su()", below:

```
// Solve the momentum equation
tmp<fvVectorMatrix> UEqn
(
    fvm::div(phi, U)
  + turbulence->divDevReff()
  + fvm::SuSp(-fvc::div(phi), U)
   ==
     momentumSource.Su()
);
UEqn().relax();
solve
(
    UEqn()
  ==
    fvc::reconstruct
    (
        (
            (
              - ghf*fvc::snGrad(rhok)
              - fvc::snGrad(p_rgh)
            )*mesh.magSf()
        )
    )

);
```

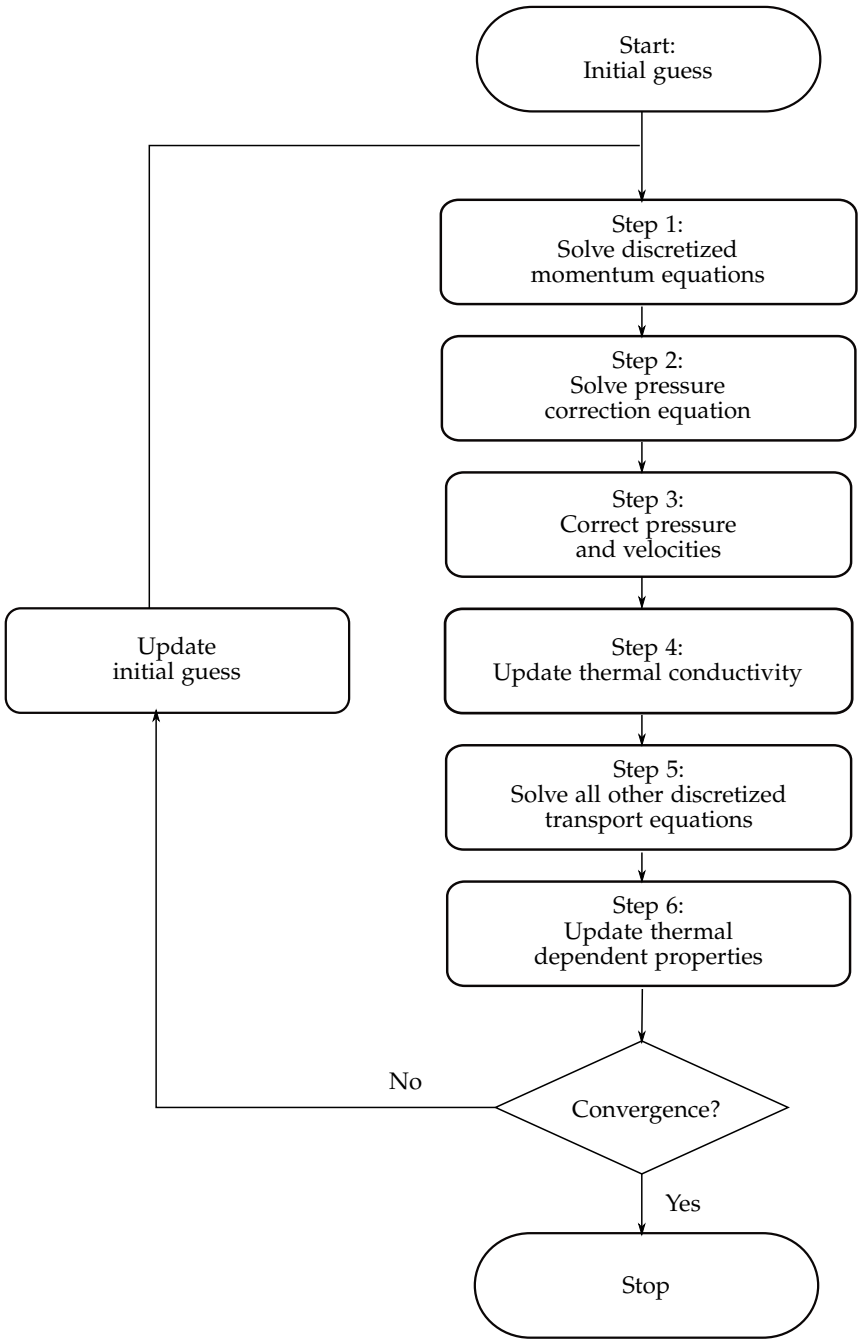


Figure IV.1: This flowchart shows the modified SIMPLE-Algorithm that is used for the model.

IV.2 Modified energy equation

To enable preiodic heat transfer as discussed earlier, the term "U.component(0)*gamma" is added to the energy equation of the fluid:

```
// Solid side
simpleControl simpleSolid(solidMesh);
```

```
 while (simpleSolid.correctNonOrthogonal())
{
    coupledFvScalarMatrix TEqns(2);
    fvScalarMatrix* TFluidEqn = new fvScalarMatrix
    (
        rho*Cp*
        (
            fvm::div(phi, T)
          + fvm::SuSp(-fvc::div(phi), T)
            + U.component(0)*gamma
        )
      - fvm::laplacian(kappaEff, T)
    );
    TFluidEqn->relax();
    fvScalarMatrix* TSolidEqn = new fvScalarMatrix
    (
      - fvm::laplacian(kSolidf, Tsolid, "laplacian(k,T)")
      + fvm::SuSp(-solidThermo.S()/Tsolid, Tsolid)
    );
    TSolidEqn->relax();
    // Add fluid equation
    TEqns.set(0, TFluidEqn);
    // Add solid equation
    TEqns.set(1, TSolidEqn);
    TEqns.solve();
}
```

IV.3 Calculation of pressure gradient

The code below show the implantation of the pressure gradient source.

```
Info<< "Creating momentum source\n" << endl;
pressureGradientExplicitSource momentumSource
(
    "momentumSource",
    U
);
```

IV.4 Temperature depended variables

The variables treated as temperature depended, are evaluated on the equations below, where the "A"-values are the intercept for linear correlation and the "B"-values are the slope.

```
dimensionedScalar nu
(
    "nu",
    dimensionSet(0, 2, -1, 0, 0, 0, 0),
    nuA.value()+(nuB.value()*Tbulk.value())
);
```

```
dimensionedScalar Pr
(
    "Pr",
    dimensionSet(0, 0, 0, 0, 0, 0, 0),
    PrA.value()+(PrB.value()*Tbulk.value())
);
dimensionedScalar rho
(
    "rho",
    dimensionSet(1, -3, 0, 0, 0, 0, 0),
    rhoA.value()+(rhoB.value()*Tbulk.value())
);

Info << "Kinematic viscosity, nu = " << nu.value() << endl;
Info << "Laminar Prandtl number, Pr = " << Pr.value() << endl;
Info << "Density, rho = " << rho.value() << endl;
```

IV.5 Calculation of Nusselt and Reynolds number

The calculation of Nusselt number based on the approach shown in chapter 2, is done by the piece of code below:

```
label wallsID = mesh.boundaryMesh().findPatchID("fluidWalls");
label inletID = mesh.boundaryMesh().findPatchID("inOut");

dimensionedScalar num
(
    "num",
    dimensionSet(0, 2, -1, 1, 0, 0, 0),
    gSum(T.boundaryField()[inletID]*U.boundaryField()[inletID].component(0)...
    *mesh.magSf().boundaryField()[inletID])
);
dimensionedScalar den
(
    "den",
    dimensionSet(0, 2, -1, 0, 0, 0, 0),
    gSum(U.boundaryField()[inletID].component(0)*mesh.magSf().boundaryField()[inletID])
);
dimensionedScalar Tbulk
(
    num/den
);
dimensionedScalar dTdn
(
    "dTdn",
    dimensionSet(0, -1, 0, 1, 0, 0, 0),
    gSum(T.boundaryField()[wallsID].snGrad()*mesh.magSf().boundaryField()[wallsID])
    /gSum(mesh.magSf().boundaryField()[wallsID])
);
dimensionedScalar Twall
(
    "Twall",
    dimensionSet(0, 0, 0, 1, 0, 0, 0),
```

```
    gSum(T.boundaryField()[wallsID]*mesh.magSf().boundaryField()[wallsID])
    /gSum(mesh.magSf().boundaryField()[wallsID])
);
dimensionedScalar Umax
(
    "Umax",
    dimensionSet(0, 1, -1, 0, 0, 0, 0),
    gSum(U.boundaryField()[inletID].component(0)*mesh.magSf().boundaryField()[inletID])
    / gSum(mesh.magSf().boundaryField()[inletID])
);
Nu = dTdn*Lc/(Twall-Tbulk);

Re = (Umax*Lc)/nu;

Info << "Cyclic temperature field solved with:" << endl;
Info << "Nusselt number, Nu = " << Nu.value() << endl;
Info << "Reynolds number, Re = " << Re.value() << endl;
Info << "Average velocity at inlet, U = " << Umax.value() << endl;
Info << "Surface average temperature gradient, dT/dn = " << dTdn.value() << endl;
Info << "Characteristic length, Lc = " << Lc.value() << endl;
Info << "Surface-averaged wall temperature, Twall = " << Twall.value() << endl;
Info << "Bulk temperature, Tbulk = " << Tbulk.value() << endl;
Info << "Density, rho = " << rho.value() << endl;
```