THESIS FOR THE DEGREE OF LICENTIATE OF PHILOSOPHY

Waterjet Inlet Design Optimization Using CFD and Direct Search

Jakob Hultén

Department of Mathematics
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
Göteborg, April 2001

Waterjet Inlet Design Optimization Using CFD and Direct Search
Jakob Hultén

ISSN 0347-2809/No 2001:20
Department of Mathematics
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-7721000

Matematiskt Centrum
Göteborg, Sweden 2001

Abstract

We adress the problem of waterjet inlet design optimization using computational fluid dynamics (CFD) techniques and direct search methods. The latter are optimization methods that proceed without calculating derivatives. They are particularly advantageous in this kind of engineering design problem where the objective function depends on a costly computer simulation and therefore does not provide explicit information about derivatives.

The inlet has been modelled as an s-duct. A simple parameterization consisting of five parameters is used for varying the geometry. The CFD model of the flow in the duct is based on the Reynolds equations with the Standard k-ϵ model. The equations are discretized by a single-block structured grid and solved using the commercial software Fluent. The CFD model in Fluent is calibrated to give results in reasonable agreement with experimental data for flows through an s-duct.

A variant of the Hooke and Jeeves algorithm called the Sherif-Boice algorithm was used to optimize the parametric geometry for maximum static pressure (minimum risk of cavitation) and minimal loss. To reduce the cost of each evaluation of the objective we have considered surrogate functions (models of the objective) based on coarse grid flow computations. The surrogates were less costly to evaluate while capturing enough of the problem to be useful in the optimization.

It was found that a range of geometries were near-optimal with respect to both the cavitation and minimal loss objectives at the same time. One reason for this is a dependency between the objective functions under the given flow conditions. The optimal designs are characterized by a long, sweeping lower bend and a contraction of the duct over the upper, sharper bend. The analysis reveals interesting relationships between the optimal geometry and the considerably improved hydrodynamic performance.

The parametric model has been used to investigate the relation between the objective functions. Although they were dependent under the given conditions, it is not a generally valid statement. This is shown by an example, which also illustrates that cavitation-free performance should not be used as the only design objective. The example is built on the fact that the minimum static pressure may, for high enough inlet velocities, be attained at the inlet so that the cavitation objective function ignores the downstream flow behaviour.

In addition to the inlet design problem, we discuss a well-known class of direct methods called pattern search methods. The convergence analysis requires the objective to be continuously differentiable. The necessity of this condition is shown by an example of a Lipschitz continuous function for which coordinate search with fixed step length converges to a non-stationary point. We also use some basic theory of positive linear dependence to derive a few results about positive bases and simplicies in connection to optimization.

Keywords: optimization, optimal design, direct search, pattern search, surrogate objective, CFD, fluid dynamics, waterjet inlet, parametric geometry

Preface

About this report and ECMI

This report serves both as a licentiate thesis and the thesis of the ECMI[1] post-graduate program in applied mathematics. This five-semester program includes a block of core courses covering several areas of applied mathematics and a block of specialization courses within a selected field. One term is spent at another European university. The final part is to work with a mathematical problem from the industry.

Acknowledgements

First of all I thank my family for their love and support.

I sincerely thank my supervisor Assoc. Prof. Michael Patriksson. He has always taken his time to discuss problems and ideas and has given me many helpful comments on the manuscript.

I would also like to thank Johan Lennblad at Caran/VM-data, who was the initiator of the project. He has been my "industrial" supervisor and a great support through his advises, discussions and kind encouragement. He has also given me valuable comments on the manuscript.

Special thanks to Dr. Gregory Seil at Rolls-Royce AB, for sharing with me his broad knowledge of waterjets and computational fluid dynamics. His co-supervising has encouraged me to do my best on this project. I am also grateful to Lennart Berghult, at Rolls-Royce as well, for his support throughout the work and for inviting me to Kristinehamn. Rolls-Royce AB has also partly funded this work, which is gratefully acknowledged.

I would like to take the chance to thank all four gentlemen mentioned above for their advices and positive attitude during the hard times of this project.

I am indebted to Sara Ågren for lending me a C++ code she had done which was important for the geometric modelling of the problem. Thanks also to Stig-Ove Nilsson at the printing departement.

Thanks to my friends and colleagues, in particular my room-mate Per Hörfelt for his friendship and every day company.

Most of all I thank the Lord Jesus Christ. "He reached down from on high and took hold of me; he drew me out of deep waters."[2]

[1] The European Consortium for Mathematics in Industry

[2] 2 Sam 22:17

Contents

Notation **7**

1 Introduction **11**
1.1 Motivation and background 11
1.2 Design optimization 12
1.3 Outline of the report 14

2 Optimization without calculating derivatives **17**
2.1 Introduction 17
2.2 Brief overview of available methods 18
2.3 The general positive basis pattern search algorithm 19
2.3.1 Definition of the general algorithm 19
2.3.2 Some properties of the general algorithm 20
2.3.3 Bound constraints 23
2.4 Examples of pattern search algorithms 24
2.4.1 Coordinate search with fixed step length 24
2.4.2 The SBA 25
2.5 Minimal and maximal positive bases in pattern search 27
2.5.1 A minimal positive basis search algorithm 27
2.5.2 How large should the positive basis be? 27
2.6 Pattern search and surrogate functions 29

3 Fluid dynamics **33**
3.1 Introduction 33
3.2 Navier-Stokes equations 33
3.3 Reynolds number 35
3.4 Boundary layer 35
3.5 Turbulent flow 36
3.5.1 Reynolds equations 36
3.5.2 Turbulent boundary layers 37
3.6 Turbulence modelling 38
3.6.1 The Standard k-ϵ turbulence model 38
3.6.2 Boundary conditions for the k-ϵ model 39

4 A waterjet inlet duct performance model **43**
4.1 Introduction 43
4.2 Geometric modelling 45
4.2.1 Geometric simplifications 45

4.2.2 Parameterization of the s-duct geometry 45
4.2.3 Representation of the parametric geometry 48
4.3 The computational grid 48
4.3.1 O-grids 49
4.3.2 Butterfly grids 49
4.3.3 Grid quality 51
4.3.4 Grid generation 52
4.4 CFD model and experimental validation 52
4.4.1 Flow through s-shaped ducts 53
4.4.2 Experimental configuration 53
4.4.3 Computational simulation 53
4.4.4 Computations compared to experimental data 54
4.4.5 Discussion of the validation 57
4.4.6 Choice of CFD model for waterjet inlet duct flow 58
4.5 Objective functions 65
4.5.1 Cavitation performance 65
4.5.2 Total pressure loss 66
4.6 Summary 69

5 Optimization of the waterjet inlet duct performance model 71
5.1 Introduction 71
5.2 The Problem 71
5.3 Optimization algorithm 72
5.4 CFD and optimization 73
5.4.1 Surrogates from coarse grid computations 73
5.4.2 Termination criterion in Fluent 75
5.4.3 Initial guesses for the CFD analysis 75
5.5 Results 76
5.5.1 Cavitation 76
5.5.2 Loss 77
5.5.3 Optimal designs 83
5.5.4 Hydrodynamics of initial and optimal designs 87
5.6 Discussion of the results 98
5.6.1 Geometry and hydrodynamic performance 98
5.6.2 Relation between the objectives 98
5.6.3 Effects of changing the radius curve 102
5.6.4 Behaviour of the optimization algorithm 104
5.6.5 The surrogate functions 104
5.7 Summary and conclusions 104
5.8 Future research 107

A Positive linear dependence and the regular simplex in optimization 109
A.1 Positive linear dependence and positive bases in $\mathbb{R}^n$ 110
A.2 The regular simplex in $\mathbb{R}^n$ 110
A.2.1 The regular simplex as a minimal positive basis 110
A.2.2 On a question of Powell 113

B NURB curves **117**
B.1 B-spline basis functions 117
B.2 NURB curves 119
B.3 Applications 119
B.3.1 Circular arcs 120
B.3.2 Connecting two curves 120
B.3.3 Interpolation and parameterizations 121

Notation for Chapter 2

Symbols

B	basis matrix
C	generating matrix
cs	exploratory moves algorithm for coordinate search
f	objective function
I	identity matrix
n	number of dimensions
P	pattern matrix
s	step, $\in \mathbb{R}^n$
x	variable, $\in \mathbb{R}^n$
Δ	step length
Γ	matrix whose columns constitute a positive basis for $\mathbb{R}^n$

Subscripts

cs	coordinate search
k	k:th iteration

Notation for Chapter 3

Symbols

C_μ	constant in the Standard k-ϵ equations
D	rate of strain tensor, 3×3 matrix
D_H	hydraulic diameter
I	turbulence intensity
k	turbulent kinetic energy
l	turbulence length scale
L	typical length
$\mathbf{n}$	outward normal
p	static pressure
Re	Reynolds number
t	time variable
$\mathbf{u}$	velocity
u_i	velocity component
u_τ	friction velocity, $= (\tau_w/\rho)^2$
U	velocity component tangential to wall
U	velocity magnitude
U	typical flow speed
U_i	averaged velocity component
V	volume element
V_t	volume element moving with the fluid
$\mathbf{x}$	space variable ($\in \mathbb{R}^3$)
y	distance to wall
δ, δ_{99}	boundary layer thickness
ϵ	rate of dissipation of turbulence energy
κ	von Karman's constant
μ	molecular viscosity

μ_t	turbulent viscosity
ν	kinematic viscosity, $= \mu/\rho$
ρ	density (constant)
σ	stress tensor, 3×3 matrix
τ_w	wall shear stress

Subscripts

i, j	Cartesian coordinate directions

Superscripts

$'$	fluctuating component
$-$	average
$+$	non-dimensionalized by means of ν, τ_w and ρ

Notation for Chapter 4 and 5

Symbols

A	surface area
C_p	static pressure coefficient, $= (p^* - p^*_{ref})/(0.5\rho U^2_{ref})$
$C_{p,min}$	minimum static pressure coefficient
C_P	total pressure loss coefficient, $= (P_{ref} - P)/(0.5\rho U^2_{ref})$
$\bar{C_P}$	mass-averaged total pressure loss coefficient
D	duct flow domain
D	diameter of duct cross section
dS	surface measure
E	kinetic energy
f	objective function
g	gravitational acceleration, $= 9.81$ m/s^2
H	duct height in y-direction
I	turbulence intensity
k	turbulent kinetic energy
l	turbulence length scale
L	duct length in x-direction
m_1, m_2, m_3	mesh parameters
$\dot{m}$	mass flow rate
n	number of dimensions
p	static pressure without hydrostatic pressure
p^*	static pressure including hydrostatic pressure
$\mathbf{p}$	geometry parameter vector
P	total pressure, $= p + 0.5\rho U^2$
r	radius (curve)
r_1, r_2	radius curve parameter
R_1, R_2	lower and upper duct bend radius
ra	mesh parameter
s	center curve arc length parameter
s_i	arc length of the first i sections of center curve
sr_1, sr_2	mesh parameters

U	velocity magnitude
x	direction alinged with sea surface
y	direction normal to sea surface
y^+	non-dimensionalized distance to wall
α	inclination of duct
Δ	step length
ϵ	rate of dissipation of turbulence energy
ϕ_g	gravitational potential
μ	molecular viscosity
Ω	set of parametric vectors fulfilling the constraints
ρ	density

Subscripts

IN	refers to duct inlet opening
OUT	refers to duct outlet
ref	reference

Superscripts

α	surrogate

Chapter 1

Introduction

1.1 Motivation and background

The work presented in this thesis has grown forth from considerations of the following engineering design problem: optimize the shape of a waterjet inlet duct for optimal hydrodynamic performance.

The problem was posed by Rolls-Royce AB in Kristinehamn[1], who manufactures waterjets and other systems for marine propulsion. They have been involved in a project called SEABUS for the design of a high-speed hydrofoil ferry, propelled by a waterjet propulsion unit with a so called ram-type inlet. Rolls-Royce AB was interested in investigating the possibility of designing such an inlet using computational fluid dynamics (CFD) techniques and formal optimization algorithms.

A waterjet propulsion unit with a ram-type inlet is shown in Fig. 1.1. Principally, it consists of an inlet, a pump and a nozzle. Sea water enters the inlet opening and is led by the inlet ducting to the pump. The pump adds energy to the water. The momentum of the water is increased as the flow accelerates through the converging nozzle, and the thrust that propells the ship is produced by the ejection of a jet of water rearward of the ship. Steering can be accomplished by deflection of this jet.

The premises and objectives when solving the problem were as follows. In order to focus the work on the investigation and solution of the problem it was decided that the commercial software Fluent should be used for the flow computations, and the commercial software Gambit for geometry representation and mesh generation. The optimization was to be done for maximum static pressure (minimum risk of cavitation[2]) and for minimum losses. If possible, a compromise between these two objectives should be considered. To simplify matters, only the flow inside the duct was taken into account. Rolls-Royce AB suggested a uniform inlet velocity condition of 8 m/s to be set at the entrance of the duct, corresponding to an "off-cruise" condition when the ship has to negotiate a local maximum in its resistance curve. At such a condition, it is required a high flow-rate in the inlet duct relative to the ship speed, which

[1] Former KAMEWA, Karlstads Mekaniska Werkstad.

[2] The flow is said to cavitate when the static pressure drops below the vapour pressure. This must be avoided since it increases losses and erosion.

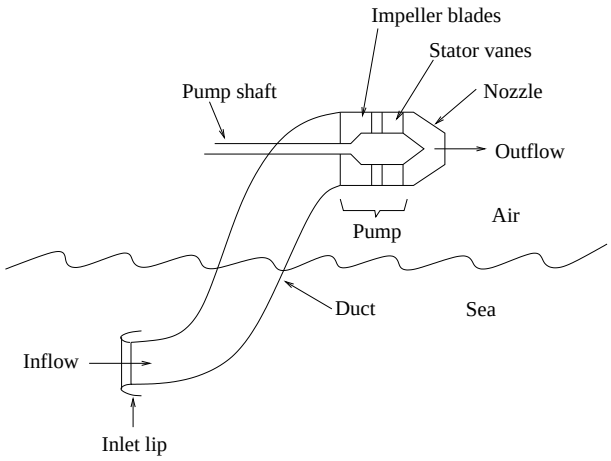


Figure 1.1: A waterjet propulsion unit with ram-type inlet duct.

causes the water in front of the inlet opening to accelerate and contract into the inlet. As a consequence, the pressure is decreased and the risk of cavitation increased.

The thesis has been done as the second part of a two-stage project. The first part was done by Sara Ågren [40]. In her master thesis, she considered, among other things, the geometric aspects of the problem, as well as the possibility to speed up the flow calculations in an optimization process by initializing each calculation with a mapping of a previously obtained solution. We have used her code, written in C++, for the automatic generation of computational grids in Gambit.

We note that our design problem deals with the geometric shape of the inlet ducting, which extends from the inlet opening to the entrance of the pump. This constitutes just a part of the water propulsion system design, which in turn is an integrated part of the total ship design. Moreover, we have restricted our attention to *internal* aspects of the duct shape, ignoring important *external* aspects such as cavitation and losses associated with the flow entering the duct, as well as external drag when the duct ploughs through the sea. It is thus clear that the results of optimizing the inlet ducting itself can not be directly applicable in reality. However, the optimization can give valuable information about the relation between the inlet hydrodynamic performance and its geometry.

1.2 Design optimization

Computer simulations are increasingly important and popular tools for solving engineering design problems such as the one considered in this thesis. Whereas earlier computer resources were used for the analysis of a *single* design, the accelerating development of hardware and software has made it possible to analyze a large number of designs in order to choose the best one. This possibility leads to an optimization problem in which the objective function, because of the inherent computer simulation, is computationally expensive to evaluate, has an

almost completely unknown structure and gives no explicit information about derivatives.

Such problems can be solved by *direct search methods* which are optimization methods that do not make explicit use of derivatives but proceed simply by evaluating the objective function itself. Examples of direct search methods include such classical algorithms as the Nelder-Mead algorithm [29], the simplex method [16] and the Hooke and Jeeves algorithm [19].

A useful concept for solving these kind of problems is *surrogate functions*, which are approximations of the objective function that are less time-consuming to evaluate. Engineering practise has been to evaluate the function at scattered points in the variable space and use these samples together with algebraic interpolation or approximation techniques to construct the surrogate function. The optimization is then performed on the surrogate, which in this case is comparatively much less expensive and also provides explicit derivatives. The main question has been what to do if the optimum of the surrogate function turns out to be an unsatisfactory design when evaluated with the actual objective function.

Lately there has been much research done on direct search methods as well as on how to handle surrogate functions in the optimization process. In [36], Virginia Torczon developed a general pattern search method which included many of the classical direct search methods. Dennis and Torczon [14] combined the general pattern search method with ideas from trust-region methods to suggest a framework for handling surrogate functions. There is no explicit discussion of the choice of surrogate function in [14] but it is indicated that such a function could depend on the mesh size in a pde code, an idea that will be investigated in this report. Other frameworks based on pattern search are found in [38, 5]. The objective function is regarded as the outcome of a stochastic process which enables the construction of algebraic approximation models from samples using statistical tools. In [5], some of the function evaluations are done in order to update the model, a procedure referred to as a *balanced search*. A different approach, not based on pattern search, is found in [8], where quadratic models are used in a trust-region framework. A balanced search is done and the search for points to update the model are governed by considering strict geometric properties of the set of model-building points. In [2], a framework is given for handling surrogates that are subject to certain mild analytic conditions.

Seil [34] considered a problem very similar to ours. Seil successfully used CFD and the Sherif-Boice algorithm (SBA) [35] to investigate and optimize a waterjet inlet of flush-type. He found that the SBA produced a marked improvement in the objective function value within only a few iterations, and could thus conclude that the algorithm works well on this kind of problem provided that the number of variables is low (5-10). The problem-solving part of our work has to a large extent been guided by [34].

The articles [5] and [9] present numerical results from applying different optimization algorithms to a helicopter test problem with 31 design variables. Among the best methods was a pattern-search-based method that uses surrogates constructed by interpolating samples of the objective function. It gave considerably improved designs after about $100 - 200$ function evaluations. The worst result was obtained with a genetic algorithm tested in [5] that, even though it used many more evaluations, gave a more moderate decrease in the objective.

Dadone and Grossman [11] present a very interesting surrogate approach,

which they term *progressive optimization*, for the solution of compressible fluid flow design problems. They use premature termination of the flow solver as well as coarse grids to simplify the flow computations. They have developed an ad hoc method for how to progressively increase the accuracy of the flow computations as the optimization proceeds by successively increasing the number of iterations in the flow solver during each flow calculation, and using progressively refined grids. The method was found to be very efficient and robust. The drawback is the lack of generality. The main difference between their problem and ours is that they could compute the gradient of the objective function by solving an adjoint problem. Hence, they used a gradient-based method. However, it should be possible to generalize their idea of a continuous interaction between the optimization process and the underlying physical simulation, and their tools for constructing the surrogates.

In this work we will not apply methods that make use of surrogates constructed from samples of the objective. We considered it more exciting and interesting to investigate the structure of this particular problem by constructing surrogates by manipulations of the turbulent flow model underlying the objective functions.

1.3 Outline of the report

The rest of the report consists of four chapters and two appendices.

In Chapter 2 we discuss a general direct search method called the *positive basis pattern search algorithm.* It has been developed by Torczon [36], and Lewis and Torczon [22]. We define the algorithm and discuss some of its properties. We state the general convergence theorem and establish the necessity of the objective function to be continuously differentiable by means of a counterexample. We give some examples of pattern search methods and show that the Sherif-Boice algorithm [35] (slightly modified) is an instance of the general pattern search algorithm. We also discuss the role of the positive basis that provides the search directions in pattern search, and, finally, the concept of surrogates and how it can be used in the framework of pattern search.

Chapter 3 gives a brief introduction to the theory of fluid dynamics. We derive the Navier-Stokes equations and discuss the modelling of turbulence and turbulent boundary layers.

In Chapter 4 we present and discuss an implemented model of the waterjet inlet duct. In order to prepare for the optimization of the inlet duct, we parameterize the inlet geometry so that it can be varied. We present the CFD model used for the flow computations, and validate it against experimental data. Finally, we discuss the two criteria of the inlet hydrodynamic performance that have been used.

In Chapter 5 we optimize the model presented in the previous chapter. We discuss how the optimization and the CFD model are, and can be, related to each other. We investigate the possibility of constructing surrogate functions from coarse grid flow simulations. Numerical optimization results are presented and analyzed. Finally, we summarize with conclusions and suggestions for future research.

Appendix A has been written quite separately from the rest of the report. Inspired by a few questions arising in connection to optimization we apply some

theory of positive linear dependence to regular simplices and minimal positive bases. We solve a problem that concerns the approximation ability of the minimal positive basis consisting of a regular simplex. We also construct a bounded infinite sequence of distinct regular simplices of constant size, thereby answering a question of Powell in [32]. Moreover, it shows that the iterates generated by the classical simplex method [16] do not have the same favourable algebraic structure as those generated by pattern search methods.

Appendix B, finally, contains some basic theory of non-rational uniform B-spline curves (NURBs), followed by three example applications. In general, NURBs provide an excellent tool for the representation and local control of geometric shapes. They have been used in this thesis to represent the piecewise defined curves that define the duct geometry.

Chapter 2

Optimization without calculating derivatives

2.1 Introduction

In this chapter we consider the unconstrained finite-dimensional optimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{2.1}$$

when f has the following properties:

i) the derivatives of f, even if they do exist, are not available,

ii) f is time-consuming to evaluate,

iii) the idealized function f can only be computed to low accuracy, and is contaminated with high-frequency, low-amplitude distortions.

Such objective functions are typical in the kind of engineering design problem considered in this work, where f is based on the outcome of a complicated computer simulation. Property iii) is caused by small variations and numerical errors in this simulation.

Most optimization algorithms make use of derivative information, but such methods are not suitable for this problem because of i). Of course, we still have the possibility of computing some numerical approximation to the gradient, but this is not a good idea because of iii): if δ is the error in f and h the step length, the error in such an approximation is proportional to δ/h. Errors in f are blown up in the gradient which may cause gradient methods, relying only on the gradient search direction, to deteriorate. On the other hand, even if the effect of iii) is small we still have an argument against gradient methods: since they use only one search direction (the negative of the gradient in the case of minimization) at each iteration, the optimization may easily get stuck in local minima.

Instead, problem 2.1 demands for *direct search*, or *derivative-free*, methods that proceed without making explicit use of the gradient.

In [25], a distinction is made between these two terms. The term "direct search" is reserved for methods that neither require nor estimate derivatives, but proceed just by considering the ranking of different objective function values, and not quantitative differences. Such a method may accept every new iterate that leads to a decrease in the objective function value. Consequently, they are often easy to understand and to implement. Examples include such classical algorithms as the Nelder-Mead algorithm [29], the simplex method by Hext, Himsworth and Spendley[16], the Hooke and Jeeves algorithm [19] etc. A derivative-free method, on the other hand, may still use some kind of approximation of the function in order to estimate derivatives, as in the classical response surface method by Box [7]. Thus any direct search method is also a derivative-free method. Since approximation functions can be used in conjunction with true direct search methods, we will not keep the strict distinction between the two terms.

In the following section we present a brief overview of direct search methods. The rest of the chapter focuses on a class of direct methods called pattern search methods. In Section 2.3 we define the general pattern search algorithm and discuss some properties of it. Some well-known examples that are instances of the general algorithm are given in Section 2.4. In Section 2.5, we discuss the positive basis that provides the search directions. We investigate how the size of the basis may affect the efficiency of the search. Finally, we discuss the concept of surrogates and how to use them in conjunction with the pattern search method.

2.2 Brief overview of available methods

Following Powell [32], we may distinguish at least six different kinds of direct search (or derivative-free) methods: approximation methods, simplex methods, random methods, discrete grid or pattern search methods, line search methods and conjugate direction methods. We now briefly discuss the first four of them.

Approximation methods use algebraic approximations of the objective function. The approximations may, for example, be linear as in [31] or quadratic as in [8]. The purpose of using algebraic approximations is to make careful and systematic use of all available information about f, which consists of the values of f at previously visited points. Thus these methods seem very well suited for our problem. A drawback, however, is the lack of a general convergence theory.

Among the simplex methods we find contributions from the early developers of the field, such as the classical simplex method [16] and the Nelder and Mead algorithm. These methods often use only rank order information (and not quantitative differences in function values at different points) to find the next iterate and could therefore be suspected to be less efficient than the approximation methods just mentioned. The convergence properties are either not well known or known not to be desirable as is the case for the Nelder-Mead algorithm. McKinnon [28] showed how the Nelder-Mead algorithm converges to a nonstationary point in a case when the objective is convex and continuously differentiable.

Simulated annealing and genetic algorithms are methods that introduce a random element in the optimization procedure. They are often easy to implement, do not get stuck in local minima and may sometimes be shown to have

a kind of convergence to a global minimum. It clearly seems, though, that one may have to pay for this in the number of function evaluations, see for example [6]. Intuitively, it is not appealing to evaluate randomly chosen points if the objective function is very expensive.

In discrete grid or pattern search methods the search is done on a fixed grid of points, such as a lattice. Reducing the step length means proceeding the search on a scaled version of the original lattice. Powell [32] explains how the restriction of the search to a discrete grid may lead to an inefficiency in the optimization. This inefficiency depends on how the optimum is related to the lattice, in combination with other properties of the objective function under consideration. Apart from this, pattern search methods have many desirable properties. Because they investigate more than one search direction in each iteration (in contrast to gradient methods), they are not as likely to get stuck in local minima. They include the classical coordinate search and the algorithm by Hooke and Jeeves [19], are easy to understand and implement and have recently been given a general form by Torczon in [36], where she also presents a general convergence theory.

Torczon's general formulation of pattern search methods gives great flexibility in the design of specific algorithms. This can be used to incorporate approximation functions, *surrogates*, into the optimization process. Surrogates built from interpolation or approximation of samples of the objective function has been used for a long time for solving engineering problems [4]. The problem has been the lack of a general strategy for a continuous interaction between the surrogate and the "real" function as the optimization proceeds. General pattern-search-based frameworks for handling surrogate optimization have been proposed in [5, 14, 38].

We consider this to be enough motivation for a more detailed presentation and study of pattern search algorithms.

2.3 The general positive basis pattern search algorithm

In the following we present a simplified version of the *general positive basis pattern search algorithm* given in [22]. It is a class of direct search algorithms developed for the unconstrained problem (2.1). In [24] they are extended to linearly constrained problems. The positive basis pattern search algorithms are natural extensions of those in [36] because the search directions in the so called pattern matrix do not have to consist of an entire basis $\{\mathbf{v}_i\}_{i=1}^n$ for $\mathbb{R}^n$ together with the opposites $\{-\mathbf{v}_i\}_{i=1}^n$, but just a *positive basis* (see Appendix A). It leads to a possible decrease in the maximal number of function evaluations per iteration, which might be desirable if f is very costly to evaluate (property ii) mentioned in the introduction to this chapter).

2.3.1 Definition of the general algorithm

The main ingredients in the general pattern search algorithm are a *pattern matrix* $P_k \in \mathbb{R}^{n \times p_k}$ (the index k indicates that it may depend on the iteration), whose columns constitute the possible search directions, an *exploratory moves algorithm* that suggests a step and an algorithm for *updating* the pattern and

the step length. Given a step length Δ_k, a step s_k is defined as any column of $\Delta_k P_k$. To indicate that a vector s is a column of a matrix P, we use the notation $s \in P$. The general pattern search algorithm can be defined as:

Algorithm 1. *The general positive pattern search algorithm for unconstrained optimization.*
Given $x_0 \in \mathbb{R}^n$ *and* Δ_0.
For $k = 0, 1, \cdots$ *do*

1. *Compute* $f(x_k)$.
2. *Use the exploratory moves algorithm to determine a step* $s_k \in \Delta_k P_k$.
3. *If* $f(x_k + s_k) < f(x_k)$, *let* $x_{k+1} = x_k + s_k$. *Otherwise, let* $x_{k+1} = x_k$.
4. *Update the pattern* P_k *and the step length* Δ_k.

To construct a specific pattern search algorithm, one has to define

1. a specific pattern P_k,
2. an exploratory moves algorithm,
3. and how to update the pattern and the step length.

We now give sufficient conditions for how this should be done in order to guarantee convergence according to Theorem 2.

Conditions on the pattern. The pattern P_k is the matrix product of a *basis matrix* B and a *generating matrix* C_k. We require $B \in \mathbb{R}^{n \times n}$ to be invertible and $C_k \in \mathbb{Z}^{n \times p_k}$, $p_k > n + 1$. The columns of C_k are partitioned as

$$C_k = [\Gamma \; L_k \; 0],$$

where Γ is required to be a positive basis for $\mathbb{R}^n$ and 0 means a column of zeros. We call $B\Gamma$ the *core pattern*.

Conditions on the exploratory moves algorithm. $s_k \in \Delta_k P_k$ and if there exists a core pattern step $y \in \Delta_k B\Gamma$ such that $f(x_k + y) < f(x_k)$ then $f(x_k + s_k) < f(x_k)$. This can also be expressed as follows: before giving up the search for a better point, the algorithm must try all steps in the core pattern.

Conditions on the updating algorithms. There are no other conditions for how the pattern should be updated than those already given for the pattern itself. Concerning the step length, we either leave it unchanged or half it. In [22] the step length is allowed to vary more freely. The condition for reducing the step length is "no decrease in the objective function during the iteration". From the condition on the exploratory moves algorithm, it follows that the step length can only be reduced when all the steps in the core pattern has been checked.

Algorithm 2. *The algorithm for updating the step length.*
If $f(x_k + s_k) \geq f(x_k)$ *then* $\Delta_{k+1} = \frac{1}{2}\Delta_k$. *Otherwise,* $\Delta_{k+1} = \Delta_k$.

2.3.2 Some properties of the general algorithm

The general pattern search algorithm is a *gradient-related adaptive grid method* which only requires *simple decrease* in the objective function. It also has nice convergence properties. In the following, we denote by $\{x_k\}_{k=0}^{\infty}$ a sequence of iterates generated by Algorithm 1 applied to problem (2.1).

The simple decrease condition means that a step s_k is accepted if and only if $f(x_k + s_k) < f(x_k)$ (see 3. in Algorithm 1). It should be considered in contrast to *sufficient* decrease, which is usually required for the convergence analysis of gradient-related methods.

The condition $s_k \in \Delta_k P_k$ on the exploratory moves algorithm ensures ([36]) that the iterates x_k lay on differently scaled versions of the lattice (or grid) generated by the columns of B and translated by x_0. The scaled lattices are nested in the sense that finer lattices contain coarser ones. More precisely, the search for x_{k+1} is done on the lattice

$$\mathbb{L}_k = \{x_k + \sum_{i=1}^{n} z_i \Delta_k b^i; z_i \in \mathbb{Z}\},$$

where b^i denotes the columns of the basis matrix B. $\mathbb{L}_k$ is generated by $\Delta_k B$ and translated by x_k. Since the step length is either halved or unchanged, we have $\mathbb{L}_k \subseteq \mathbb{L}_{k+1}$. In this sense pattern search methods are adaptive grid methods. Let us consider the following example in $\mathbb{R}^2$:

$$B = I \text{ (the identity matrix)}, \quad C_1 = [I \ -I \ 0], \quad C_2 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \end{bmatrix}.$$

The corresponding patterns are illustrated in Fig. 2.1. It can be seen how the basis matrix and the generating matrix interacts to define the possible steps in the iteration. $\Delta_k B$ generates the lattice and C determines the possible steps (indicated by arrows) to take on that lattice.

We note that the only condition on the part L_k of the generating matrix is to have integer entries. We have been quite restrictive with the possible steps from the current iterate in our example. However, L_k might have arbitrarily many columns. For example, if its columns are all vectors in $\mathbb{Z}^n$, then any step on the lattice generated by $\Delta_k B$ will be included in the pattern and the next iterate can be any point on $\mathbb{L}_k$.

The adaptive grid property of general pattern search methods is used in the convergence analysis to ensure that the step length tends to zero.

The pattern search method is *gradient-related* in the sense that there always exists a search direction that captures a certain part of the direction of steepest descent. The reason is that Γ is a positive basis and therefore guarantees the existence of an upper bound on the angle between the direction of steepest descent and the best search direction. This is the content of the following proposition, the proof of which can be found in [22].

Proposition 1. *For a positive basis pattern search method in $\mathbb{R}^n$ there exists a constant $c > 0$, depending only on Γ, such that for any k, there is a core pattern step $s_k \in \Delta_k B\Gamma$ for which*

$$\frac{-\nabla f(x_k) \bullet s_k}{\|\nabla f(x_k)\| \|s_k\|} \geq c \frac{1}{n\sqrt{n}}. \tag{2.2}$$

The bound in this theorem indicates a loss of efficiency for the pattern search algorithms with increasing dimension. If $\Gamma = [I \ -I]$, we have from [36] that the sharp bound for (2.2) is $1/\sqrt{n}$. In Theorem 10 in Chapter A we show that when Γ is the minimal positive basis consisting of a regular simplex, the sharp bound is $1/n$.

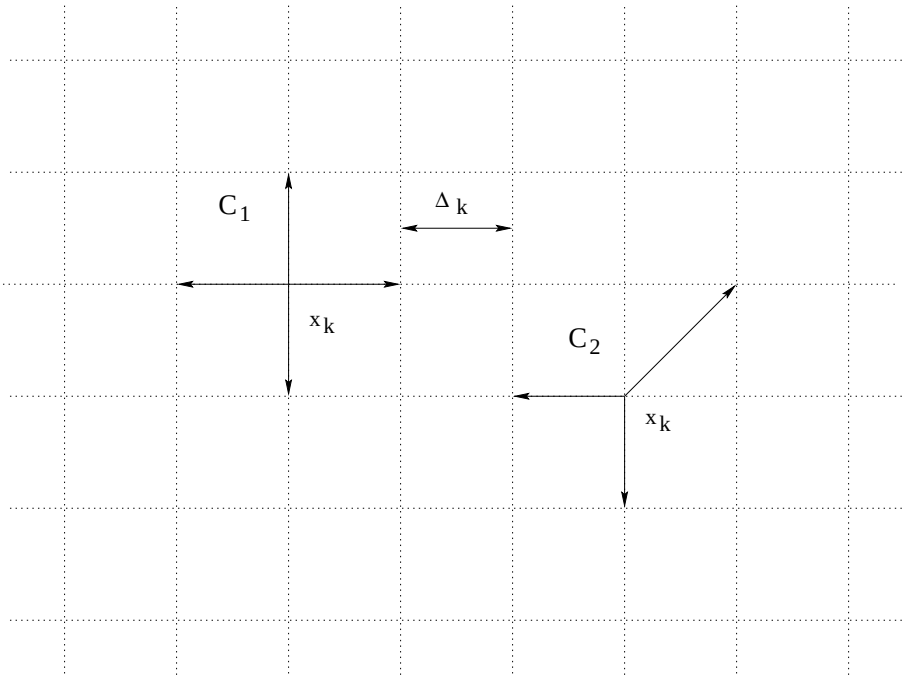


Figure 2.1: The dashed lines represent the lattice generated by $\Delta_k B$ when $B = I$. The generating matrix C determines the possible steps from the current iterate x_k.

The positive basis pattern search algorithm has nice convergence properties, as stated by the following theorem taken from [22].

Theorem 2. *Suppose that* $L(x_0) := \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ *is compact and that* $f \in C^1(\Omega)$ *for some open set* $\Omega \supset L(x_0)$. *Then*

$$\liminf_{k \to \infty} \|\nabla f(x_k)\| = 0.$$

Under stronger conditions (see [22]) on the general algorithm the lim inf in this theorem can be replaced by lim. The stated result means that the pattern search algorithm behaves nicely. For example, there always exists a subsequence of the sequence of iterates $\{x_k\}_{k=0}^{\infty}$ that converges to a stationary point.

To see the necessity of the condition on f to be continuously differentiable, we consider the function

$$f(x) = \|x\|^2 + \sum_{i=1}^{n-1} |x_i - x_{i+1}| + \sum_{i=1}^{n} x_i \, .$$

We get

$$f(x) \geq \|x\|^2 - n\|x\| \geq -\frac{n^2}{4}.$$

By the first inequality, f tends to infinity with $\|x\|$ and by the second f has a lower bound. Since f is also continuous, $L(x_0)$ is compact. Moreover, f is

Lipschitz continuous over any ball with radius R since

$$\begin{aligned}
|f(x)-f(y)| =& \left|(\|x\|^2-\|y\|^2)+2\sum_{i=1}^{n-1}(|x_i-x_{i+1}|-|y_i-y_{i+1}|)+\sum_{i=1}^{n}(x_i-y_i)\right| \\
\leq& \big|\|x\|-\|y\|\big|(\|x\|+\|y\|)+ \\
&+2\sum_{i=1}^{n-1}|x_i-x_{i+1}-y_i+y_{i+1}|+\sum_{i=1}^{n}|x_i-y_i| \\
\leq& 2R\|x-y\|+5\sum_{i=1}^{n}|x_i-y_i| \\
\leq& (2R+5n)\|x-y\|
\end{aligned}$$

Now, let e^i be the unit vectors, i.e, $e^i_j=\delta_{ij}$ where δ_{ij} is the Kronecker delta function. We have $f(0)=0$ and

$$f(he^i)=\begin{cases} h^2+2|h|+h\geq|h|, & \text{if } i=1,n \\ h^2+4|h|+h\geq 3|h|, & \text{if } i=2,\cdots,n-1 \end{cases}$$

so that $f(he^i)>0=f(0),\forall h\neq 0, i=1,\cdots n$. Hence a search along the coordinate directions from $x=0$ will not result in a lower function value. Note also that $(-1,-1,\ldots,-1)$ is a direction of descent since

$$f(-h,\cdots,-h)=nh^2-nh=nh(h-1)<0,\quad 0<h<1.$$

This means that if for example simple coordinate search, which in [36] is shown to be an instance of the general pattern search algorithm, starts off from the origin it will stay there in spite the existence of a descent direction.

2.3.3 Bound constraints

The shape optimization problem motivating this thesis is, by its nature, constrained (see Section 5.2). We therefore note the extension made in [23] of pattern search algorithms to bound constrained problems,

$$\min_{l\leq x\leq u} f(x), \tag{2.3}$$

where f is real-valued as before, $l,u,x\in\mathbb{R}^n$ and $l<u$. (The vector inequalities are to be understood coordinate-wise). Denote by B the bound constrained domain defined by l and u and let P be the projection onto B. Define

$$q(x)=P(x-\nabla f(x))-x.$$

This is the appropriate "gradient" to consider in the case of constrained problems since $q(x)=0$ if and only if x is a constrained stationary point for (2.3), see [23]. In order to make sure that the pattern always contains search directions along the boundary of B, the core pattern should contain a nonsingular diagonal matrix and its negative. This is the only additional restriction required to obtain the same convergence result as in Theorem 2 but with ∇f replaced

by q and $L(x_0)$ replaced by $L_B(x_0) = \{x \in B : f(x) \leq f(x_0)\}$. They also note that the extended pattern search gives exactly the same sequence of iterates as does the unconstrained pattern search presented above applied to the function

$$F(x) = \begin{cases} f(x), & \text{if } x \in B, \\ \infty, & \text{otherwise.} \end{cases}$$

Hence, under the restriction that the core pattern contains a diagonal matrix and its negative, the class of pattern search methods presented in this chapter can be applied to the problem (2.3).

2.4 Examples of pattern search algorithms

We now give examples of two instances of the general pattern search algorithm. The first is coordinate search with fixed step length and the second is a variant of the Hooke and Jeeves algorithm [19], called the Sherif-Boice algorithm [35], or SBA.

2.4.1 Coordinate search with fixed step length

This algorithm searches through each coordinate direction in turn. Let e_i denote the standard unit vectors in $\mathbb{R}^n$. The basis matrix is the identity, $B = I$. The generating matrix C_{cs} is fixed and has 3^n columns that contains all possible combinations of $\{-1, 0, 1\}$. For $n = 2$ we have

$$C_{cs} = \begin{bmatrix} 1 & 0 & -1 & 0 & 1 & 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 1 & -1 & -1 & 1 & 0 \end{bmatrix}.$$

We note that C_{cs} contains a positive basis, namely $\Gamma_{cs} = [I \quad -I]$. Thus, $B\Gamma_{cs} = \Gamma_{cs}$ is a core pattern for coordinate search.

Algorithm 3. *The exploratory moves algorithm for coordinate search.*
Given x_k, Δ_k *and* $f(x_k)$.
Set $s_k = 0$ *and* $min = f(x_k)$.
For $i = 1, \cdots, n$ *do*
  $s_k^i = s_k + \Delta_k e_i$
  If $f(x_k + s_k^i) < min$
    $min = f(x_k + s_k^i)$
    $s_k = s_k^i$
  else
    $s_k^i = s_k - \Delta_k e_i$
    If $f(x_k + s_k^i) < min$
      $min = f(x_k + s_k^i)$
      $s_k = s_k^i$
Return $s_k, f(x_k + s_k)$.

We denote this algorithm by $(s_k, f(x_k + s_k)) = cs(x_k, f(x_k), \Delta_k)$. Unless a successful step s_k is found, the algorithm examines all $2n$ core pattern steps $y \in \Delta_k B \Gamma_{cs}$. We may conclude that coordinate search with fixed step length is an instance of the general positive basis pattern search algorithm.

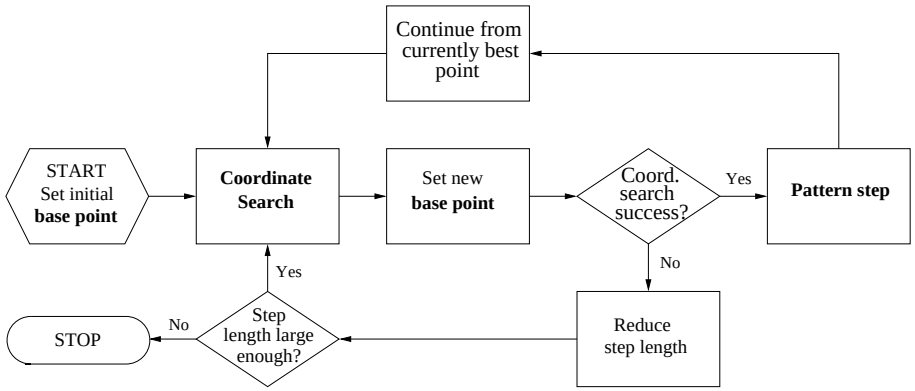


Figure 2.2: Flowchart of the SBA.

2.4.2 The SBA

The SBA [35] is a variant of the Hooke and Jeeves algorithm [18]. It alternately performs coordinate searches and *pattern steps*, which are attempts to further investigate promising directions built up by a preceding coordinate search. Also, some of the currently best points are set as *base points*. We say that a search or step *succeeds* if it leads to a lower function value and otherwise that it *fails*. As we will show, the SBA is an instance of the general pattern search algorithm. It is also a true direct search method (see the discussion in the introduction to this chapter): it uses only rank information about the objective function.

First, the initial point is evaluated and set as a base point. Then a coordinate search is performed and the resulting best point is set as a base point. If the coordinate search succeeds, a pattern step is performed. The pattern step has the same direction and magnitude as the step between the current and the previous base points. Then the algorithm starts all over again from the currently best point, i.e., a coordinate search is performed from the result of the pattern step if it was successful and otherwise from the result of the previous performed coordinate search. Every time a coordinate search fails, the step length is halved, and every time it succeeds a new base point is marked. See the flowchart in Figure 2.2.

The SBA is less opportunistic than the Hooke and Jeeves algorithm. While the latter always performs a coordinate search from the result of a pattern step, the former only does it *if the pattern step was succesful*.

The formulation above differs from the original one in [35] because it reduces the step length also when a failed coordinate search follows after a successful pattern step. Otherwise it could happen that we search through the same points once more when we start the next iteration with a coordinate search.

We now formulate the SBA as an instance of the general algorithm. The basis matrix is the identity matrix, $B = I$. In coordinate search, the generating matrix, which we denote by C_{cs}, was fixed. For the SBA, we allow the generating matrix C_k to change from iteration to iteration in order to capture the effect of the pattern step. If the pattern step fails in the SBA, the result of one iteration is to perform a coordinate search step s_k^c. Hence, C_{cs} must be included in

C_k. On the other hand, if the pattern step s_k^p succeeds, an examination of the flowchart in Fig. 2.2 reveals that it then is the sum of s_k^p and the pattern step from the previous iteration s_{k-1}^p (which possibly is zero), i.e. $s_k^p = s_k^c + s_{k-1}^p$. The step produced by the iteration then becomes

$$s_k = s_k^c + s_k^p = 2s_k^c + s_{k-1}^p. \tag{2.4}$$

Now, denote the columns of C_k by c_k^i and its last column by c_k^p. We let c_k^p represent the previous pattern step, i.e. $\Delta_k c_k^p = s_{k-1}^p$, and define $c_0^p = 0$. It is then clear from Eq. (2.4) and the preceding discussion that

$$C_k = [C_{cs} \quad 2C_{cs} + c_k^p],$$

where the addition is to be understood columnwise. Hence, C_k consists of $2 * 3^n$ columns.

Only the last 3^n columns of C_k are updated from iteration to iteration. This is done with the following algorithm. (Recall that s_k is the entire step produced by the iteration and s_k^c the step of the coordinate search performed during the iteration.)

Algorithm 4. *Updating of the generating matrix for the SBA.*
For $i = 3^n + 1, \ldots, 2 \cdot 3^n$
  $c_{k+1}^i = c_k^i - c_k^p + (s_k - s_k^c)/\Delta_k$
Return.

Since s_k/Δ_k and s_k^c/Δ_k are columns of C_k, and $C_0 \in \mathbb{Z}^{n \times 2 \cdot 3^n}$, an argument by induction shows that the columns of C_k consist of integers. Moreover, it contains the matrix C_{cs} and therefore the same positive basis and the same core pattern as coordinate search. We conclude that C_k fulfils the conditions for being a generating matrix.

The exploratory moves for the SBA is given below. Note that the notation and logic differ from the original SBA and, in particular, from the Hooke and Jeeves algorithm. In the latter algorithm, the pattern step is defined as the difference between the current iterate and the previous iterate and hence only contains information from the previous iteration. In our formulation, the pattern step contains information both from the previous iteration (the previous pattern step) and the current (the result of the coordinate search). We have used *cs* to denote Algorithm 3.

Algorithm 5. *Exploratory moves for the SBA.*
Given $x_k, f(x_k), \Delta_k$.
Set $s_k = 0, s_k^c = 0, s_{k-1}^p = \Delta_k c_k^p$.
$s_k^c = cs(x_k, f(x_k), \Delta_k)$
If $f(x_k + s_k^c) < f(x_k)$
  $s_k = s_k^c$
  $s_k^p = s_{k-1}^p + s_k^c$
  $x^p = x_k + s_k^c + s_k^p$
  If $f(x_p) < f(x_k + s_k^c)$
    $s_k = s_k^c + s_k^p$
Return s_k, s_k^c.

Unless Algorithm 5 produces a successful step s_k, a failed coordinate search has been performed. Hence, all core pattern steps have been examined. We conclude that the SBA, too, is an instance of the general positive basis algorithm.

2.5 Minimal and maximal positive bases in pattern search

In this section we will see how the general pattern search algorithm defined above can be used to produce new algorithms by replacing a maximal positive basis with a minimal. Since a smaller positive basis often gives a worse approximation of the gradient we are led to the natural question of whether we should have a maximal or a minimal positive basis. We analyze this in a simple case and find, perhaps contrary to intuition, a maximal positive basis to be preferable.

2.5.1 A minimal positive basis search algorithm

In the coordinate search algorithm (Algorithm 3), a step is built up by considering, if necessary, $2n$ directions: each coordinate direction and its opposite, which form a maximal positive basis. However, as soon as a better point is found in some direction, that direction is not further considered, and so a step is built up from at most n linearly independent "small steps". These remarks will now be used to construct an algorithm that mimic coordinate search but only use a minimal positive basis pattern matrix.

Let $\Gamma \in \mathbb{Z}^{n\times(n+1)}$ be a minimal positive basis for $\mathbb{R}^n$. Then any n of the vectors in Γ form a basis for $\mathbb{R}^n$ (Theorem 3 in Chapter A). We may therefore modify the coordinate search to be a search along directions that form a minimal positive basis in the following way ($\Gamma(i)$ denotes the i:th column of Γ):

Algorithm 6. *Exploratory moves for a minimal positive basis search algorithm.*
Given $x_k, f(x_k), \Delta_k$.
Set $s_k = 0, min = f(x_k), i = 0, j = 0$.
While $i \leq n+1$ *and* $j < n$
  $s_k^i = s_k + \Delta_k \Gamma(i)$
  If $f(x_k + s_k^i) < min$
    $min = f(x_k + s_k^i)$
    $s_k = s_k^i$
    $j = j + 1$.
  $i = i + 1$
Return $s_k, f(x_k + s_k)$.

In this way we reduce the highest number of evaluations per iteration from $2n$ to $n+1$. If we replace the ordinary coordinate search with the new one we get variants of Hooke and Jeeves and SBA that use a minimal positive basis core pattern.

2.5.2 How large should the positive basis be?

Often the maximal number of evaluations per iteration with a pattern search method is the number of vectors in the positive basis, or a multiple of it. The argument for using a minimal positive basis is then that it reduces the highest number of evaluations per iteration. On the other hand, a maximal positive basis better approximates the gradient. The question is if the advantages with a minimal positive basis outweighs that of a maximal. To investigate this, we

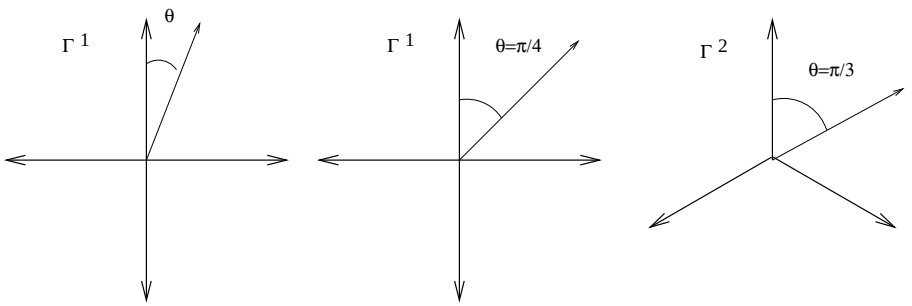


Figure 2.3: Illustration of how a positive basis Γ approximates an arbitrary vector. The basis to the left is maximal and the basis on the right is minimal. θ is the angle between some vector and the closest vector in the basis. In the middle and to the left is seen a vector that is furthest possible from any vector in the basis.

consider the following simple optimization algorithm. Given a positive basis Γ and a current iterate x_k we compute

$$s_k = \arg \min_{y \in \Delta_k \Gamma} f(x_k + y) \tag{2.5}$$

and if $f(x_k + s_k) < f(x_k)$ we accept the step, otherwise we reduce the step length. Whether one should use a maximal or minimal basis in this algorithm is to ask whether it is worth the effort to check many points around the current iterate before deciding on a step, or if it is better to take a step as soon as possible (with preserved convergence properties).

In order to determine how well we can approximate an arbitrary vector x we are interested in the smallest angle between x and any of the vectors in the positive basis, i.e. the angle defined by

$$\cos(\theta) := \max_i \frac{\Gamma(i) \bullet x}{\|\Gamma(i)\| \|x\|}.$$

This angle can be seen to the left in Figure 2.3.

The ability of Γ to approximate the direction of steepest descent may then be measured by

$$d(\Gamma) := \min_{\|x\|=1} \max_i \frac{\Gamma(i) \bullet x}{\|\Gamma(i)\|},$$

which is the cosine of the angle between the vector x that is furthers away from any vector in Γ, and the vector in Γ that is closest to x.

Let Γ^1 be the maximal positive basis that consists of the standard unit vectors and their negatives and let Γ^2 be a minimal positive basis consisting of a normalized regular simplex (Appendix, Definition 1 and Theorem 7). From [36] we have $d(\Gamma^1) = 1/\sqrt{n}$ and Theorem 10 in the Appendix states that $d(\Gamma^2) = 1/n$. For $n = 2$, the situation is depicted in Figure 2.3; $d(\Gamma^1)$ is just the cosine of $\pi/4$ and $d(\Gamma^2)$ the cosine of $\pi/3$.

Obviously $\Gamma^1 \in \mathbb{Z}^{n \times 2n}$ and therefore passes as a core pattern matrix in Algorithm 1. It is not true that $\Gamma^2 \in \mathbb{Z}^{n \times 2n}$, but we use $d(\Gamma^2)$ as an upper bound for $d(\Gamma)$ for any minimal positive basis Γ.

We now suppose that the reduction in the objective function is proportional to the approximation of the gradient and that the gradient is of a fixed size (as is the case if f is linear). A step s according to (2.5) then leads to a decrease in the objective function value bigger than or equal to

$$\delta f = \mu \cdot \|s\| \cdot d(\Gamma^i),$$

where μ is some positive constant. The function f is evaluated $2n$ times per iteration with Γ^1 and $n+1$ times with Γ^2. Hence the quotient of the reduction in f per evaluation using Γ^2 and the reduction in f per evaluation using Γ^1 becomes $2n\sqrt{n}/(n(n+1))$. This quotient is less than 1 for all $n > 1$ and tends to zero as $1/\sqrt{n}$ when n tends to infinity. Hence, a maximal positive basis should pay off in any dimension and become even more effective with increasing dimension. The reason is that the basis ability to approximate the gradient (as measured by $d(\Gamma)$) does not vary linearly with the size of the basis. However, the analysis is made under substantial simplifications.

2.6 Pattern search and surrogate functions

In this section we consider the concept of a *surrogate function* $\tilde{f}$ which approximates f and is less expensive to evaluate. We discuss, inspired by [2, 5, 14, 38], how surrogates can be used in a systematic way in order to optimize f.

We can distinguish at least two types of surrogate functions in the literature. Surrogates of the first type are algebraic constructions that can be expressed in a closed mathematical form. For example, they can be linear or quadratic functions that interpolate or approximate a set of samples of f [8, 31]. This approach is problem-independent in the sense that it treats f as a "black box". They can also be based on a truncated Taylor series expansion of f, as in the classical trust-region method.

Surrogates of the second type are of relevance when the evaluation of f relies on some complicated simulation, as discussed in the introduction to this chapter. The surrogate is based on a simplification of the underlying simulation; we could say that one looks inside the "black box". Consequently, this approach is more problem-dependent. For example, if the evaluation of f involves the numerical solution of a system of partial differential equations (PDE), the surrogate could be based on the use of a coarse computational grid in the PDE solver. This idea is explored in Section 5.4. They could also be based on an analytical approximation of the system of PDE:s that are easier to solve and whose solutions are approximate solutions to the original system. Typically, a surrogate of the second type provides a global approximation of f.

How should surrogates be used in the optimization process? The simplest would perhaps be to first optimize on the surrogate and then use the optimum as the initial guess for optimization on the "real" function. The drawback is that we have no control of the quality of $\tilde{f}$. If the surrogate is a bad approximation, this strategy may lead us even further away from an optimum of f than if we had not used the surrogate at all. Furthermore, surrogates of the first type are typically only reliable approximations to f in a neighbourhood of the current iterate. They must be updated during the optimization, and we need a procedure for determining the region where we can expect the surrogates to give meaningful information about f.

It should thus be clear that efficient use of surrogates requires some kind of framework. Such frameworks that are based on the pattern search method are proposed in for example [5, 14, 38]. The frameworks in all these references can, in principle, handle surrogate functions of both types. However, no explicit examples of surrogates of the second type are given.

The reason why the pattern search algorithms are well suited for providing a surrogate optimization framework is the freedom of choice of the exploratory moves algorithm, which need only consist of two things: an *oracle* that suggests a number of steps on the current grid, and a *core pattern check* that has to be done before the grid may be refined. A core pattern check is simply a search through all the steps in the core pattern (Section 2.3.1). We can therefore write the following simple exploratory moves algorithm [5], which is actually just a way of expressing the conditions on the exploratory moves algorithm in Section 2.3.1.

Algorithm 7. *Exploratory moves with an oracle.*
Let $x_k, f(x_k), \Delta_k$ *be given and let* $s_k = 0$.
Let the oracle suggest some steps $s^1, \dots, s^m$ *on the current grid.*
If $f(x_k + s^i) < f(x_k)$ *for some* $i \in \{1, \dots, m\}$ *then* $s_k = s^i$,
else perform a core pattern check:
    If there exists $s \in \Delta_k B\Gamma$ *s.t.* $f(x_k + s) < f(x_k)$ *then*
        take $s_k \in \Delta_k B\Gamma$ *s.t.* $f(x_k + s_k) < f(x_k)$.
Return $s_k, f(x_k + s_k)$.

A surrogate function can now be incorporated into the optimization via the oracle. For example, the oracle may consist of some iterations with an arbitrary optimization procedure on the surrogate. The efficiency of the algorithm evidently depends on how the oracle works and is updated.

We now give the exploratory moves algorithm for a coordinate search algorithm that uses a surrogate function. We write

$$(s_k, f(x_k + s_k)) = cs(x_k, f(x_k), f, \Delta_k)$$

for the ordinary coordinate search algorithm 3, where f is now added to the arguments to indicate on which function the coordinate search is performed.

Algorithm 8. *Exploratory moves for surrogate coordinate search.*
Let $x_k, f(x_k), \Delta_k$ *be given and let* $s_k = 0$.
$(\tilde{s}_k, \tilde{f}(x_k + \tilde{s}_k)) = cs(x_k, \tilde{f}(x_k), \tilde{f}, \Delta_k)$
If $\tilde{s}_k \neq 0$ *then*
    If $f(x_k + \tilde{s}_k) < f(x_k)$
        $s_k = \tilde{s}_k$
    else
        $(s_k, f(x_k + s_k)) = cs(x_k, f(x_k), f, \Delta_k)$
else
    $(s_k, f(x_k + s_k)) = cs(x_k, f(x_k), f, \Delta_k)$.
Return $s_k, f(x_k + s_k)$.

We note how the step is the result of a coordinate search on either f or $\tilde{f}$. Therefore, the basis matrix and the generating matrix are as for the ordinary coordinate search. Furthermore, unless a successful step s_k is found by this algorithm, an unsuccessful coordinate search has been done on f, i.e., a core

pattern check has been done on f. We conclude that the surrogate coordinate search algorithm is an instance of the general pattern search algorithm applied to f.

By plugging the surrogate coordinate search algorithm into the Hooke and Jeeves algorithm and the SBA, we get versions of them that make use of surrogate functions and also are instances of the general algorithm. Note that if $\tilde{f}$ is a bad model of f so that the steps suggested by coordinate search on $\tilde{f}$ never result in lower values of f, these versions turn into the usual ones apart from the extra evaluations of $\tilde{f}$.

Chapter 3

Fluid dynamics

3.1 Introduction

We give a brief introduction to fluid dynamics which serves as the theoretical background for the waterjet inlet flow modelling in Chapter 4. Everything in this chapter can be found in most introductory textbooks on fluid dynamics.

We derive the Navier-Stokes equations of motion for a viscous, incompressible flow when the fluid has constant density and viscosity. We discuss the modelling of steady, turbulent flows with the Reynolds equations and the Standard k-ϵ model. Since it is very important for a flow model to correctly represent the near-wall flow we describe the structure of a turbulent boundary layer and how it can be modelled.

3.2 Navier-Stokes equations

We use the conservation laws of mass and momentum to derive the equations of motion for a viscous incompressible Newtonian fluid of constant density in the absence of body forces. The derivation is valid also for non-stationary flows.

Let $\varphi(\mathbf{x}, t) \in \mathbb{R}^3$ denote the trajectory of a fluid particle initially at $\mathbf{x} \in \mathbb{R}^3$ (i.e. $\varphi(\mathbf{x}, 0) = \mathbf{x}$). Let $V \subseteq \mathbb{R}^3$ be some volume element of the fluid with boundary ∂V and outward normal $\mathbf{n}$ and let $V_t = \{\varphi(\mathbf{x}, t) : \mathbf{x} \in V\}$ denote the volume moving with the fluid.

The law of conservation of mass says that the rate of increase of mass in V equals the mass flow rate into V. Since we consider a fluid of constant density we get, after applying Gauss' Theorem,

$$\int_V \nabla \bullet \mathbf{u} dV = 0. \tag{3.1}$$

The momentum of the fluid in V_t is

$$\mathbf{m}(t) = \int_{V_t} \rho \mathbf{u}(\mathbf{x}, t)\, dV. \tag{3.2}$$

The forces on a volume element are usually divided into body and surface forces,

of which we will ignore the former in order to simplify the presentation[1]. The surface stresses of the volume element are pressure and internal friction and may be represented by the *stress tensor*[2] $\sigma(\mathbf{x},t) \in \mathbb{R}^3 \times \mathbb{R}^3$ so that $\sigma(\mathbf{x},t)\mathbf{n}$ is the force per unit area at time t on a surface element through $\mathbf{x}$ perpendicular to $\mathbf{n}$. Hence, using the transport theorem [10] to differentiate (3.2) with respect to t, we get

$$\mathbf{m}'(t) = \int_{V_t} \rho \frac{D\mathbf{u}}{Dt}\, dV = \int_{\partial V_t} \sigma\, \mathbf{n}\, dV = \int_{V_t} \nabla \bullet \sigma\, dV, \tag{3.3}$$

where the second equality is Newton's second law and the third equality is Gauss' Theorem. Here, $D\mathbf{u}/Dt = \partial\mathbf{u}/\partial t + (\mathbf{u} \bullet \nabla)\mathbf{u}$ is the *the total derivative* of $\mathbf{u}$ (the acceleration of a fluid particle following the fluid), and $(\nabla \bullet \sigma)_i = \partial\sigma_{ij}/\partial x_j$ (a repeated index means a summation over that index).

We now make appropriate regularity assumptions on p and $\mathbf{u}$. Then, since Equations (3.1) and (3.3) are valid for arbitrary volumes $V(= V_{t=0})$, we may remove the integral signs to get

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla \bullet \sigma \tag{3.4}$$

and

$$\nabla \bullet \mathbf{u} = 0. \tag{3.5}$$

Water is an example of a *Newtonian* fluid [1] for which

$$\sigma = pI + 2\mu D, \tag{3.6}$$

where μ is the *coefficient of viscosity* and

$$D_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3.7}$$

is *the rate of deformation* (or strain) tensor. The second term in (3.6) represents the *viscous stresses.*

For an inviscid fluid viscous stresses are ignored and $\sigma = pI$. Substitution into (3.4) gives the Euler equations. For viscous incompressible fluids it follows from Equation (3.5) that $\nabla \bullet 2D = \nabla^2\mathbf{u}\,(= (\triangle u_1, \triangle u_2, \triangle u_3))$. We then arrive at the following four partial differential equations:

$$\begin{cases} \rho \dfrac{D\mathbf{u}}{Dt} = -\nabla p + \mu\nabla^2\mathbf{u}, \\ \nabla \bullet \mathbf{u} = 0. \end{cases} \tag{3.8}$$

These are the Navier-Stokes equations. The first three equations represent the balance of momentum and the last equation expresses the conservation of mass. In the balance of momentum equations, the first term is called the inertia force, the second the pressure force and the third the viscous force.

[1] Actually, when modelling the s-duct flow in the following chapter, it is possible to account for gravitational effects without including gravitation in the equations. See discussion in Section 4.4.6.

[2] One can ask why σ should be a matrix. Actually, using Newton's second law and the assumption that σ is a continuous function it can be *proved* that σ is a linear function of $\mathbf{n}$. This is done in [20].

3.3 Reynolds number

An important quantity for describing a flow is the *Reynolds number*, defined as

$$Re = \frac{UL}{\nu},$$

where U is a typical flow speed and L a typical distance over which the flow changes in a significant way. In the case of flow through a duct with circular cross section, L can be taken to be the duct diameter. For a stationary flow the Navier-Stokes equations become

$$(\mathbf{u} \bullet \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}.$$

We may expect to have

$$|(\mathbf{u} \bullet \nabla)\mathbf{u}| \sim \frac{U^2}{L} \qquad \text{and} \qquad |\nu\nabla^2\mathbf{u}| \sim \frac{\nu U}{L^2},$$

which gives

$$\frac{|(\mathbf{u} \bullet \nabla)\mathbf{u}|}{|\nu\nabla^2\mathbf{u}|} \sim \frac{U^2/L}{\nu U/L^2} = Re.$$

That is, the Reynolds number indicates the relative importance of inertia to viscous forces in typical parts of the flow domain. A flow with a high Reynolds number differs a lot from a flow with a low one. For the latter, viscous forces dominates over inertia forces so that the flow equations may be approximated by

$$\nabla p = \mu\nabla^2\mathbf{u}, \tag{3.9}$$

which is sometimes called the equations of creeping motion. Since they are linear they are much simpler than the full Navier-Stokes equations and because they are of the same degree one can use the same boundary conditions.

With a high Reynolds number, viscous flow effects are so small that they may be ignored, giving the approximation

$$\rho\mathbf{u} \bullet \nabla\mathbf{u} = -\nabla p. \tag{3.10}$$

These are the Euler equations. The flow is driven by pressure differences so that fluid particles are accelerated in the direction of the pressure gradient at each point. Since they are of lower degree than the Navier-Stokes equations the number of boundary conditions has to be reduced.

3.4 Boundary layer

For a high Reynolds number, viscous effects are negligible in most part of the fluid domain so that Equation (3.10) is valid. Since these equations are of lower degree, one has to reduce the number of boundary conditions. The appropriate procedure is to replace the no-slip condition at the walls with an impermeability condition. Experimentally, though, it is found that the no-slip condition

continues to apply no matter how high the Reynolds number is. Hence there is a thin region close to the walls where the flow adjusts itself to the no-slip condition, resulting in much larger values of $\partial^2 u/\partial y^2$ than far from the wall (u being the velocity parallel and y the distance perpendicular to the wall). This region, where viscous effects remain important, is called the *boundary layer* and its approximate thickness denoted by δ. It can be shown that ([1])

$$\frac{\delta}{L} \sim \frac{1}{Re^{1/2}}.$$

Sometimes δ is defined as the distance from the wall where the flow differs one percent from the inviscid flow solution.

Even if the boundary layer is very thin for high Reynolds number flows, its presence is still very important for certain aspects of the flow. (Consider for example d'Alembert's paradox: the drag force on an obstacle in a potential flow is zero.)

3.5 Turbulent flow

We discuss some basic properties of turbulent flows. In Section 3.5.1 we state and motivate the Reynolds equations. In Section 3.5.2 we briefly describe the structure of a turbulent boundary layer.

With an increasing Reynolds number (for example as the result of an increasing mean velocity) the flow eventually becomes unstable. Small disturbances are amplified and may lead to a fully turbulent flow. A necessary condition for turbulence to develop and sustain is the existence of a mean velocity gradient ([27]). A turbulent flow is characterized by chaotic and irregular flow particle trajectories. Local quantities change unpredictably in time, even if the imposed boundary conditions are stationary. However, turbulence is not a completely random phenomenon since, often, it gives rise to some kind of patterns at a large scale. For example, a turbulent velocity field have certain spatial structures known as *eddies*. There are always eddies of a wide range of sizes, small eddies existing inside larger eddies ([27]). A turbulent flow is much more dissipative than a laminar flow, because of the work of the small eddies against viscous stresses. Also, "important constituents of the turbulence phenomenon take place in eddies of the order of a millimetre in size, while the whole flow domain may extend over meters or kilometers" ([21]).

3.5.1 Reynolds equations

It is clear from what has been said above that even though the Navier-Stokes equations are valid theoretically, they are not appropriate to be used in a practical model for turbulent flows since such a model would require an impractical number of grid nodes in order to resolve the small-scale (stochastic) effects on the large-scale (average) flow behaviour. The remedy is to adopt a statistical aproach. Each flow quantity q is divided into

$$q = Q + q',$$

where Q is an averaged component and q' is a fluctuating component. We denote averaging by bar, i.e., $Q = \bar{q}$. By definition, $\bar{q'} = 0$. In a stationary

flow the average may be seen as a time average. If the flow is explicitly time dependent, the average quantity may be considered to be the average of the quantity at corresponding instances and locations over a number of identical flow setups ([37], p. 300).

If we put $u_i = U_i + u_i', p = P + p'$ into the steady form of the Navier-Stokes equations and then take the average, we get the *Reynolds equations* for the mean quantities U, P:

$$\begin{cases} \rho U_j \dfrac{\partial U_i}{\partial x_j} = -\dfrac{\partial P}{\partial x_i} + \dfrac{\partial}{\partial x_j}\Big(\mu \dfrac{\partial U_i}{\partial x_j} - \rho\overline{u_i' u_j'}\Big), \\ \dfrac{\partial U_i}{\partial x_i} = 0. \end{cases} \tag{3.11}$$

The six quantities $\tau_{ij} = -\rho\overline{u_i' u_j'}$ behave as additional stresses on the fluid and are known as the *Reynolds stresses*. They represent the effect of the small-scale fluctuations on the large-scale mean flow arising from the non-linearity of the Navier-Stokes equations. Mathematically, the Reynolds stresses appear as six new variables and must be modelled in order to get a closed system (a system with enough number of equations to determine all the unknowns). This closure problem is the fundamental issue of *turbulence modelling*.

3.5.2 Turbulent boundary layers

The presence of solid boundaries, "walls", significantly affects turbulent flows. The no-slip condition enforces large velocity gradients close to the wall which give rise to the production of turbulent energy. Seil [34] states that solid boundaries "act as sources of turbulence and energy loss".

The structure of a turbulent boundary layer may be described by several layers depending on the behaviour of the turbulence in each layer. According to Hinze ([17], p. 587) a turbulent boundary layer of thickness δ consists of an inner region, where the flow is directly influenced by viscous effects, and an outer region, where the flow is fully turbulent and viscous effects are negligible. In the inner region ($0 \leq y/\delta \lesssim 0.2$) the shear stress is approximately constant and equal to the wall shear stress τ_w ([26]). The inner region, in turn, consists of the very thin *viscous sublayer* ($y/\delta < 0.002$)), where viscous stresses dominate, the *buffer layer*, where both viscous and turbulent (i.e. Reynolds) stresses are important, and the *log-law layer* ($0.02 \lesssim y/\delta \lesssim 0.2$) where turbulent stresses dominate.

In the log-law layer it is possible to derive a universal, logarithmic relation between the velocity U tangential to the wall and the distance y normal to the wall. This relation is known as the *log-law* and reads as

$$U^+ = \frac{1}{\kappa} \ln(Ey^+), \tag{3.12}$$

where

$$U^+ = \frac{U}{u_\tau}$$

and

$$y^+ = \frac{u_\tau y}{\nu}.$$

The friction velocity u_τ is defined by

$$u_\tau = (\tau_w/\rho)^{1/2},$$

where τ_w is the wall shear stress. The log-law can be derived from arguments based on the Boussinesq hypothesis, the assumption of a constant shear stress, an assumption concerning the length scale of the near-wall turbulence, and dimensional analysis. The numerical values of the constants are found from experiments to be $\kappa = 0.41$ (von Karman's constant) and $E = 9.8$ for a smooth wall ([26]). Experiments also show the log-law to be valid for $30 < y^+ < 500 \sim 1000$ ([17]).

3.6 Turbulence modelling

Many models for the closing of the Reynolds equations have been developed. They differ in accuracy, range of applicability and computational cost. One of the most popular and well-established ways of closing the Reynolds equations is the Standard k-ϵ model. According to the Fluent manual [15], its popularity for industrial flow calculations is due to its "robustness, economy, and reasonable accuracy for a wide range of turbulent flows". Malalasekera and Versteeg [26] remark that the k-ϵ model is the most widely used and validated turbulence model. Fluent also allows for the option of using more elaborate turbulence models. They state that for certain flows it is necessary to choose a Reynolds stress model (RSM), but that the RSM is computationally more expensive and less robust than the k-ϵ model.

Since the optimization presented later in this work requires a large number of flow simulations to be done, it is essential to choose a robust and computationally efficient CFD model. We have therefore decided to use the Standard k-ϵ turbulence model, which is presented in greater detail in the following section. The boundary conditions for the k-ϵ model are discussed in Section 3.6.2. In particular, we discuss how to specify the inlet conditions in practice, and how the effects of boundary layers are taken care of in order to improve the accuracy and efficiency of the model.

3.6.1 The Standard k-ϵ turbulence model

The *Standard k-ϵ model* ([21], [15]) consists of two transport equations for the turbulent kinetic energy k,

$$k = \frac{1}{2}\overline{u_i' u_i'},$$

and its dissipation rate ϵ,

$$\epsilon = \nu \overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}}.$$

The version of the model described here is derived for high Reynolds numbers, using the hypothesis of Boussinesq (see below) and assuming that the turbulence is *isotropic*, which means that the statistical properties of the turbulence are equal in all spatial directions. Hinze [17] points out that even though the

assumption of isotropy is not true for any actual flow, it often yields valuable approximations also when the turbulence has essential nonisotropic characteristics.

In the Reynolds equations (3.11), the Reynolds stresses appear together with the viscous stresses. Therefore, it is tempting to assume that also the former are proportional to the mean velocity gradient (*the Boussinesq hypothesis*). Together with the assumption of an isotropic turbulence, this leads to the modified Boussinesq hypothesis [17]:

$$-\rho\overline{u_i'u_j'} = 2\mu_t\overline{D}_{ij} - \frac{2}{3}\rho k\delta_{ij}. \tag{3.13}$$

δ_{ij} is the Kronecker delta function, $\overline{D}_{ij}$ the average rate of deformation and μ_t the *turbulent viscosity*.

Arguments based on dimensional analysis under the assumption that one length scale and one velocity scale suffice to describe the effects of turbulence yields ([26])

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon}, \tag{3.14}$$

where $C_\mu = 0.09$.

By a Reynolds-averaging procedure it is possible to derive two transport equations for k and ϵ from the Navier-Stokes equations. Modelling some of the terms in these two equations (see for example [26]) gives the Standard k-ϵ model for high Reynolds numbers ([21]):

$$\begin{cases} \rho\dfrac{Dk}{Dt} = \dfrac{\partial}{\partial x_i}(\mu_t\dfrac{\partial k}{\partial x_i}) + 2\mu_t\bar{D}_{ij}\bar{D}_{ij} - \rho\epsilon \\ \rho\dfrac{D\epsilon}{Dt} = \dfrac{\partial}{\partial x_i}(\dfrac{\mu_t}{\sigma_\epsilon}\dfrac{\partial \epsilon}{\partial x_i}) + C_{1\epsilon}\dfrac{\epsilon}{k}2\mu_t\bar{D}_{ij}\bar{D}_{ij} - C_{2\epsilon}\rho\dfrac{\epsilon^2}{k}, \end{cases} \tag{3.15}$$

where

$$\sigma_\epsilon = 1.3, \qquad C_{1\epsilon} = 1.44, \qquad C_{2\epsilon} = 1.92.$$

Once Equation (3.15) has been solved, k and ϵ determine the Reynolds stresses through the modified Boussinesq hypothesis (3.13) and Equation (3.14).

To sum up, in the Standard k-ϵ model, the six Reynolds stresses are modelled by two new unknowns, k and ϵ. A closed system for the six unknowns U_i, P, k and ϵ are made up of the six partial differential equations in (3.11) and (3.15), together with the seven algebraic relations (3.13) and (3.14).

3.6.2 Boundary conditions for the k-ϵ model

In Fluent, the boundary conditions for k and ϵ at the outlet and at the walls are taken care of by the software, while the user has to specify the inlet condition. In the first part of this section we discuss how the inlet conditions may be specified in practice. In the second part we outline how low-Reynolds-number boundary layer effects can be taken into account by the high-Reynolds-number Standard k-ϵ model given above, by the use of *wall functions*.

Specifying the inlet boundary conditions

In industrial practice it is rare to have access to experimental measurements of the exact distributions of k and ϵ at the inlet. Instead, uniform values of the turbulent quantities can be obtained from empirical formulas, which we now discuss. In the Fluent manual [15] it is noted that this procedure is appropriate, for example, in the case of a fluid entering a duct, and that often the calculation is relatively insensitive to the turbulence inlet data, because the shear layers generate more turbulence than enters the duct at the inlet. Nevertheless, unphysically large inlet values can significantly affect the solution and the convergence behaviour.

One way to specify constant inlet boundary values for k and ϵ in Fluent is to use the turbulence intensity I and the turbulence length scale l. The relationship between k and I is ([15], pp. 6-14)

$$k = \frac{3}{2}(U_{ref}I)^2,$$

where U_{ref} is some reference flow speed. The turbulent rate of dissipation can be determined from l by ([15] pp. 6-14)

$$\epsilon = C_\mu^{3/4}\frac{k^{3/2}}{l}.$$

Furthermore, since the eddy viscosity is related to k and ϵ by Eq. (3.14) we get

$$\mu_t \propto I\, l.$$

Thus, the turbulent viscosity increases when either I or l is increased.

In fully-developed duct flows, l can be approximated by

$$l = 0.07D_H, \tag{3.16}$$

where D_H is the hydraulic diameter ([15] pp. 6-12). If one knows the boundary layer thickness, δ_{99}, at the inlet, a better estimate of l is ([15], pp. 6-13)

$$l = 0.4\delta_{99}. \tag{3.17}$$

Near-wall treatment

As discussed in Section 3.5.2 turbulent flows are significantly affected by the presence of walls. Fluent [15] notes that an accurate near-wall modelling is important in order to get reliable numerical solutions. In general, there are at least two methods for the numerical modelling of the turbulent boundary layer ([21]): the *wall-function-method* and the *low-Reynolds-number-modelling method*. In the latter, boundary layers are resolved by using fine enough computational grids. The former avoids the need for fine near-wall grids by the use of *wall functions* that describe the relations between the involved quantities (velocity etc) at a certain distance from the wall. For high Reynolds number flows, it therefore saves computational resources, as noted by Fluent [15].

Since the Standard k-ϵ model described above was developed for high Reynolds number flows, it will give incorrect solutions if the low-Reynolds-number-modelling

method is used. Instead the wall function approach is employed. A set of wall functions for the Standard k-ϵ model are:

$$\frac{U}{u_\tau} = \frac{1}{\kappa} \ln\left(\frac{E\, u_\tau\, y}{\nu}\right), \qquad k = \frac{u_\tau^2}{C_\mu^{1/2}}, \qquad \epsilon = \frac{u_\tau^3}{\kappa y}.$$

These relations should be fulfilled at the wall-adjacent nodes. Since the log-law is only valid in the log-law layer, these nodes should be well inside the log-law layer, but not closer; $30 < y^+ < 200$, say. It should be noted that the friction velocity u_τ is not known a priori and hence it is not obvious how to employ the above formulae in practice. Fluent [15] follows the procedure outlined by Launder and Spalding [21].

The first relation is the log-law (Eq. (3.12)). To derive the other two, we note that in the near-wall region the production term in the transport equation for k becomes ([34])

$$G_k = -\rho\, \overline{u'v'}\, \frac{\partial U}{\partial y}, \tag{3.18}$$

where v is the velocity normal to the wall. In the inner region, the shear stress is approximately constant and equal to the wall shear stress. Therefore, since Reynolds stresses dominate in the log-law layer, we have

$$-\rho\overline{u'v'} = -\rho u_\tau^2 \,(= \tau_w).$$

Furthermore, differentiation of the log-law gives

$$\frac{\partial U}{\partial y} = \frac{u_\tau}{\kappa y}. \tag{3.19}$$

The last two equations inserted into Equation (3.18) result in

$$G_k = \rho \frac{u_\tau^3}{\kappa y}. \tag{3.20}$$

We now assume that the production G_k of turbulent kinetic energy equals the rate of dissipation $\rho\epsilon$ (see [17], pp. 649). Hence,

$$\epsilon = \frac{u_\tau^3}{\kappa y}.$$

Inserting this expression for ϵ into Equation (3.14) and solving for k gives

$$k^2 = \frac{\mu_t u_\tau^3}{C_\mu \rho \kappa y}. \tag{3.21}$$

The Boussinesq hypothesis,

$$-\rho\, \overline{u'v'} = \mu_t \frac{\partial U}{\partial y},$$

together with Equation (3.19) allow us to write Equation (3.18) as

$$G_k = \mu_t \frac{u_\tau^2}{\kappa^2 y^2}. \tag{3.22}$$

Equating the two expressions (3.20) and (3.22) for G_k and solving for μ_t results in

$$\mu_t = \rho \, u_\tau \, \kappa \, y.$$

By inserting this expression for μ_t into Equation (3.21) we finally arrive at

$$k = \frac{u_\tau^2}{C_\mu^{1/2}}.$$

Chapter 4

A waterjet inlet duct performance model

4.1 Introduction

This chapter and the following are concerned with the waterjet inlet design problem stated in the introduction to the thesis. We present a model of the inlet duct and its hydrodynamic performance, which is then optimized in the next chapter. The inlet performance model consists of

- a model for the generation of duct geometries,
- a model of the duct flow,
- a model for the assessment of the hydrodynamic performance.

Before going into the details, we briefly discuss each one of them, and their interaction with the optimization to come.

Geometry

In order to generate geometries, a *parameterization* of the inlet duct geometry has been performed. The idea is to describe the geometry by finitely many real numbers, or parameters. Each parameter represents an important aspect of the geometry, such as a radius, characteristic length etc, and to each set of given parameter values is associated a geometry (shape, inlet duct flow domain) D. When the parameters are varied, they generate a set of geometries on which the optimization can be performed. Mathematically, a parameterization can be seen as a mapping

$$D : \Omega \subseteq \mathbb{R}^n \to \mathbb{P}_3, \qquad D = D(\mathbf{p}), \qquad \mathbf{p} \in \mathbb{R}^n, \tag{4.1}$$

where Ω is a subset of $\mathbb{R}^n$, n is the number of parameters and $\mathbb{P}_3$ the set of all subsets of $\mathbb{R}^3$. $\mathbf{p}$ is the vector of parameters and D is the geometric shape. The details of the parameterized geometry are given in Section 4.2.

The parameterization reduces the number of degrees of freedom for the duct geometry from infinitely to finitely many. This is necessary in order to obtain

a finite dimensional optimization problem, which can be solved by a computer. It is clear that this drastic reduction in degrees of freedom makes the choice of parameters a crucial step in our modelling. For the optimization of the model to be meaningful, the parameters must generate a relevant set of duct geometries. For the optimization to be efficient, the parameters should be as few as possible, and somehow independent of each other. To choose the parameters one has to be familiar with the problem, and have a feel for what geometric aspects that could be of relevance. Physical parameters (length, radii etc) will make it easier to obtain independent parameters, control the behaviour of the model and to interpret the optimization results.

Flow

We have used computational fluid dynamics (CFD) techniques to compute the steady, incompressible, viscous and turbulent flow through the inlet. A CFD analysis consists of two steps. Firstly, the flow domain is discretized by the generation of a computational *grid*, or *mesh*. This step has been performed with the commercial software Gambit. Secondly, the discretized flow equations are solved on the grid. This step has been done with the commercial software Fluent.

The fact that it will be used in the context of optimization imposes special requirements on the CFD model. First of all, we must find a procedure for the automatic generation of acceptable grids for a wide range of geometries. Grid quality is important for fast and accurate CFD calculations. In general, the generation of high quality grids is a demanding task, but in our case it is quite straightforward because of the simplicity of the flow domain geometry. Furthermore, in view of the fact that a large number of CFD calculations are performed in the course of an optimization, robustness and minimal execution time become essential properties of the CFD model. This is further discussed in Section 5.4.

Our CFD model, presented in Section 4.4, is based on the Reynolds equations (3.11) with k-ϵ modelling of the Reynolds stresses (Section 3.6). The reliability and accuracy of the model is validated by comparing solutions obtained in Fluent to the experimental data of Bansod and Bradshaw [3]. The experimental validation is also a way of adjusting parameters such as boundary conditions, grid etc, to improve the performance of the CFD model.

Performance

In Section 4.5, we discuss a model for the assessment of the hydrodynamic performance of a given duct. The model can be seen as a mapping which to each duct represented by the parameter vector $\mathbf{p}$ assigns a real number $f(\mathbf{p})$ that measures some aspect of the hydrodynamic performance. The aspects of hydrodynamic performance that we consider are total pressure loss and the risk of inception of cavitation as measured by the minimum static pressure in the duct.

4.2 Geometric modelling

In this section we discuss the modelling of the waterjet inlet duct geometry. The geometric simplifications that have been done are accounted for in Section 4.2.1. The parametric model of the duct geometry is described in Section 4.2.2. Section 4.2.3 briefly explains how the duct geometry can be represented by NURB curves.

The implementation of the geometry has been done in Gambit. (Grid generation with Gambit is discussed in Section 4.3.4.)

4.2.1 Geometric simplifications

In order to perform a CFD evaluation of the flow in the waterjet inlet, it would be desirable to model at least the following aspects of the geometry:

1. the shape of the waterjet inlet itself,
2. the pump shaft,
3. the inlet grille.

The emphasis of this work is on the geometric shape of the waterjet inlet and in fact it is the only geometric feature that has been modelled. We now briefly discuss why the other two features can, and have been, neglected.

As can be seen in Fig. 1.1, the pump shaft passes through the waterjet inlet and will clearly influence the flow, whence its effect on the flow behaviour should be modelled somehow. Nevertheless, this geometric feature has not been taken into account in what follows, mainly because the grid topologies used in this work make a direct geometric representation of the pump shaft impractical.

Also the effect of the grille has been neglected, for three reasons. Firstly, to represent the individual bars of the grille would require a very large mesh. Secondly, it can be assumed that the effects on the flow caused by the grille is more or less independent from the effects caused by the shape of the waterjet inlet. Finally, not all waterjet inlets have a grille fitted.

4.2.2 Parameterization of the s-duct geometry

The waterjet inlet has been modelled by an expanding s-duct consisting of a combination of straight duct sections and circular arc sections. The cross-sectional shape is circular, but the radius of each cross-section is allowed to vary. The dimensions of the duct have been proposed by Rolls-Royce AB. The diameter of the inlet opening is set to $D_{in} = 0.4$ m and the diameter of the outlet opening is set to $\sqrt{2} \cdot D_{in}$. The height (y-direction) is fixed to $H = 3$ m. The total length in the x-direction will be denoted by L.

The s-duct can be described by the *center curve* of the duct, and a *radius curve*, which gives the radius of the cross sections as a function of the arc length parameter along the center curve.

The generic form of the center curve and the three parameters it has been parameterized with is shown in Fig. 4.1. In words, the center curve consists of

1. a straight line of length D_{in},
2. a circular arc of radius R_1 (variable),

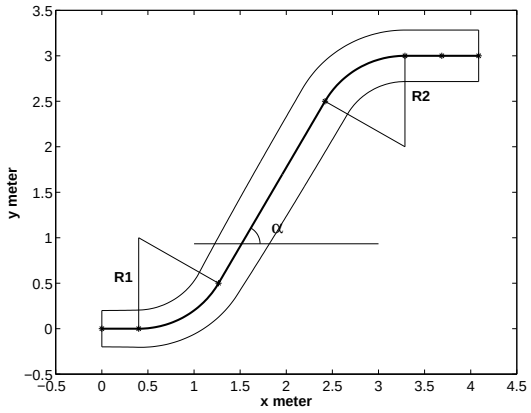


Figure 4.1: The duct geometry and the six segments of the duct center curve.

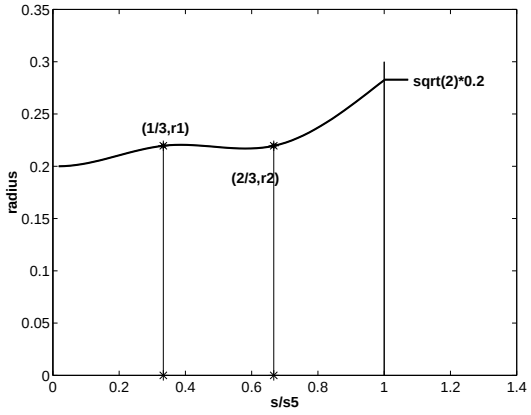


Figure 4.2: The defining parameters for the radius curve. Note that the arc length parameter s has been scaled by s_5.

3. a straight line, inclined by α degrees (variable),
4. a circular arc of radius R_2 (variable),
5. a straight line of length D_{in},
6. a straight line of length D_{in} along which the radius of the cross section is constant equal to $\sqrt{2} \cdot D_{in}/2$.

The radius curve r is a continuous curve with two parameters. More specifically, let s be the arc length parameter of the center curve and denote by s_i the arc length of the first i center curve segments. We define r by

$$r(s) = \begin{cases} q(s), & 0 \le s \le s_5, \\ \sqrt{2} \cdot D_{in}, & s_5 \le s \le s_6, \end{cases}$$

where q is a cubic spline with initial value $q(0) = D_{in}/2$, final value $q(s_5) = \sqrt{2} \cdot D_{in}/2$ and initial derivative and final second derivative equal to zero. Also,

we require $q(s_5/3) = r_1$ and $q(2s_5/3) = r_2$. As a consequence, the radius curve is continuous on the interval $[0, s_6]$ and has continuous second derivative on $[0, s_5)$. Its parameters are r_1 and r_2. For practical reasons, we will use $r(s/s_5)$ instead of $r(s)$, as in Fig. 4.2.

Constraints

The five parameters α, R_1, R_2, r_1, r_2 given above are subject to certain constraints. For example, we have the natural restriction $R_i > 0$. We now give a complete list of constraints. It should be noted that they are *sufficient* for guaranteeing that the parameterization makes sense, and sometimes, in order to guarantee that the construction of the duct is possible in practice, they have been chosen stronger than necessary.

$$\alpha \leq \frac{\pi}{2}, \tag{4.2}$$

$$3D_{in} + H \cdot \tan\left(\frac{\pi}{2} - \alpha\right) + (R_1 + R_2)tan\left(\frac{\alpha}{2}\right) \leq L_{max}, \tag{4.3}$$

$$(R_1 + R_2)\tan\left(\frac{\alpha}{2}\right) + \frac{D_{in}}{2} \leq \frac{H}{\sin(\alpha)}, \tag{4.4}$$

$$\frac{D_{in}}{4} \leq r_i, \quad i = 1, 2, \tag{4.5}$$

$$r_i + \frac{D_{in}}{2} \leq \min\{R_1, R_2\}, \quad i = 1, 2. \tag{4.6}$$

Constraint (4.2) expresses that we do not allow the mid-section of the centreline to be inclined more than 90 degrees. Constraint (4.3) restricts the total duct length in the x-direction. To be able to connect the two centreline bends smoothly with a straight line, the bend radii can not be too large. This is guaranteed by constraint (4.4), which also separates the bends by at least $D_{in}/2$. Eq. (4.5) imposes lower bounds on the radius along the duct. If the duct radius at one of the bends are greater than the bend radius, the duct surface will "intersect itself", and the practical geometric construction will fail. This is avoided, with some margin, by constraint (4.6).

Summary of parametric model

The following five parameters have been used to parameterize the inlet duct geometry (Fig. 4.1 and 4.2):

- α the angle of inclination of the duct,
- R_1 the upstream bend radius,
- R_2 the downstream bend radius,
- r_1 the duct radius one third along the duct center curve,
- r_2 the duct radius two thirds along the duct center curve,

where r_1 and r_2 are interpolated by the constinuous radius curve. These parameters are subject to certain constraints that can be represented by the set

$$\Omega = \{\mathbf{p} \in \mathbb{R}^5 : \mathbf{p} = (\alpha, R_1, R_2, r_1, r_2) \text{ and } \alpha, R_1, R_2, r_1 \text{ and } r_2 \text{ fulfils the constraints in Eq. (4.2) to (4.6)}\}. \tag{4.7}$$

The parameterization may now be expressed as

$$D : \Omega \subseteq \mathbb{R}^n \to \mathbb{P}_3, \qquad D = D(\mathbf{p}), \qquad \mathbf{p} = (\alpha, R_1, R_2, r_1, r_2), \tag{4.8}$$

where D is the s-duct flow domain and, again, $\mathbb{P}_3$ is the set of all subsets of $\mathbb{R}^3$.

4.2.3 Representation of the parametric geometry

The implementation of the geometry has been done in Gambit; see Section 4.3.4 for further discussion. To communicate the geometry to Gambit it is necessary to have a practical and efficient representation of the center and the radius curves discussed in the previous section. We have used NURB (non-rational uniform B-spline) curves for this task. A NURB curve of degree p is a piecewise rational polynomial of degree p, which can be defined as

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}, \qquad 0 \le u \le 1,$$

where $\{\mathbf{P}_i\}$ are called the *control points*, $w_i > 0$ are the *weights* and $N_{i,p}$ is the p:th B-spline basis function (a piecewise polynomial of degree p with compact support, i.e., which is zero outside a subinterval of $[0, 1]$).

Details about the properties of NURB curves are deferred to the Appendix. Let us just remark that NURB curves provide an excellent tool for the representation and local shape control of geometric features. The main reason for this is the compact support of the B-spline basis functions so that the perturbation of a control point will influence only a small part of the curve. The main reason, however, that we have used NURB curves, is that they provide an efficient and compact representation of piecewise defined curves, and they can be used for an exact representation of, for example, circular arcs. The definition of NURBs allows for flexibility in the choice of parameterization and it is possible to approximate a uniform parameterization. It should thus be clear that NURB curves are a natural choice for the representation of the center and radius curves.

4.3 The computational grid

In order to perform a numerical simulation of the s-duct flow, the flow domain must be discretized. This is accomplished by the computational *grid*, or *mesh*. The mesh is an essential part of a CFD model. The complete CFD model is discussed in Section 4.4. In Sections 4.3.1 and 4.3.2 we describe two kinds of structured grids appropriate for an s-duct. The grid quality is discussed in Section 4.3.3. The practical meshing procedure is described in Section 4.3.4. Note that because of the symmetry of the flow in an s-duct only half the geometry is meshed.

4.3.1 O-grids

We first consider a single-block mesh with a cylindrical geometry that fits naturally to the s-duct geometry. It consists of an O-type grid over the cross-section of the s-duct (Figure 4.3), which is then swept along the centreline to generate the complete mesh (Figure 4.4). We will call such a grid an O-grid. The O-grid consists of both wedge elements and hexahedral elements.

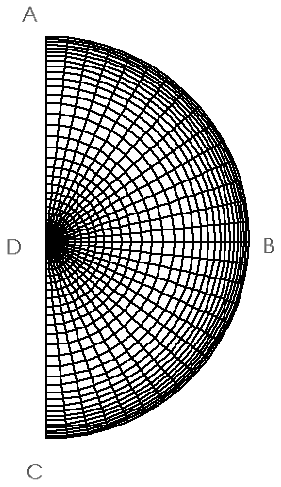


Figure 4.3: Cross-sectional grid topology for the O-grid.

The following parameters have been used to specify the exact size and shape of the O-grid:

- m_1: Number of mesh intervals along the duct, i.e., on edge AE in Figure 4.4.
- m_2: Number of mesh intervals on the arc AB (half of the semi-circle ABC) in Figure 4.3.
- m_3: Number of mesh intervals on edge AD.
- sr_1: Successive ratio of mesh intervals at end A of edge AD.
- sr_2: Successive ratio of mesh intervals at end D of edge AD.

The parameters m_1, m_2, m_3 determine the size of the grid. The number of mesh intervals along the duct, m_1, are uniformly distributed along the centreline. The purpose of sr_1 is to stretch the mesh on edge AD towards end point A in order to obtain a clustering of grid points close to the wall. This is needed to obtain a grid that resolves the near-wall flow with only a moderate number of grid cell elements. sr_2 is used to stretch the mesh on edge AD towards end point D. The motivation for this is given in Section 4.3.3.

4.3.2 Butterfly grids

The second mesh we consider is a multi-block structured hexahedral grid. The cross-section of the s-duct has been divided into an inner quadratic region and four outer regions, giving the butterfly-like cross-sectional grid shown in Figure

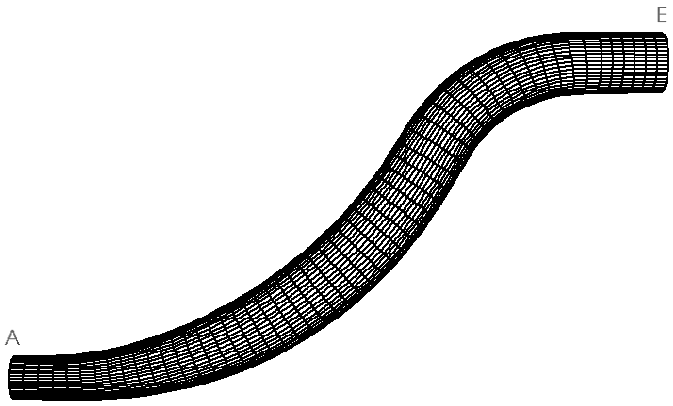


Figure 4.4: Wall grid.

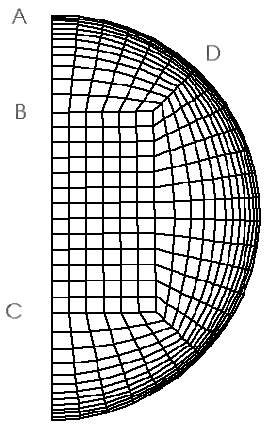


Figure 4.5: Cross-sectional grid topology for the butterfly grid.

4.5. This cross-sectional grid is then swept along the centreline of the s-duct. We will call this type of grid a butterfly grid.

The following parameters have been used to specify the exact size and shape of the butterfly grid:

- m_1: Number of mesh intervals along the duct, i.e. on edge AE in Figure 4.4.
- m_2: Number of mesh intervals on edge BC in Figure 4.5.
- m_3: Number of mesh intervals on edge AB.
- sr: Successive ratio of mesh intervals at end A of edge AB.
- ra: The ratio between the duct radius and half the length of the diagonal in the inner quadratic region of the cross-sectional grid.

Again, the parameters m_1, m_2, m_3 determine the size of the grid and the number of mesh intervals along the duct, m_1, are uniformly distrubuted along the centreline. sr serves the same purpose as sr_1 for the O-grid and it is used together with ra, that determines the relative size of the inner quadratic region, to obtain a grid that resolves the near-wall flow with only a moderate number of grid cell elements.

4.3.3 Grid quality

The mesh quality plays a significant role in the accuracy and convergence behaviour of the numerical computation. The quality of the mesh is affected by the grid node distribution, the grid smoothness and the shape of the grid cells, as noted by Fluent [15].

The grid node distribution is important in order to resolve characteristic features of the flow. For example, the resolution of a boundary layer requires a high grid node density. For turbulent flows, the required near-wall grid density depends on the turbulence model being used.

A grid is smooth if the changes in cell volumes between adjacent cells are small. Non-smoothness will increase truncation errors in the discretized equations and thus decrase the accuracy of the solution.

Two cell shape attributes are skewness and aspect ratio. The skewness measures how much the cell element angles deviates from the angle in an equilateral element of the same type. Highly skewed cells can decrease the accuracy and destabilize the solution. The aspect ratio tells how stretched the cell is.

Both the O-grid and the butterfly grid admit for good resolution of the near-wall region. However, this involves a stretching of the grid, which will decrease the grid smoothness. The O-grid posses very good orthogonality in all cells except for in the wedge cells near the centreline. The skewness of the wedge cells will increase with the number of grid points in the circumferential direction, but their influence on the solution can be decreased by a stretching of the grid towards the centreline. The block-structure of the butterfly grid results in both skewness and further decreased smoothness. As a consequence of the stretching in the radial direction, some cells will have large aspect ratios. In both grids, the gridlines are aligned with the primary flow direction, which should minimize the numerical diffusion [15].

4.3.4 Grid generation

The generation of O-grids and butterfly grids for the s-duct has been done with the commercial software Gambit. The meshing procedure in Gambit starts with the definition of the flow domain geometry. The user defines the vertices that are connected by edges. The edges are assembled to form surfaces and finally volumes are constructed from the surfaces. The actual meshing is accomplished in a similar way by first meshing the edges, then the surfaces and finally the volumes. Gambit can be executed manually from a graphical interface, or automatically by reading the commands from an external text file, called the journal file, provided by the user.

Since the grid generation will be used in an optimization procedure, it has to be done automatically. We have had access to a computer program written in C++ by Sara Ågren [40] that, given the geometry data and a chosen set of mesh parameters, prints the corresponding Gambit journal file. Gambit takes the journal file as input, executes all the commands in it and finally prints a mesh data file which can be read by Fluent.

4.4 CFD model and experimental validation

The intension of this section is to describe the calibration and validation of a CFD model for the turbulent flow in the waterjet inlet s-duct model. As pointed out in the introduction to the chapter, our choice of CFD model has been influenced by the fact that it will be used in the context of optimization.

The model is calibrated and validated by comparing computed solutions, obtained using Fluent, to experimental data by Bansod and Bradshaw [3]. In their experiment, the fluid was air. The Reynolds number was $5 \cdot 10^5$, compared to about $5 \cdot 10^6$ for the s-duct water flow considered in this thesis. Still, it is hoped that the result of a validation against the Bansod and Bradshaw experiment can be carried over to our waterjet model. The investigated parameters of the CFD model are

- grid topology,
- turbulence model (Standard or RNG k-ϵ),
- turbulence boundary conditions.

We stress that the purpose of this section has not been to fully evaluate the CFD model that has been used, but rather to improve its performance by making appropriate choices of the parameters given above.

The generic flow in s-ducts is discussed in Section 4.4.1. The experimental setup of Bansod and Bradshaw is described in Section 4.4.2. The details of the computational simulation of the experimental setup in Fluent are found in Section 4.4.3. A comparison of the computed solutions to experimental data is given in Section 4.4.4. The result of the validation is discussed in Section 4.4.5, and our choice of CFD model for the waterjet inlet duct flow is given in Section 4.4.6.

4.4.1 Flow through s-shaped ducts

The flow through s-ducts is characterized by a complex interaction of pressure gradients, streamline curvature and secondary flows. The following description has been taken largely from [34] and [3].

As the flow enters the first bend it is deflected by a cross-stream pressure gradient which gives higher static pressure at the outside of the bend than at the inside. The inviscid core flow takes time to adjust to the curvature of the bend which result in a cross-flow in the core region from the inside to the outside of the bend. In combination with a flow of boundary layer fluid towards the inside of the bend caused by the cross-stream pressure gradient, this yields a secondary flow consisting of a pair of counter-rotating vortices that convects low-momentum fluid to the inside of the bend. Assisted by the adverse pressure gradient at the inside of the second half of the bend, this gives rise to a rapid increase in boundary layer thickness at the inside towards the end of the bend.

The build-up of low momentum fluid at the inside of the first bend displaces the core flow towards the inside of the second bend. Of course the cross-stream pressure gradient reverses as the flow enters the second bend and it tends to convect boundary layer fluid towards the inside of that bend. However, the vortices built up in the first bend take time to reverse, and they still cause boundary layer fluid to convect towards what is now the outside of the second bend. Therefore, the cross-stream pressure gradient will only affect the boundary layer fluid in the core-region which has been displaced towards the inside of the bend. In combination with the cross-stream flow in the core region (similar to what happened in the first bend) toward the outside of the bend, another two counter-rotating vortices appear that convects boundary layer fluid towards the inside of the bend. The result at the duct exit is a characteristic region of low total pressure and velocity at the outside of the second bend and a secondary flow pattern consisting of two pairs of counter-rotating vertices.

4.4.2 Experimental configuration

The s-duct geometry called C3 in the paper by Bansod and Bradshaw [3] has a diameter D of 0.15 m and consists of two 45^o bends. The upstream bend has $R/D = 2.25$ and the downstream bend has $R/D = 3.5$, where R denotes the radius of the centreline. The bends are separated by a straight duct section of length $0.5D$ and after the downstream bend follows a straight duct section of length $0.5D$. Air of velocity 45 m/s is blowed into the inlet. The Reynolds number of the flow based on the duct diameter is approximately $5 \cdot 10^5$.

4.4.3 Computational simulation

The computational flow domain is shown in Fig. 4.6. It consists of the s-duct of Bansod and Bradshaw (referred to as the physical s-duct or flow domain) extended by two straight duct sections of length D, one placed upstream of the duct and one downstream. We will refer to certain cross-sections of the computational domain as follows.

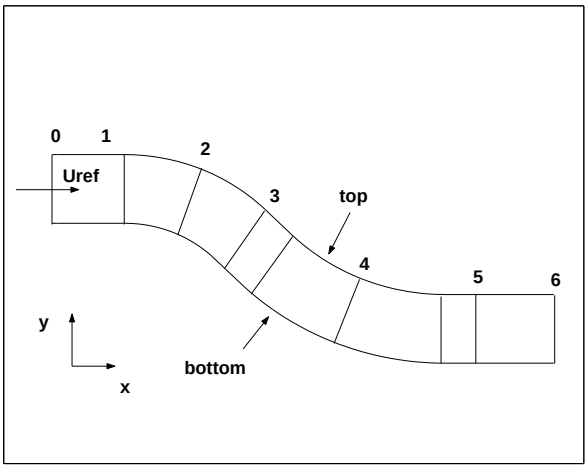


Figure 4.6: Computational flow domain for the s-duct of Bansod and Bradshaw. The physical flow domain corresponds to the part between station 1 and 5.

Station 0	the inlet of the computational flow domain
Station 1	the cross-section at the entrance of the upstream bend, which is also the inlet of the physical s-duct
Station 5	the cross-section 0.5D downstream of the exit of the downstream bend, which is also the outlet of the physical s-duct
Station 6	the outlet of the computational flow domain

In all computations, an inlet velocity of 45 m/s was specified normal to the inlet boundary and the fluid was specified as air with the density 1.293 kg/m^3 and the molecular viscosity $\mu = 1.7455 \cdot 10^{-5}$ kg/ms, giving the experimental Reynolds number of $5 \cdot 10^5$. A Neumann outflow condition (OUTFLOW in Fluent) was specified at the outlet. The turbulence was modelled by the Standard and the RNG k-ϵ models with the standard wall functions.

The 'standard' scheme in Fluent was used for the pressure and the QUICK scheme for the other equations. Either the SIMPLE scheme or the PISO scheme was used for the pressure-velocity coupling. When SIMPLE was used, the under-relaxation factors where 0.2 for the pressure and 0.5 for the momentum, k and ϵ equations. When PISO was used, the under-relaxations factors where 0.3 for the pressure, 0.7 for the momentum equations and 0.8 for the k and ϵ equations.

4.4.4 Computations compared to experimental data

In this section we present the results of five computations done in Fluent and compare them to the experimental data of Bansod and Bradshaw. We note that our computations are comparable in accuracy to those obtained by Seil [34] in a similar validation.

The first two computations were done in order to evaluate the different grid topologies discussed in Section 4.3. Another two computations were performed in order to calibrate the boundary conditions of the k-ϵ model (Section 3.6.2). In the last computation the turbulence model was changed from Standard k-ϵ

Grid	Topology	Parameters	Size	y^+ range
Grid 1	O-grid	(70,15,30,1.1,1.06)	63000	[25,82]
Grid 2	Butterfly	(70,22,16,1.13,0.7)	66220	[25,77]

Table 4.1: Grid data.

to RNG k-ϵ.

Grid evaluation

Two computational grids have been examined. The first grid, called Grid 1, has an O-type topology, and the second, called Grid 2, has a butterfly topology (Section 4.3). The two grids are comparable in size and near-wall resolution, as can be seen from Table 4.1.

One computation was run on Grid 1 and another on Grid 2. In both computations, the boundary conditions for k and ϵ were determined by the turbulence intensity $I = 1\%$ and the turbulence length scale $l = 0.0105$. The value of l was obtained from formula (3.16), using the hydraulic diameter $D_H = 0.15$ m. The results were almost identical. The main difference was found in the convergence behaviour. Fig. 4.8 shows the average total pressure and the mass flow rate at the exit at each iteration during the computation. It can be seen that Grid 2 gives faster convergence than Grid 1. It is also interesting to compare the turbulence velocity scale distributions in Fig. 4.22 and Fig. 4.23. The distribution for Grid 1 has a small cusp at the entrance to the upstream bend. Since this cusp is not present for Grid 2, it might be a consequence of the high degree of skewness in the O-grid wedge elements along the centreline. The skewed wedge elements probably also explains the slower rate of convergence for Grid 1.

Distribution of static pressure

The measured distribution of C_p (defined in Eq. (4.10)) compared to that calculated using the Standard k-ϵ model with $l = 0.0105$ and $I = 1\%$ on Grid 1 and 2, along the top and bottom of the duct symmetry plane is presented in Fig. 4.9. All other computations gave more or less identical results. The overall qualitative agreement is good, but the static pressure was under-predicted at the bottom of the upstream bend, as well as at the duct exit. Also, the static pressure is over-predicted at the bottom of the downstream bend, which, according to Seil [34], is a consequence of the failure of the turbulence model to adequately predict a measurable region of separation/near-separation.

Distribution of total pressure loss

The computed contours of the total pressure loss coefficient C_P at station 5 are presented in Fig. 4.11 to 4.13. The Standard k-ϵ model shows good qualitative agreement with the measured distribution in Fig. 4.14. Most noticeable is the under-prediction of the "tucking-in" of the contours of C_P toward the centreline. This under-prediction is reduced when l is decreased from 0.0105 to 0.0015. The reduction of l corresponds to calculating l from the boundary layer thickness, δ_{99}, using Eq. (3.17), instead of computing it from the hydraulic diameter using

Grid	k-ϵ	l	I	Method 1		Method 2	
				$\bar{C_P}$	Rel. error	$\bar{C_P}$	Rel. error
Grid1	Std	0.0105	1%	0.104	71%	0.088	45%
Grid2	Std	0.0015	1%	0.101	66%	0.085	40%
Grid2	RNG	0.0015	1%	0.098	61%	0.082	35%

Table 4.2: Mass-averaged total pressure loss coefficient at station 5. Measured value is $\bar{C_P} = 0.061$.

Eq. (3.16). The boundary layer thickness was chosen to be $\delta_{99} = 0.025 * 0.15$ m, in accordance with the measurements by Bansod and Bradshaw. The RNG k-ϵ model gives a better tucking-in behaviour, but the overall qualitative agreement is poor.

Fig. 4.15 shows the radial distribution of C_P at the lower half of the duct at station 5. The Standard k-ϵ with $l = 0.0105$ and $I = 1\%$ gives a smeared out curve compared to the experimental data. Better agreement is obtained when l is decreased to 0.0015. The RNG k-ϵ model succeeds well in predicting the distinct local maximum and minimum of the curve, but fails to adequately locate the region of large loss at $y/D \approx 0.23$.

The computed boundary layer as measured by the contour of $C_P = 0.01$ is shown in Fig. 4.16 to Fig. 4.18. Bansod and Bradshaw reported a boundary layer thickness of one duct radius at the bottom of the duct at station 5. Thus the boundary layer thickness at station 5 is under-predicted by the Standard k-ϵ with $l = 0.0105$ and $I = 1\%$, but fairly well predicted when l is decreased to 0.0015. This can also be seen from the distribution of C_P over station 5.

Mass-averaged total pressure loss coefficient

Table 4.2 presents the mass-averaged total pressure loss $\bar{C_P}$ at station 5. Bansod and Bradshaw measured $\bar{C_P} = 0.061$. $\bar{C_P}$ has been calculated with two methods. With Method 1, $\bar{C_P}$ was calculated according to the definition of Bansod and Bradshaw as the mass-average of C_P over station 5, see formula (4.14). With Method 2, $\bar{C_P}$ was calculated from formula (4.15). Since the computational flow domain has been extended compared to the experimental flow domain, the computed velocity is not completely uniform over the cross-section at station 1. Therefore, as discussed in Section 4.5.2, Method 1 and 2 give different values. Method 2 gives a value closer to the measurement, perhaps because its physical meaning is the one intended by Bansod and Bradshaw.

It can be seen from the Table that the RNG k-ϵ model gives a value of $\bar{C_P}$ closer to the measurement than does the Standard k-ϵ model. The ability of the latter model to predict $\bar{C_P}$ is slightly increased by reducing l from 0.0105 to 0.0015.

In an attempt to further improve the agreement between computed and measured $\bar{C_P}$ and distributions of C_P, one computation was done using the Standard k-ϵ model with $l = 0.0015$ and I decreased to 0.1%. However, the result was identical to that using $I = 1\%$.

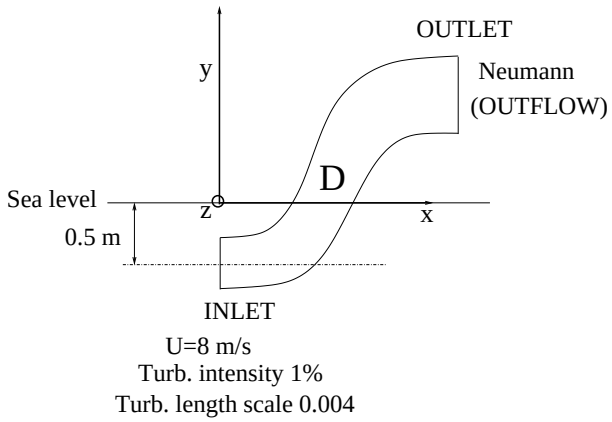


Figure 4.7: Boundary conditions in CFD model.

Secondary flow

All computations gave the same kind of secondary flow behaviour, as can be seen in Fig. 4.19 to 4.21. A strong streamwise vortex is predicted near the duct symmetry plane. A second, weaker vortex smeared out along the wall is also predicted. The computed secondary flow behaviour is physically realistic, as discussed by Bansod and Bradshaw [3] and Seil [34]. It is the first-mentioned strong vortex that causes the tucking-in of the contours of C_P, as well as the boundary layer at station 5 to be much thicker than for a straight duct.

Decreasing the turbulence length scale l from 0.0105 to 0.0015 results in a stronger vortex, as can be seen by comparing Fig. 4.19 with Fig. 4.20. Also, the RNG k-ϵ model predicts a stronger vortex than does the Standard k-ϵ.

4.4.5 Discussion of the validation

Grid 1 (O-type topology) and Grid 2 (butterfly topology) was found to give more or less identical solutions, but with a faster rate of convergence for Grid 2.

All investigated combinations of grids, turbulence models and turbulence boundary conditions gave almost identical distributions of static pressure.

It was found that the performance of the k-ϵ model could be improved by calibrating the turbulence boundary conditions. The best conditions found were given by $I = 1\%$ and $l = 0.0015$. The latter value was obtained from the approximate formula (3.17), with the boundary layer thickness at the entry plane taken in accordance with experimental data to be $\delta_{99} = 0.025 * 0.15$.

The RNG k-ϵ model was found to more accurately predict the loss over the duct than the Standard k-ϵ model. However, the latter model showed slightly better qualitative agreement between measured and computed contours of the total pressure loss at the duct exit. Furthermore, industrial experience tells that the Standard model is more robust than the RNG model.

4.4.6 Choice of CFD model for waterjet inlet duct flow

For the optimization-related turbulent s-duct flow calculations presented later, we will use the following CFD model in Fluent. The preceding validation and discussion lends credibility to the accuracy and robustness of the model.

The purpose of the CFD model is to numerically solve the Reynolds equations (3.11). We have not included gravitation in the equations since it will be cancelled out by the hydrostatic pressure. Instead, gravitation is taken into account in the model *after* the CFD analysis, by adding the hydrostatic pressure to the static pressure reported by Fluent. The sea level has been set 0.5 m above the center of the inlet opening.

We use a butterfly grid to discretize the flow domain. The turbulence is modelled by the Standard k-ϵ model. By default in Fluent, the turbulent quantities k and ϵ are subject to a Neumann condition at the outlet. At the inlet, we specify a uniform boundary condition by the turbulence intensity I and the turbulence length scale l. We take $I = 1\%$ and calculate l according to formula (3.17). If the boundary layer thickness at the inlet is 2.5% of the inlet duct diameter D_{in}, and $D_{in} = 0.4$ m, then $l = 0.004$. The inlet velocity is set to 8 m/s and a Neumann (OUTFLOW) condition is specified at the outlet; see Fig. 4.7. The reference pressure location has been set to the center of the inlet opening, and the pressure p reported by Fluent will always be zero there.

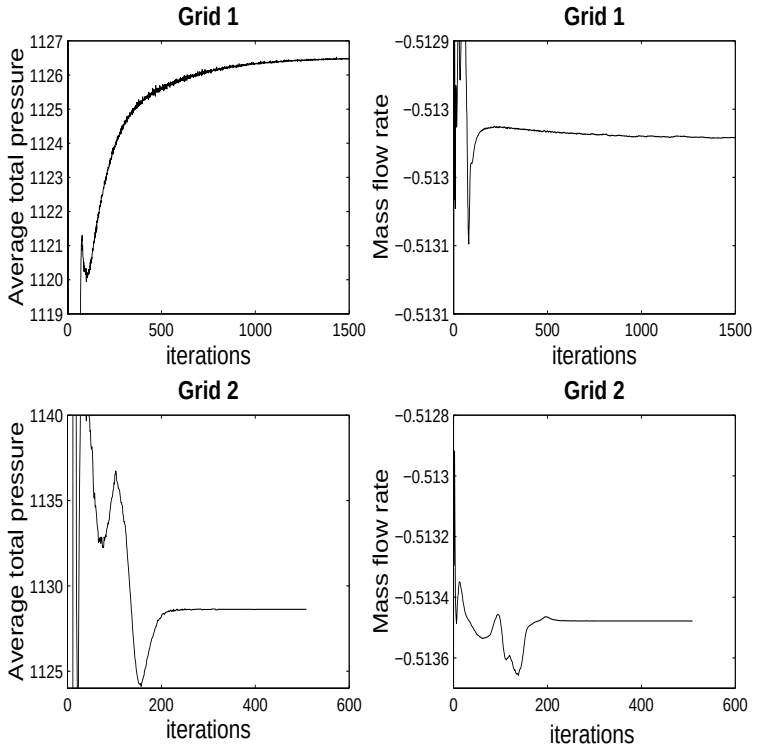


Figure 4.8: Solution history at station 6 (exit). Standard k-ϵ, $l = 0.0105$ and $I = 1\%$.

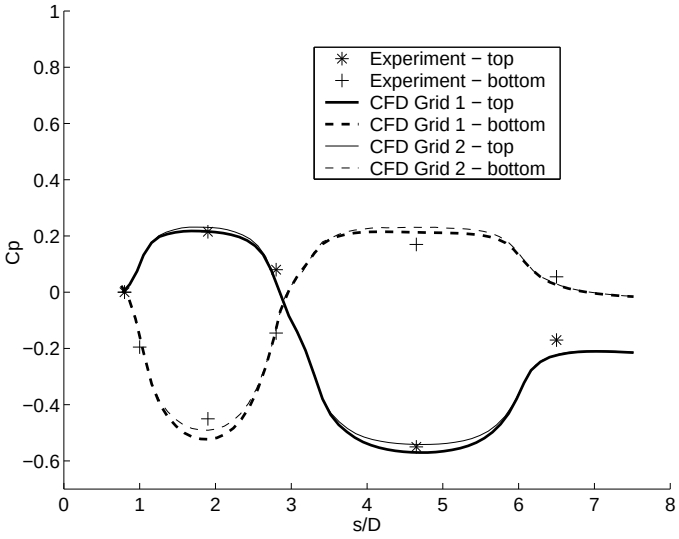


Figure 4.9: Static pressure distribution along the top and the bottom of the duct center plane. Standard k-ϵ, $l = 0.0105$ and $I = 1\%$.

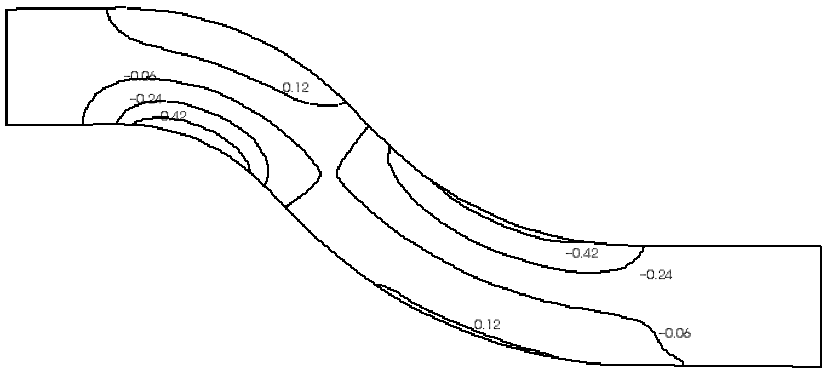


Figure 4.10: Distribution of C_p over duct symmetry plane.

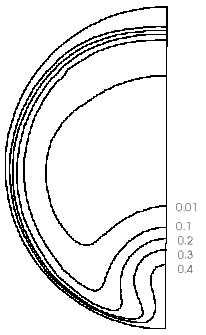


Figure 4.11: Contours of C_P at station 5. Grid 2, Standard k-ϵ, $l = 0.0105$ and $I = 1\%$.

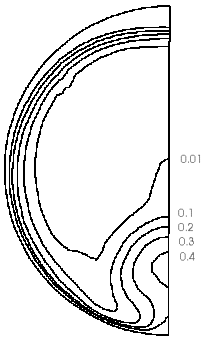


Figure 4.12: Contours of C_P at station 5. Grid 2, Standard k-ϵ, $l = 0.0015$ and $I = 1\%$.

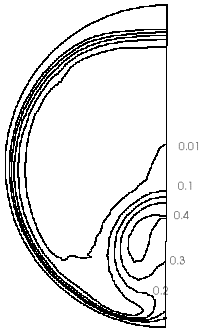


Figure 4.13: Contours of C_P at station 5. Grid 2, RNG k-ϵ, $l = 0.0015$ and $I = 1\%$.

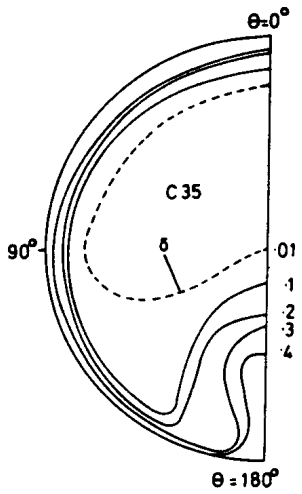


Figure 4.14: Contours of C_P at station 5. Experimental measurements from [3].

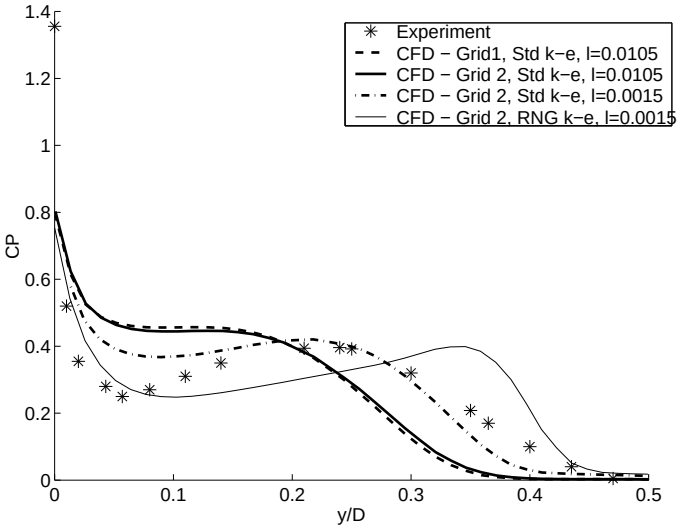


Figure 4.15: Radial distribution of C_P at the bottom of station 5.

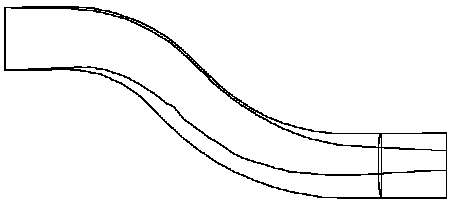


Figure 4.16: Boundary layer contour ($C_P = 0.01$). Grid 2, Standard k-ϵ, $l = 0.0105$ and $I = 1\%$.

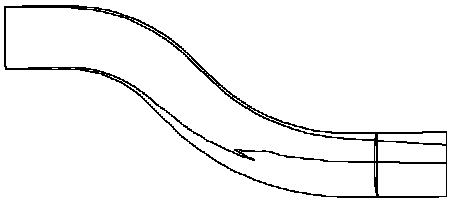


Figure 4.17: Boundary layer contour ($C_P = 0.01$). Grid 2, Standard k-ϵ, $l = 0.0015$ and $I = 1\%$.

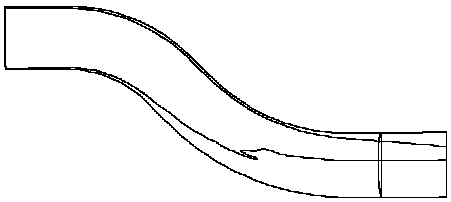


Figure 4.18: Boundary layer contour ($C_P = 0.01$). Grid 2, RNG k-ϵ, $l = 0.0015$ and $I = 1\%$.

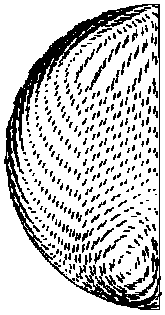


Figure 4.19: Secondary flow at station 5. Grid 2, Standard k-ϵ, $l = 0.0105$ and $I = 1\%$.

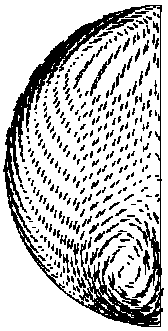


Figure 4.20: Secondary flow at station 5. Grid 2, Standard k-ϵ, $l = 0.0015$ and $I = 1\%$.

Figure 4.21: Secondary flow at station 5. Grid 2, RNG k-ϵ, $l = 0.0015$ and $I = 1\%$.

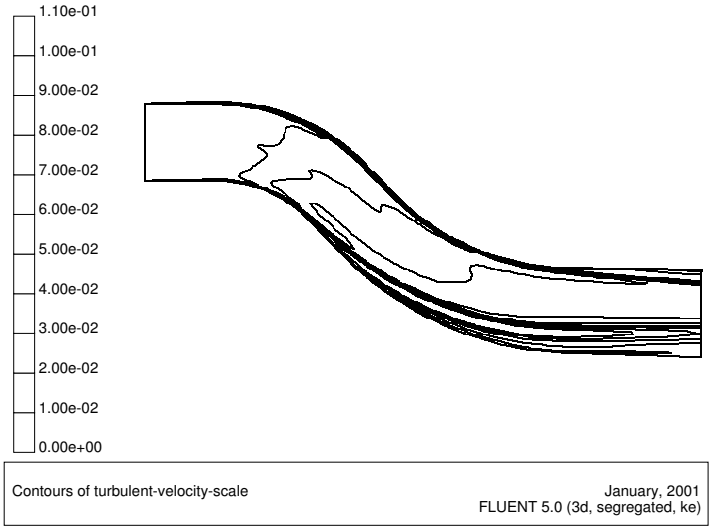


Figure 4.22: Turbulent velocity-scale ($\sqrt{k}/U_{ref}$) on the duct center plane. Grid 1, Standard k-ϵ, $l = 0.0105$ and $I = 1\%$..

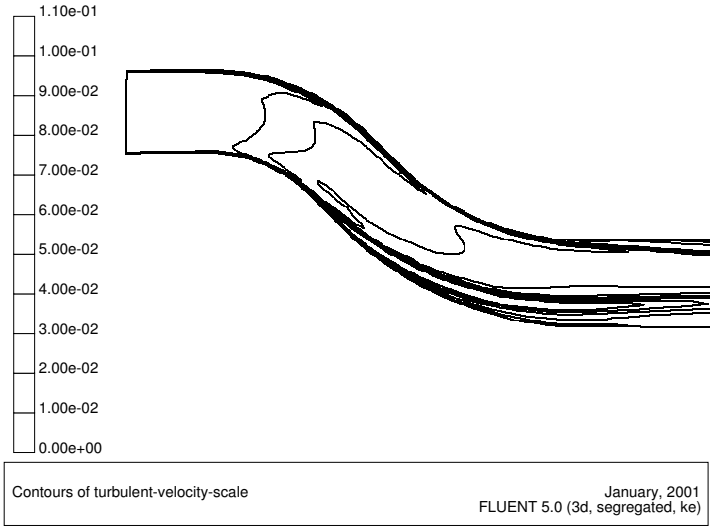


Figure 4.23: Turbulent velocity-scale ($\sqrt{k}/U_{ref}$) on the duct center plane. Grid 2, Standard k-ϵ, $l = 0.0105$ and $I = 1\%$..

4.5 Objective functions

We present the objective functions that will be used in the optimization of the waterjet inlet duct geometry. The purpose of an objective function in this context is to assess the hydrodynamic performance of a given waterjet inlet s-duct geometry. The parameterization in Eq. (4.1) allows us to regard the objective function as a mapping

$$f : \Omega \subseteq \mathbb{R}^5 \to \mathbb{R}, \qquad f = f(\mathbf{p}), \qquad \mathbf{p} = (\alpha, R_1, R_2, r_1, r_2).$$

We consider two aspects of the hydrodynamic performance of the s-duct. The first is the cavitation performance and the second is the total pressure loss. These aspects of the flow are now explained in greater detail. Throughout the discussion we assume a stationary, incompressible viscous flow in a gravitational field.

4.5.1 Cavitation performance

Cavitation is said to occur if the static pressure (including hydrostatic pressure) is below the vapour pressure at some location in the duct. The inception of cavitation in a waterjet inlet will increase losses and may cause erosion, and must therefore be avoided. We proceed to define a measure of the risk of cavitation.

Let p denote the static pressure *without* the hydrostatic pressure, and p^* the static pressure *with* the hydrostatic pressure. Hence,

$$p^* = p - \rho g y, \tag{4.9}$$

where ρ is the fluid density, g the gravitational constant and y the height above the sea level, Fig. 4.24. We split up the pressure like this for practical reasons. Firstly, the hydrostatic pressure should be taken into account in the objective function presented in this section, but not in the one presented in the next. Secondly, as discussed in Section 4.4.6, p is the quantity reported by Fluent and gravitational effects are taken into account by computing p^* *after* the flow calculation.

In hydrodynamics it is common to scale the flow quantities. This facilitates interpretation of flow data. The pressure is often represented by the *static pressure coefficient*,

$$C_p(\mathbf{x}) = \frac{p^*(\mathbf{x}) - p^*_{ref}}{\frac{1}{2}\rho U^2_{ref}}, \tag{4.10}$$

where U the velocity magnitude and index ref refers to some reference location. For the s-duct, we have chosen the reference location to be the center of the inlet opening (Fig. 4.24). Thus, at the center of the inlet opening, we always have $C_p = 0$.

The risk of inception of cavitation in a flow domain D is measured by the minimum static pressure coefficient,

$$C_{p,min} = \min_{\mathbf{x} \in D} C_p(\mathbf{x}). \tag{4.11}$$

Since $C_p = 0$ at the reference location, we have

$$C_{p,min} \leq 0.$$

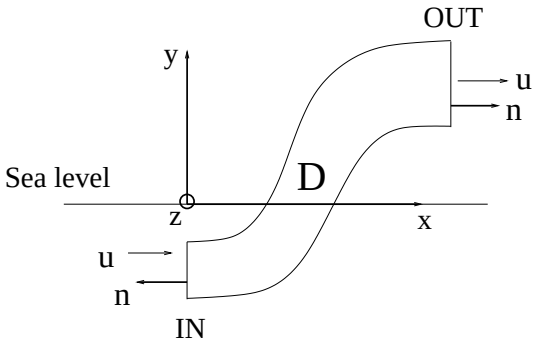


Figure 4.24: S-duct flow domain geometry.

When D is the waterjet inlet s-duct parameterized by Eq. (4.8), $C_{p,min}$ depends on the geometry parameter vector $\mathbf{p}$. In order to find the parameterized geometry that minimizes the risk of cavitation, $C_{p,min}(\mathbf{p})$ should be maximized. This is equivalent to minimizing $-C_{p,min}(\mathbf{p})$. Since all optimization problems in this work are formulated as minimization problems, we take the objective function to be

$$f(\mathbf{p}) = -C_{p,min}(\mathbf{p}).$$

Since $C_{p,min} \leq 0$, we have

$$f(\mathbf{p}) \geq 0, \quad \forall \mathbf{p}.$$

4.5.2 Total pressure loss

All physical flows experience a loss of energy due to viscous effects. In their study of the flow in s-shaped ducts with circular cross section of constant radius, Bansod and Bradshaw [3] recommended the losses to be measure by the *mass-averaged total pressure loss coefficient*

$$\bar{C_P} = \frac{1}{A} \int_{OUT} \frac{\rho U}{\rho_{ref} U_{ref}} C_P \, dS,$$

where ref refers to the condition at the inlet, A is the cross-sectional area, OUT the outlet surface, dS the surface measure and

$$C_P = \frac{P_{ref} - P}{\frac{1}{2}\rho_{ref} U_{ref}^2} \tag{4.12}$$

the *total pressure loss coefficient.* P is the total pressure,

$$P = p + \frac{1}{2}\rho U^2. \tag{4.13}$$

We only consider incompressible water flow in this work, and hence the density is constant. For constant flow conditions over the inlet surface IN (Fig. 4.24),

the above definition of $\bar{C_P}$ is equivalent to

$$\bar{C_P} = \int_{OUT} C_P \frac{d\dot{m}}{\dot{m}}, \tag{4.14}$$

where $d\dot{m} = \rho \mathbf{u} \bullet \mathbf{n} dS$ is the mass flow rate measure, $\mathbf{u}$ the flow velocity, $\mathbf{n}$ the outward normal to surface OUT (Fig. 4.24) and $\dot{m} = \int_{IN} d\dot{m}$ the mass flow rate. Formula (4.14) explains the name of $\bar{C_P}$. Furthermore, it is valid also for ducts with varying cross-sectional radius. When the flow domain is parameterized by Eq. (4.8), $\bar{C_P}$ becomes dependent on the parameter vector $\mathbf{p}$, and since the losses should be minimized, we take the objective function to be

$$f(\mathbf{p}) = \bar{C_P}(\mathbf{p}).$$

As a more general measure of loss in a stationary, incompressible flow in an s-duct like the one in Fig. 4.24 we suggest

$$\bar{C_P} = \frac{-\int_{IN} P\mathbf{u} \bullet \mathbf{n}\, dS - \int_{OUT} P\mathbf{u} \bullet \mathbf{n}\, dS}{\int_{IN} \frac{1}{2}\rho U^2 \mathbf{u} \bullet \mathbf{n}\, dS.} \tag{4.15}$$

Note that this formula is equivalent to (4.14) for constant inlet conditions, and that $\mathbf{u} \bullet \mathbf{n}$ is positive on the outlet and negative on the inlet.

We now motivate that $\bar{C_P}$ is a measure of losses and that the last formula can be read as "the rate of viscous loss over the duct between surfaces IN and OUT, normalized by the inflow of kinetic energy through surface IN". To this end, let V_t denote a volume following the fluid (Section 3.2). Let D/Dt be the total, or material, derivative,

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + (\mathbf{u} \bullet \nabla)\mathbf{u}.$$

Consider the kinetic energy in V_t,

$$E(t) = \int_{V_t} \frac{1}{2}\rho \|\mathbf{u}\|^2 dV.$$

By the transport theorem, the rate of change of kinetic energy in V_t equals

$$\frac{dE(t)}{dt} = \int_{V_t} \rho \frac{D}{Dt}\Big(\frac{\|\mathbf{u}\|^2}{2}\Big) dV.$$

We have

$$\frac{D}{Dt}\Big(\frac{\|\mathbf{u}\|^2}{2}\Big) = \frac{D\mathbf{u}}{Dt} \bullet \mathbf{u}.$$

Thus, by the Navier-Stokes equations for a flow in a gravitational field $\mathbf{g}$,

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p^* + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g},$$

we get

$$\int_{V_t} \rho \frac{D\mathbf{u}}{Dt} \bullet \mathbf{u} = \int_{V_t} (-\nabla p^* + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g}) \bullet \mathbf{u}\, dV.$$

The left-hand side is the rate of change of kinetic energy. The terms on the right-hand side represent, respectively, the rate of work done by the pressure field on the fluid, the rate of work done by viscous stresses and the rate of work done by the gravitational field. We now fix t and take $V_t = V$ to be the entire s-duct flow domain. Let

$$\phi_g = gy$$

be the gravitational potential so that $\mathbf{g} = -\nabla\phi_g$, and note that $D\mathbf{u}/Dt \bullet \mathbf{u} = \nabla(\frac{1}{2}\|\mathbf{u}\|^2)$ for a stationary flow. Then we get, after re-arranging the terms, that

$$-\int_V \mu\nabla^2\mathbf{u} \bullet \mathbf{u} = -\int_V \nabla(p^* + \rho\phi_g + \frac{1}{2}\rho\|\mathbf{u}\|^2) \bullet \mathbf{u}\, dV. \tag{4.16}$$

The sum between the paranthesis on the right-hand side equals P (see Eq. (4.9) and (4.13)). Since we consider an incompressible flow, we have

$$\nabla P \bullet \mathbf{u} = \nabla \bullet (P\mathbf{u}).$$

If we put this into Eq. (4.16) and apply Gauss' Theorem, we obtain

$$-\mu\int_V \nabla^2\mathbf{u} \bullet \mathbf{u} = -\int_S P\mathbf{u} \bullet n.$$

The s-duct surface S consists of the wall W, the inlet IN and the outlet OUT. Since $\mathbf{u} \bullet \mathbf{n} = 0$ on W, we get

$$-\int_V \mu\nabla^2\mathbf{u} \bullet \mathbf{u} = -\int_{IN} P\mathbf{u} \bullet n - \int_{OUT} P\mathbf{u} \bullet n.$$

This is the nominator in $\bar{C_P}$. As mentioned above, the right-hand side is the negative rate of work done on the fluid by viscous stresses, i.e., the viscous losses. This explains the physical meaning of formula (4.15).

It is interesting to note that not all viscous work necessarily dissipates. By Green's formula we have

$$-\mu\int_V \nabla^2\mathbf{u} \bullet \mathbf{u} = \mu\int_V \|\nabla u_i\|^2 dV - \mu\int_S u_i\frac{\partial u_i}{\partial n}dS,$$

where summation overe repeated indices is understood. The first term on the right-hand side is evidently always positive. It corresponds to viscous dissipation of energy due to deformation of fluid elements ([1], p. 216). We now consider the second term on the right-hand side. Again, S consists of wall, inlet and outlet. u_i is zero on the wall, so

$$\mu\int_S u_i\frac{\partial u_i}{\partial n}dS = \mu\int_{IN} u_i\frac{\partial u_i}{\partial n}dS + \mu\int_{OUT} u_i\frac{\partial u_i}{\partial n}dS.$$

This expression should be as large as possible in order to minimize the losses. (Note, however, that it is correlated to the deformation term, since both contain the derivative of u_i.) It is zero if the flow is fully developed ($\partial u_i/\partial n = 0$) at the inlet and the outlet. In this thesis we use the outlet condition $\partial u_i/\partial n = 0$ (Section 4.4.6), which implies

$$\mu\int_S u_i\frac{\partial u_i}{\partial n}dS = \mu\int_{IN} u_i\frac{\partial u_i}{\partial n}dS.$$

The size of this term only depends on the condition at the inlet. If the inlet velocity is normal to the inlet surface (Fig. 4.24), then

$$\mu \int_S u_i \frac{\partial u_i}{\partial n} dS = -\mu \int_{IN} u \frac{\partial u}{\partial x} dS,$$

where u is the velocity in the x-direction, which is the direction normal to the inlet. The term on the left-hand side should be positive to reduce losses. Since u is positive, this is the case if $\partial u/\partial x$ is negative. Hence the flow should diffuse (streamlines should expand) into the inlet opening in order to reduce the loss. However, whether this contribution has any practical significance or not we do not know.

To sum up, we have motivated why $\bar{C_P}$ is a measure of energy losses due to viscous effects. We found that the losses are affected not only by the dissipation inside the duct, but also by the conditions at the inlet. The impact of these conditions are neglected in our work because we specify uniform inlet conditions. This might be a short-comming in our modelling of the waterjet inlet duct performance.

4.6 Summary

We have discussed an implemented model of a waterjet inlet duct. The purpose of the model is to allow for an optimization of the inlet geometry for optimal hydrodynamic performance. The model can be summarized by two mappings D and f, which to each parameteric representation $\mathbf{p}$ of the inlet geometry associates a duct $D(\mathbf{p})$ and a real number $f(\mathbf{p})$ that measures the hydrodynamic performance.

The inlet geometry was modelled by an s-duct having circular cross sections of varying diameter. We presented a simple parameterization of the inlet, consisting of five parameters wich will also be the variables in the optimization in the following chapter.

A CFD model in Fluent for the flow through the inlet was calibrated and validated against experimental data. We decided to use the Standard k-ϵ turbulence model.

We have discussed two measures of the hydrodynamic performance of the inlet. The first was the minimum static pressure coefficient, which measures the risk of cavitation in the duct. The other was the mass-averaged total pressure coefficient, which we showed to be a measure of the rate of viscous loss over the duct. We found the losses to be affected not only by the the dissipation of energy inside the duct, but also by the conditions at the inlet opening. The latter conditions are not properly represented by the model since we specify a uniform inlet velocity condition.

Chapter 5

Optimization of the waterjet inlet duct performance model

5.1 Introduction

The waterjet inlet model developed in the previous chapter was summarized by the parameterized geometry $D(\mathbf{p})$ and a hydrodynamic performance measure $f(\mathbf{p})$. In this chapter we consider the minimization of f.

The optimization problem and its properties are discussed in Section 5.2. In Section 5.3, we present our choice of optimization algorithm. Section 5.4 contains suggestions for the interaction between the optimization and the CFD modelling. The numercial results from five optimization runs are presented and analyzed in Section 5.5, and further discussed in Section 5.6. We summarize and conclude in Section 5.7 and give suggestions for improvements and future research in Section 5.8.

5.2 The Problem

We now consider the optimization of the waterjet inlet model for maximal static pressure and for minimal loss. This can be written as

$$\min_{\mathbf{p}\in\Omega} -C_{p,min} \tag{5.1}$$

and

$$\min_{\mathbf{p}\in\Omega} \bar{C_P}, \tag{5.2}$$

where $\mathbf{p} = (\alpha, R_1, R_2, r_1, r_2)$ is the parametric representation of the flow domain D as described in Section 4.2.2. Ω is the set of all $\mathbf{p}$ that fulfil the non-linear constraints also discussed in Section 4.2.2. The objective functions were described in Section 4.5.

The two problems are characterized mainly by the properties of the objective functions. In order to evaluate any of them at a point $\mathbf{p}$ we must solve a

three-dimensional turbulent flow problem on the domain $D(\mathbf{p})$. In practice, this is done by performing a CFD analysis of the flow. A CFD analysis is a complicated and (most often a) time-consuming numerical process. As a consequence, the objective functions will inherit all three of the properties listed in the introduction to Chapter 2.

Not much can be said *a priori* about the regularity of the objective functions. We note that since $-C_{p,min}$ is obtained by taking the minimum over the flow domain, we can not expect it to be smoother than C^0, even if the flow solution should vary smoothely with the parameter vector $\mathbf{p}$. This is because, loosely speaking, the location in the duct where the minimum static pressure is attained may "jump" from, for example, the upstream to the downstream bend.

5.3 Optimization algorithm

We have solved the above problems with the Sherif-Boice (SBA) algorithm described in Section 2.4.2. Our choice of algorithm is based on the properties of the objective functions mentioned in the preceding section (see the introduction to Chapter 2 for a discussion). The SBA is simple and easy to understand. It is a true direct search method in the sense that it relies only on rank order information about the objective function and use neither explicit nor numerical estimates of the derivatives. It proceeds by taking steps in the coordinate directions, evaluating the objective function and reducing the step length when necessary.

Being a pattern search method, the SBA has nice convergence properties under certain circumstances. The convergence analysis of pattern search methods for unconstrained problems required the objective function to be at least C^1 (Theorem 2). The result could be extended to bound constrained (which is a special case of linearly constrained) problems, as discussed in Section 2.3.3. However, our optimization problems are subject to non-linear constraints, and we can not guarantee that $-C_{p,min}$ has higher regularity than C^0. Therefore, the theoretical robustness of the method when applied to our problem can not be guaranteed, and we must carefully observe how the optimization proceeds in practice.

Let $p_i, i = 1, \ldots, 5$ be the coordinates of $\mathbf{p}$ ($p_1 = \alpha$, etc). The steps Δp_i taken by the SBA should be of different sizes in different directions. This could be taken care of by the algorithm, but a better solution is to change the variables to $\tilde{p_i} = p_i/\Delta p_i$. This is called a scaling and corresponds to a multiplication $\tilde{\mathbf{p}} = D\mathbf{p}$ with the diagonal matrix $D = diag(1/\Delta p_i)$. A scaling will transform the constraints Ω to $\tilde{\Omega}$ and change the objective function to $\tilde{f}(\tilde{\mathbf{p}}) = f(D^{-1}\tilde{\mathbf{p}})$. In the following, we will simply refer to f, Ω and $\mathbf{p}$, assuming that the scaling problem has been solved. Δ will denote the step size, equal in all directions,

The optimization was terminated when $\Delta \leq \Delta_{min}$. Because of the high cost of each function evaluation, another appropriate stopping criterion could be the maximal number of function evaluations. In the following, the number of evaluations will also be referred to as "the number of iterations".

5.4 CFD and optimization

As pointed out already, the evaluation of the objective functions is very time-consuming since it depends on the result of a CFD analysis. To avoid that the optimization, which requires a large number of function evaluations, takes too long, we would like to reduce to a minimum the time spent on each flow calculation. The parameters that affect the amount of time required for each CFD analysis include

1. the choice of turbulence model,
2. the choice of boundary conditions,
3. the grid size,
4. the termination criterion for each computation in Fluent,
5. the quality of the initial guess for the computation in Fluent.

The choices of turbulence model and boundary conditions were done in Section 4.4.6. In the case when a *single* CFD analysis is to be done, 3. and 4. would probably not be a problem. We would simply choose a "fine enough" grid, which perhaps consists of 80000 elements and gives very good resolution of the near-wall flow. The computation in Fluent would be run until the numerical residuals no longer decrease; this would require approximately 1000-1500 iterations. The time spent in Fluent would be several hours on a SUN ULTRA 1 workstation. One optimization (100-150 evaluations) under these conditions would take 3-4 weeks.

We now discuss how we have used, and how one *could* use, the three parameters 3.-5. to speed up the optimization process. We remark that here we are only interested in how these parameters influence the objective functions and the time it takes to evaluate it, and not how they influence all the other aspects of the flow. Throughout the discussion f will denote any of the two objective functions.

5.4.1 Surrogates from coarse grid computations

We first investigate the possibility of constructing surrogate functions of the second type (Section 2.6) by performing the flow calculations on a coarse grid. We would like the grid to be as coarse as possible in order to minimize the time required for each evaluation of f. The trade-off, of course, is that if the mesh is too coarse, the values of f will be meaningless.

The flow in the two different s-ducts shown in Fig. 5.1 and 5.2 has been calculated in Fluent using six different grids. Duct 1 is determined by the parameter vector $\mathbf{p}_1 = (45, 1, 1.4, 0.22, 0.25)$ and Duct 2 by $\mathbf{p}_2 = (65, 3, 1, 0.305, 0.32)$. Data about the different grids is found in Table 5.1. The grid sizes ranges from almost 80000 elements (very fine mesh) down to 3150 (very coarse mesh). Table 5.2 presents the values of the objective functions $\bar{C_P}$ and $-C_{p,min}$ for both geometries and for each grid.

All grids were fine enough to capture the main features of the flow, such as the secondary flow pattern at the exit.

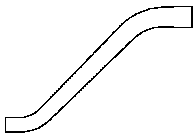


Figure 5.1: Duct 1.

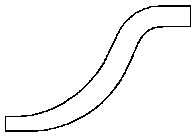


Figure 5.2: Duct 2.

For Duct 1, the coarsest grid (Grid 6) gives a value of $\bar{C_P}$ which is 18% higher than that given by the finest grid (Grid 1). For Duct 2, the corresponding difference is 11%. The values of $-C_{p,min}$ calculated using the finest and coarsest grids for Duct 1 and Duct 2 differs by 3% and 2%, respectively. Thus, $\bar{C_p}$ does show some sensitivity to grid size variations. This is not a surprise since the losses mainly occur in the boundary layer, which is badly resolved by the coarse grids as is reflected by the high values of y^+ in Table 5.1. $-C_{p,min}$, on the other hand, shows a remarkable insensitivity to the grid size.

For rank order methods like the SBA it is not the value of f at a certain point that is interesting in itself, but this value relative to values of f at other points. Therefore it is interesting to note that the relative rank of the values of $\bar{C_P}$ and $-C_{p,min}$ for the two ducts are preserved with all grids, i.e.,

$$\bar{C_P}(\mathbf{p}_1) > \bar{C_P}(\mathbf{p}_2) \quad \text{and} \quad -C_{p,min}(\mathbf{p}_1) > -C_{p,min}(\mathbf{p}_2).$$

Hence, the conclusion that Duct 2 has more desirable cavitation and loss properties than has Duct 1 is independent of the grid used.

We have chosen Grid 3 (20790 elements) in Table 5.1 as our "nominal" grid, and in the following when we write $\bar{C_P}$ and $-C_{p,min}$ it is understood that this mesh has been used in the CFD model. This grid is illustrated in Fig. 4.4 and 4.5. We have also used the coarser Grid 6 (3150 elements), and the values of objective functions calculated with this grid will be denoted $\bar{C_P}^{\alpha}$ and $-C^{\alpha}_{p,min}$. These are the surrogate functions: they are faster to evaluate and even though the values of $\bar{C_P}^{\alpha}$ and $-C^{\alpha}_{p,min}$ at a given point differ from the values of $\bar{C_P}^{\alpha}$ and $-C^{\alpha}_{p,min}$ the above discussion shows that the former functions approximate the latter. To obtain some of the numerical results presented later, we have used the surrogates to find good initial guesses for the optimization on the "real" functions.

Grid	Topology	Parameters	Size	y^+ range
1	Butterfly	(70,18,27,1.15,1.15,0.7)	79380	$[15, 70]$
2	Butterfly	(70,22,16,1.13,1.13,0.7)	66220	$[75, 400]$
3	Butterfly	(55,13,12,1.2,1.2,0.7)	20790	$[100, 460]$
4	Butterfly	(50,10,10,1.21,1.21,0.7)	12500	$[200, 670]$
5	Butterfly	(40,8,8,1.21,1.21,0.7)	6400	$[330, 1060]$
6	Butterfly	(35,6,6,1.25,1.25,0.7)	3150	$[430, 1580]$

Table 5.1: Grid data.

	Duct 1		Duct 2	
Grid	C_P	$-C_{p,min}$	C_P	$-C_{p,min}$
1	0.1003	0.5237	0.07099	0.3236
2	0.1025	0.5264	0.07324	0.3254
3	0.1077	0.5336	0.07418	0.3259
4	0.1080	0.5312	0.07499	0.3264
5	0.1170	0.5149	0.07738	0.3289
6	0.1183	0.5089	0.07850	0.3303

Table 5.2: Summary of CFD analysis of Duct 1 and Duct 2 for different grids.

5.4.2 Termination criterion in Fluent

For the CFD analysis to be fully automatic we must have an appropriate termination criterion for the computation in Fluent. Fluent allows the user to choose between termination of the computation when the numerical residuals are below some set treshold, or simply after a fixed number of iterations, n_{it}. We have used the latter method with $n_{it} = 300$, which was the number of iterations required for the flow through Duct 1 in Fig. 5.1 to converge. Different geometries require different number of iterations for the flow solution to converge, and to be on the safe side we should have chosen n_{it} twice as large. However, that would also double the time required for each evaluation of the objective functions. What we have gained in time we have paid for in accuracy, but we think this cost should not bee too high, because it was observed that in general the values of the objective functions converged to an acceptable level long before the numerical residuals were converged.

It should be clear that surrogate functions could be constructed by performing even fewer iterations in Fluent than we have chosen. This possibility is examined in [11] but has not been considered any further in this thesis.

5.4.3 Initial guesses for the CFD analysis

The SBA proceeds by taking small trial steps around a current iterate. From a shape optimization point of view, this corresponds to comparing the hydrodynamic performance of a current geometry with that of slightly perturbed geometries. Since we expect the flow in two almost identical geometries to be very similar, a natural idea is to use the flow solution from the current geometry as an initial guess for the flow calculation in the perturbed geometries. The hope is that it would reduce the time required for each CFD analysis. Indeed, Seil

[34] found that this procedure reduced the time required for each CFD analysis by a factor of 6-10. However, we have used another version of Fluent, and found only very limited positive effects of good initial guesses.

5.5 Results

We present the results from applying the optimization procedure dicussed above to the cavitation objective function $-C_{p,min}$ and the loss objective function $\bar{C_P}$. The analysis of the results, which is started in this section, is continued in Section 5.6. If nothing else is said, the initial step length in each run was set to $\Delta = 1$, and the optimization algorithm was terminated when the step length was reduced below $\Delta_{min} = 0.01$.

5.5.1 Cavitation

We present the results from three optimization runs on the cavitation objective function $-C_{p,min}$.

Run 1. The optimization was started from

$$\mathbf{p}_1^0 = (90, 1, 1, 0.22, 0.25).$$

The algorithm converged ($\Delta_{min} = 0.01$) after 133 function evaluations. The optimal vector was

$$\mathbf{p}_1^* = (65, 3, 1, 0.305, 0.32).$$

The optimization history is shown in Fig. 5.3. $-C_{p,min}$ decreased from 0.5423 to 0.3258.

Run 2. Run 2 was done in two steps. In step I, the surrogate $-C_{p,min}^{\alpha}$ was optimized from the initial vector

$$\mathbf{p}_2^0 = (90, 1, 1, 0.22, 0.25).$$

The algorithm converged ($\Delta_{min} = 0.01$) after 133 evaluations. The optimal vector,

$$\mathbf{p} = (60, 2.8, 1.2, 0.31125, 0.32),$$

was used as the initial vector in step II, where $-C_{p,min}$ was optimized with initial step length 1/8. Step II converged after 104 evaluations. The optimal vector was

$$\mathbf{p}_2^* = (66.25, 2.9, 0.975, 0.3025, 0.315).$$

The optimization history of run 2 is shown in Fig. 5.4. $-C_{p,min}$ decreased from 0.5423 to 0.3275 in step I, and from 0.3275 to the optimal value 0.3260 in step II.

Run 3. In step I, the surrogate $-C_{p,min}^{\alpha}$ was optimized from the initial vector

$$\mathbf{p}_3^0 = (40, 4, 4, 0.32, 0.3325).$$

	Run 1		Run 2		Run 3	
	Initial	Optimal	Initial	Optimal	Initial	Optimal
α	90	65	90	66.25	40	58.59375
R_1	1	3	1	2.9	4	4.05
R_2	1	1	1	0.975	4	1.55
r_1	0.22	0.305	0.22	0.3025	0.32	0.32
r_2	0.25	0.32	0.25	0.315	0.3325	0.3325
$\bar{C_P}$	0.1172	0.07413	0.1172	0.07420	0.0827	0.07324
$-C_{p,min}$	0.5423	0.3258	0.5423	0.3260	0.3422	0.3269

Table 5.3: Results from optimization on the cavitation objective function.

Step I converged after 124 evaluations and the optimal vector,

$$\mathbf{p} = (60, 4, 1.55, 0.32, 0.3325),$$

was used as the initial vector in step II, where $-C_{p,min}$ was optimized with initial step length 1/4. Step II converged after 52 evaluations. The optimum was

$$\mathbf{p}_3^* = (58.59375, 4.05, 1.55, 0.32, 0.3325).$$

The optimization history of run 3 is shown in Fig. 5.5. $-C_{p,min}$ was reduced from 0.3422 to 0.3269. The optimization in step II only contributed to a very small part of this reduction.

The data from the runs are summarized in Table 5.3. The initial and optimal duct geometries from all runs are shown in Fig. 5.6. Details of the radius curves are summarized in Fig. 5.7. Run 1 and run 2 resulted in approximately the same optimum, and in the following we only report the result from run 1. The optimum of run 3 differs significantly from the optimum of the first two runs.

5.5.2 Loss

We present the results from two optimization runs on the total pressure loss objective function $\bar{C_P}$.

Run 4. In step I, the surrogate $\bar{C_P}^{\alpha}$ was optimized from the same initial vector as in run 1. The optimization was terminated when the step length was less than $\Delta_{min} = 0.2$. This happened after 70 evaluations, and the best vector found was

$$\mathbf{p} = (55, 3.4, 1.5, 0.315, 0.315).$$

This vector was the initial vector in step II, where $\bar{C_P}$ was optimized with initial step length 1/4. Step II converged after 120 evaluations and the optimum was

$$\mathbf{p}_4^* = (55.15625, 3.775, 1.275, 0.316875, 0.316875).$$

Fig. 5.8 shows the optimization history from run 4. $\bar{C_P}$ was reduced from 0.1172 to 0.07330 in step I, and from 0.07330 to the optimal value 0.07249 in step II.

	Run 4		Run 5	
	Initial	Optimal	Initial	Optimal
α	90	55.15625	40	54.0625
R_1	1	3.775	4	4.6125
R_2	1	1.275	4	1.3
r_1	0.22	0.316875	0.32	0.3225
r_2	0.25	0.316875	0.3325	0.325
$\bar{C_P}$	0.1172	0.07249	0.0827	0.07246
$-C_{p,min}$	0.5423	0.3266	0.3422	0.3268

Table 5.4: Results from optimization on the loss objective function.

Run 5. In step I, $\bar{C_P}^{\alpha}$ was optimized from the same initial vector as in run 3. The optimization was terminated after 65 evaluations, when the step length was less than 1/2. The best vector found so far,

$$\mathbf{p} = (50, 5, 1.8, 0.32, 0.33),$$

was the initial vector in step II, where $\bar{C_P}$ was optimized with the initial step length 1/2. Step II converged after 135 evaluations, and the optimum was

$$\mathbf{p}_5^* = (54.0625, 4.6125, 1.3, 0.3225, 0.325).$$

Fig. 5.9 shows the optimization history from run 5. $\bar{C_P}$ was reduced from 0.0827 to 0.07350 in step I, and further reduced to the optimal value 0.07246 in step II.

The data from the runs are summarized in Table 5.4. The optimal values of $\bar{C_P}$ for the two runs are almost equal. However, the optimized geometries differ slightly from each other, as can be seen in Fig. 5.10. The optimized radius curves are very close to each other, as illustrated in fig. 5.11.

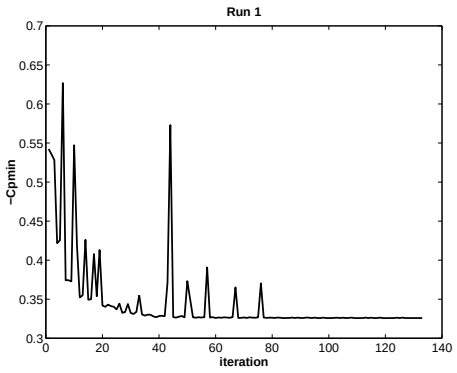


Figure 5.3: Optimization history of Run 1.

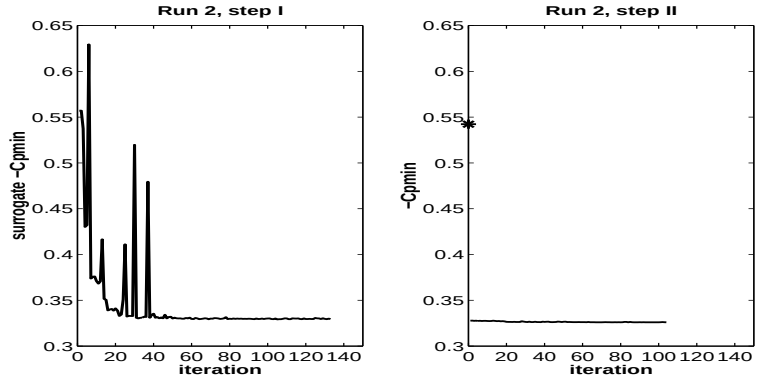


Figure 5.4: Optimization history of Run 2. The surrogate $-C^{\alpha}_{p,min}$ was optimized in step I, and $-C_{p,min}$ in step II. The asterix indicates the value of $-C_{p,min}$ at the initial vector in step I.

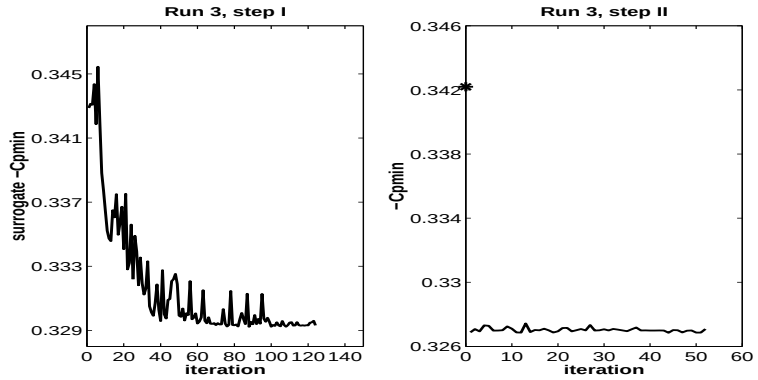


Figure 5.5: Optimization history of Run 3. The surrogate $-C^{\alpha}_{p,min}$ was optimized in step I, and $-C_{p,min}$ in step II. The asterix indicates the value of $-C_{p,min}$ at the initial vector in step I.

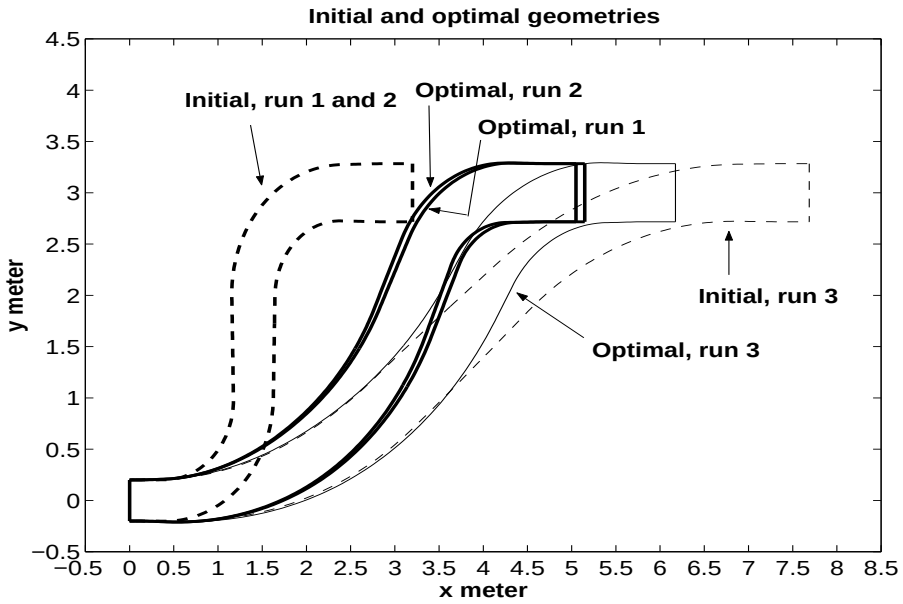


Figure 5.6: Initial and optimized geometries for the cavitation objective function.

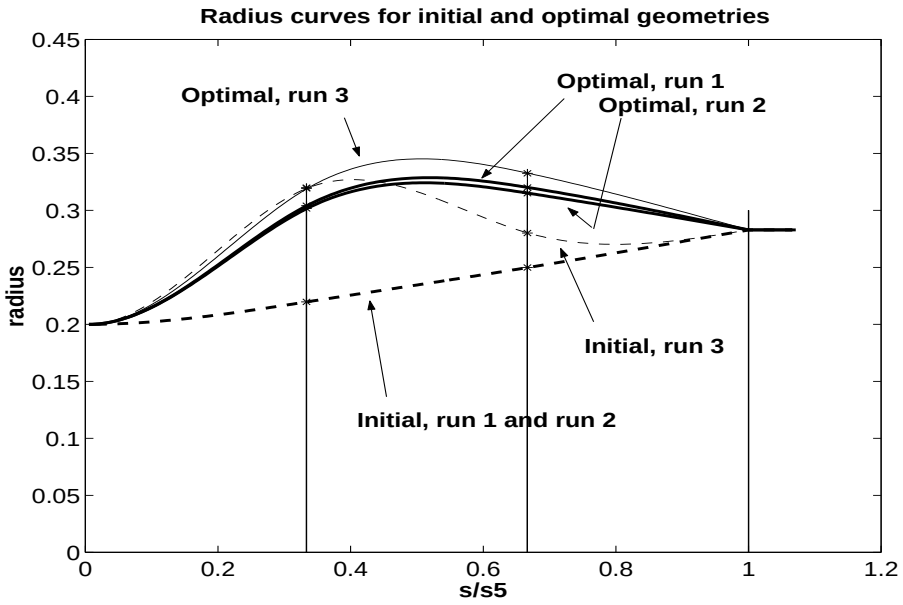


Figure 5.7: Radius curves for initial and optimized ducts with the cavitation objective function.

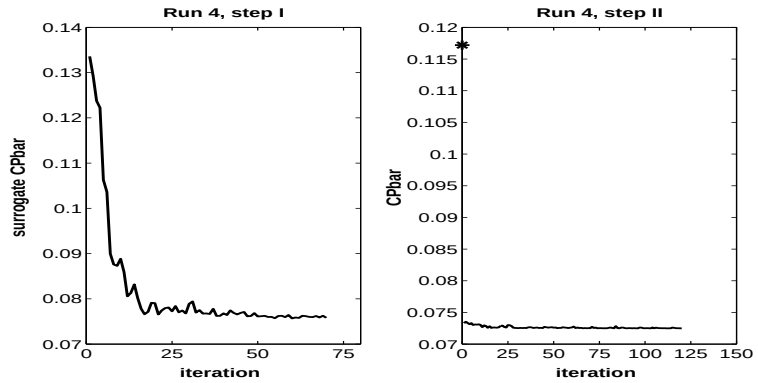


Figure 5.8: Optimization history of Run 4. The surrogate $\bar{C_P}^{\alpha}$ was optimized in step I, and $\bar{C_P}$ in step II. The asterix indicates the value of $\bar{C_P}$ at the initial vector in step I.

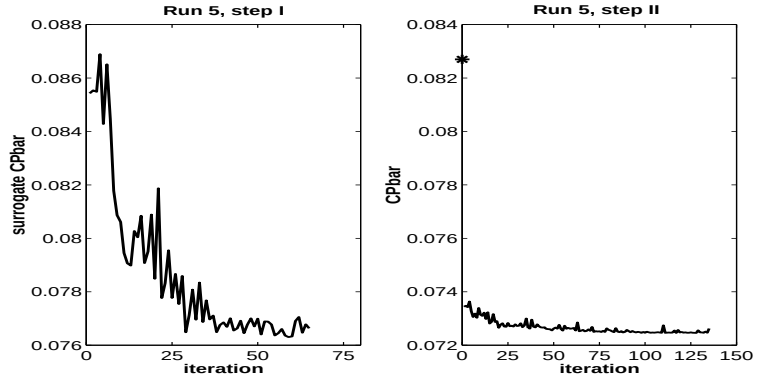


Figure 5.9: Optimization history of Run 5. The surrogate $\bar{C_P}^{\alpha}$ was optimized in step I, and $\bar{C_P}$ in step II. The asterix indicates the value of $\bar{C_P}$ at the initial vector in step I.

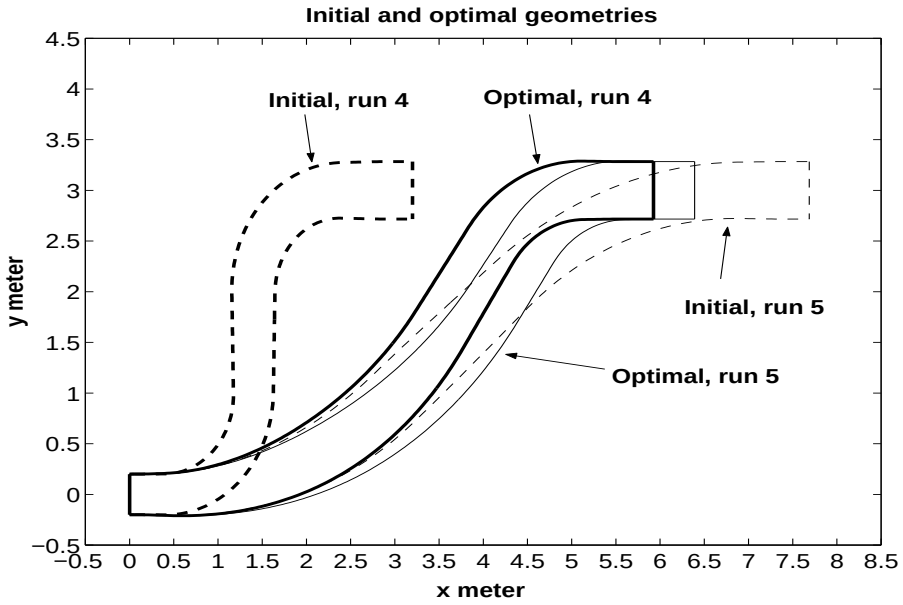


Figure 5.10: Initial and optimized geometries for the loss objective function.

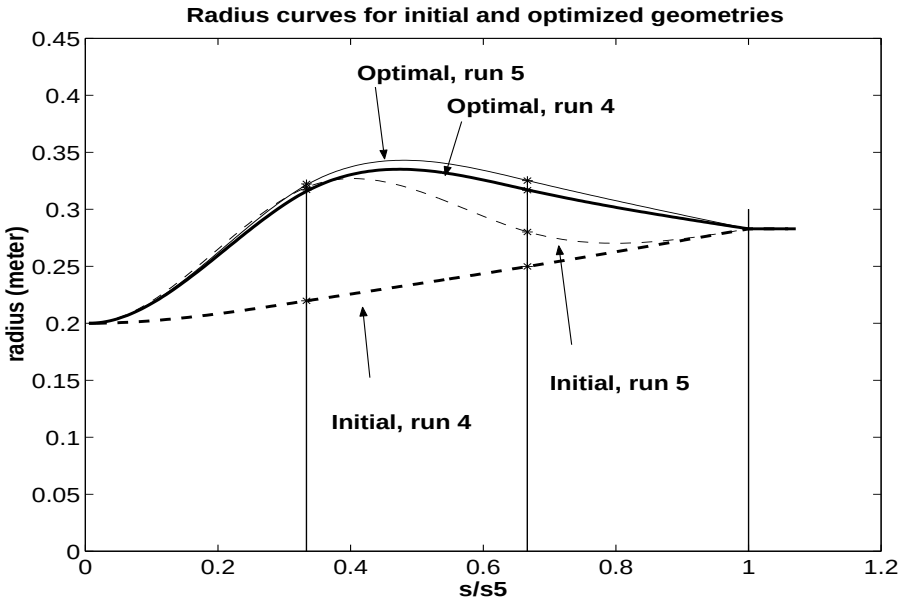


Figure 5.11: Radius curves for initial and optimized ducts with the loss objective function.

5.5.3 Optimal designs

In Tables 5.3 and 5.4 we see that optimization with respect to losses (runs 4 and 5) yield very good cavitation performance as well. The optimal value of $-C_{p,min}$ in run 4 and 5 (loss objective function) are even better than the optimal value of $-C_{p,min}$ from run 3. Similarily, but less striking, the optimization of the cavitation performance (runs 1 to 3) yields low losses.

Hence, ***under the given flow conditions, the two objective functions are dependent of each other.*** This may seem strange because $-C_{p,min}$ is a local measure in the sense that its value is attained at a particular point in the duct, whereas $\bar{C_P}$ is a measure of the loss over the entire flow domain. However, consider the distribution C_p (defined in Eq. (4.10)) over the top and bottom of the symmetry plane of the ducts, shown in Fig. 5.12, 5.13, 5.14 and 5.15. The x-axis is labelled by the distance in the x-direction from the inlet opening. We note that $-C_{p,min}$ is attained at the top of the duct exit in all optimized geometries. It is also at the exit that we calculate $\bar{C_P}$. Hence, at least close to the optimum of $-C_{p,min}$, increasing static pressure at the duct exit is benefitial for both objective functions. The relationship between the objectives are further investigated in Section 5.6.2.

The geometries of the optimized ducts differ significantly from each other, as can be seen in Fig. 5.6 and 5.10. Let $\mathbf{p}_1^*$ denote the optimal duct from run 1, and $\mathbf{p}_5^*$ the optimal duct from run 5. $\mathbf{p}_1^*$ has the lowest value of $-C_{p,min}$ of all the optimized ducts, and $\mathbf{p}_5^*$ the lowest value of $\bar{C_P}$. Consider the ducts

$$\mathbf{p}(\alpha) = (1-\alpha)\mathbf{p}_1^* + \alpha\mathbf{p}_5^*, \quad \alpha \in [0,1]$$

on a line between $\mathbf{p}_1^*$ and $\mathbf{p}_5^*$. In Fig. 5.17 we have plotted $-C_{p,min}(\mathbf{p}(\alpha))$ and $\bar{C_P}(\mathbf{p}(\alpha))$ for $\alpha = 0, \frac{1}{10}, \frac{2}{10}, \ldots, 1$. It can be seen that all the ducts $\mathbf{p}(\alpha)$ are near-optimal with respect to both cavitation and losses. Hence, all the optimized ducts correspond to the same "flat optimum", i.e., the same large basin in parameter space where the objective functions attain low, near-optimal values. We may draw the conclusion that ***under the given flow conditions, there are many duct designs which are near-optimal with respect to both cavitation and loss, at the same time.***

Consequently, it makes sense to summarize the optimal parameter values from all runs:

α	attains values between 54.0625 and 66.25 degrees,
R_1	attains values between 2.9 and 4.6125,
R_2	attains values between 0.975 and 1.55,
R_1/R_2	attains values between 2.6 and 3.5
r_1	attains values between 0.3025 and 0.3225,
r_2	attains values between 0.315 and 0.3325.

The optimal designs are thus characterized by

1. A long, sweeping upstream bend followed by a sharper downstream bend. The relation between the two bend radii is approximately $R_1/R_2 = 3$.

2. A moderate inclination of the duct, with $\alpha \approx 60^o$.

3. A contraction of the duct over the downstream bend. The cross-sectional radius (Fig. 5.7 and Fig. 5.11) increases rapidly over the first part of the

upstream bend, reaches its maximum mid-way along the duct and then decreases over the downstream bend.

Due to the dependency between the objective functions, it is hard, from the results obtained so far, to point out how the cavitation-optimal and loss-optimal designs differ from each other.

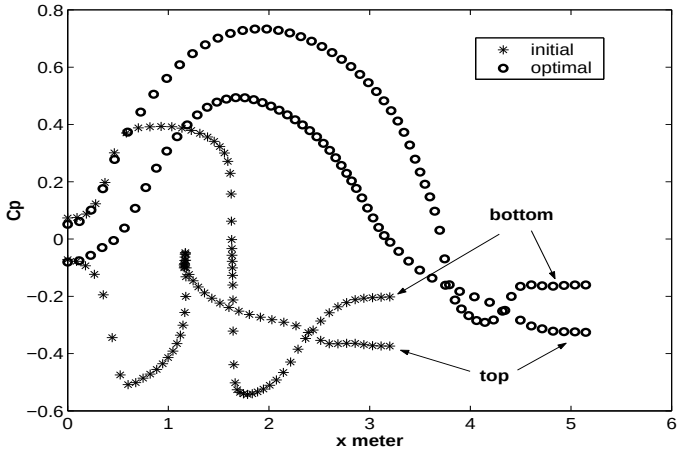


Figure 5.12: Distribution of C_p along the top and the bottom of the initial and optimal ducts for run 1. The corresponding result for run 2 is identical.

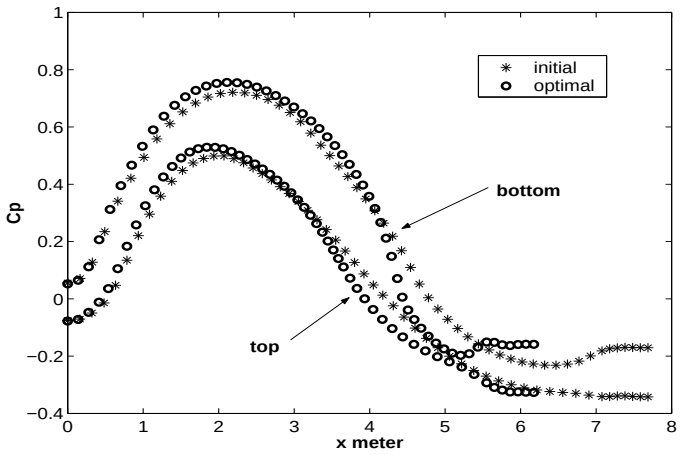


Figure 5.13: Distribution of C_p along the top and the bottom of the initial and optimal ducts for run 3.

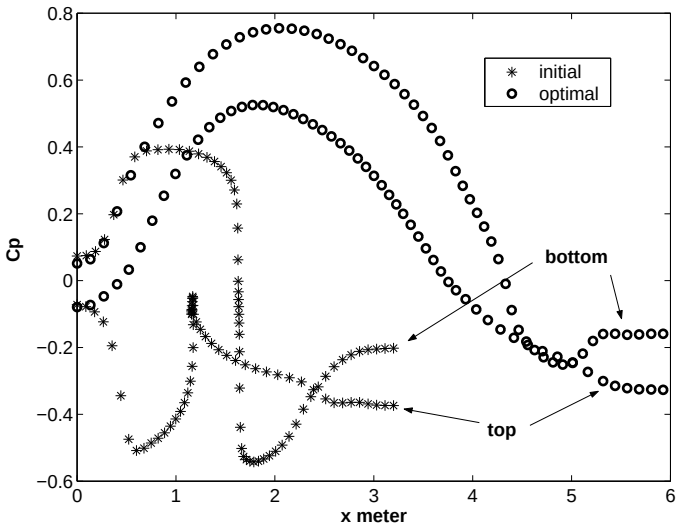


Figure 5.14: Distribution of C_p along the top and the bottom of the initial and optimal ducts for run 4.

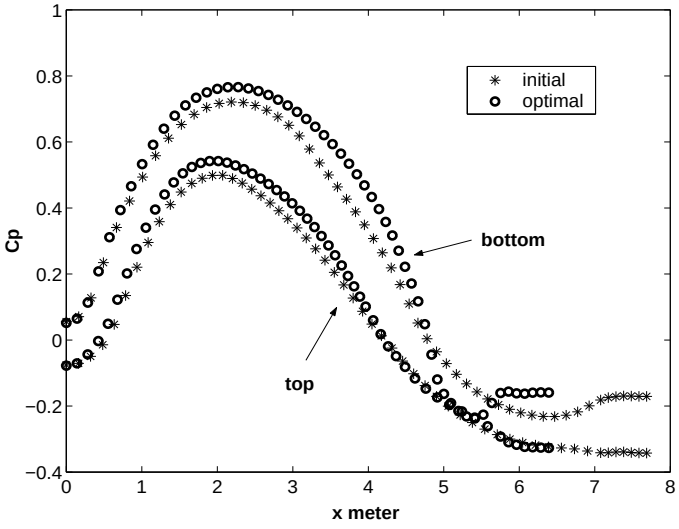


Figure 5.15: Distribution of C_p along the top and the bottom of the initial and optimal ducts for run 5.

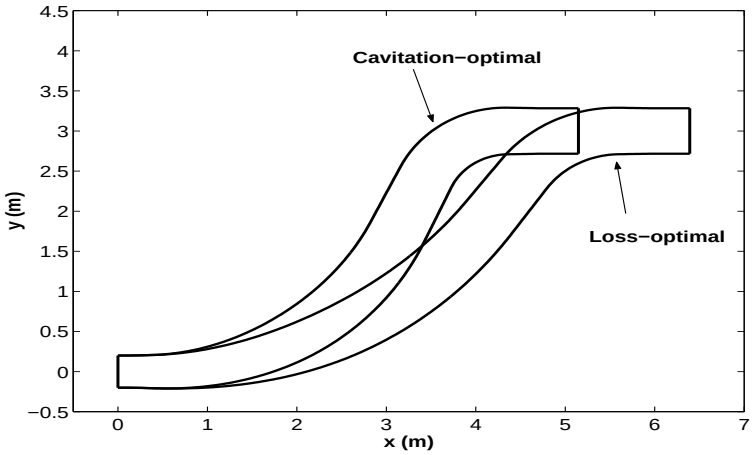


Figure 5.16: The best ducts found. The cavitation-optimal duct is from run 1 and the loss-optimal from run 5.

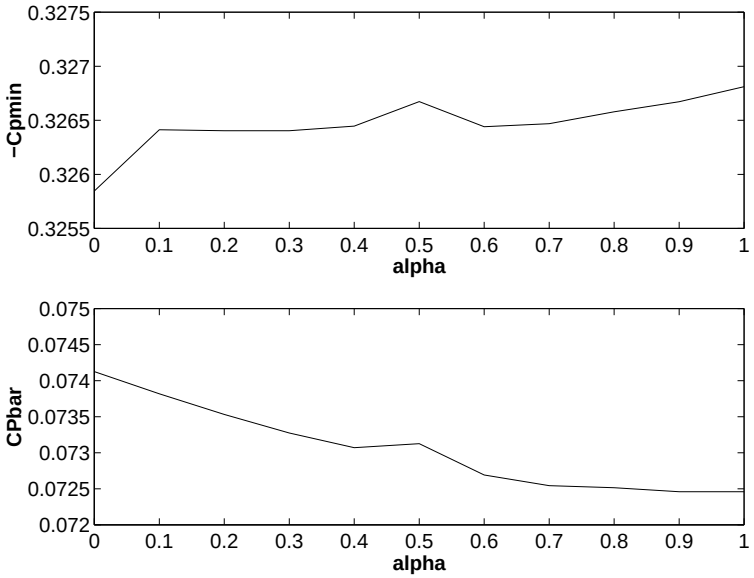


Figure 5.17: $-C_{p,min}$ and $\bar{C_P}$ for the ducts on a line in parameter space between the cavitation-optimal ($\alpha = 0$) and the loss-optimal ($\alpha = 1$) duct.

5.5.4 Hydrodynamics of initial and optimal designs

We now compare in detail the hydrodynamics of the initial and optimized ducts from run 1. The radius curve for this duct is shown in Fig. 5.7. The main differences between the hydrodynamics of the two ducts are that the flow separates in both bends in the initial duct but not in the optimal, and the boundary layer thickness over the bottom of the upper bend has been radically reduced in the optimal duct.

Fig. 5.18 and 5.19 show the distribution of C_p over the symmetry plane for the two ducts. Due to hydrostatic effects, both ducts have low values of C_p in the upper parts. The initial duct has low values of C_p at the inside of both bends and $C_{p,min}$ is attained in the middle of the downstream bend. The optimal duct has high values of C_p throughout the upstream bend and the location of the minimum C_p has moved to the top of the duct exit.

The distribution of C_p can be understood by considering the distribution of the pressure without the hydrostatic pressure in Fig. 5.20 and 5.21. The initial duct has large cross-stream pressure gradients in both bends, which give rise to low static pressure at the inside of both bends. The pressure gradients are caused by large bend angles and a relatively high bend curvature. The optimal duct has a much higher static pressure from the entrance to the upstream bend throughout the duct to the exit of the downstream bend, as a consequence of the larger cross-sectional duct diameter (Fig. 5.7). In combination with smaller cross-stream pressure gradients, in particular in the upstream bend, we get a much higher static pressure on the inside of the bends. In the downstream bend, the reduced cross-stream pressure gradient is a result of the smaller bend angle, and in the upstream bend it is due to smaller bend angle and curvature.

The perhaps largest qualitative difference between the flow in the two ducts is revealed by Fig. 5.30, 5.31, 5.32 and 5.33. We can see that *the flow separates at the inside of both bends in the initial duct but not in the optimal* - at least according to our computations.

The flow separation in the initial duct is caused by a rapid increase of boundary layer thickness (Fig. 5.26) in combination with adverse pressure gradients (Fig. 5.20).

Fig. 5.22 and 5.23 show the distribution of the normalized velocity U/U_{ref}, where U_{ref} is the inlet velocity. The overall impression is that the flow in the optimal duct is more uniform over each cross-section. The flow in the initial duct has larger velocity gradients than the optimal duct, which results in a larger total pressure loss. We can see how the boundary layer grows rapidly after the upstream bend in both ducts[1]. Since the flow actually separates in the initial duct we get a sharper border between the low-momentum boundary layer fluid and the fast-moving core flow, and thus larger cross-stream velocity gradients. The same thing happens on the inside of the downstream bend in the initial duct, but not in the optimal. Moreover, the cross-stream velocity gradient in the upstream bend is larger in the initial duct, due to the cross-stream pressure gradient present there.

We note in Fig. 5.33 that the velocity distribution is remarkably uniform over the upper bend in the optimal duct, and the boundary layer, represented by the contour of $C_P = 0.05$ (defined in Eq. (4.12)), is much thinner on the inside

[1]This is caused by the secondary flow in combination with the adverse pressure gradient over the second half of the inside of the bend; see Section 4.4.1.

after that bend in the optimal duct (Fig. 5.27) than in the initial (Fig. 5.26). This can be attributed to the accceleration of the flow due to the contraction of the duct there, in combination with the reduced strength of the secondary flow, which is seen Fig. 5.34 and 5.35.

The distribution of C_P over the duct exit is shown in Fig. 5.28 and 5.29.

Fig. 5.24 and 5.25 show the distribution of turbulence intensity over the symmetry plane. High levels of turbulence are associated with the separation/near-separation regions, due to the large velocity gradients present there. The optimal duct has lower levels of turbulence. In particular, compared to the initial duct, the turbulence intensity has decreased significantly over the second half of the inside of the upper bend.

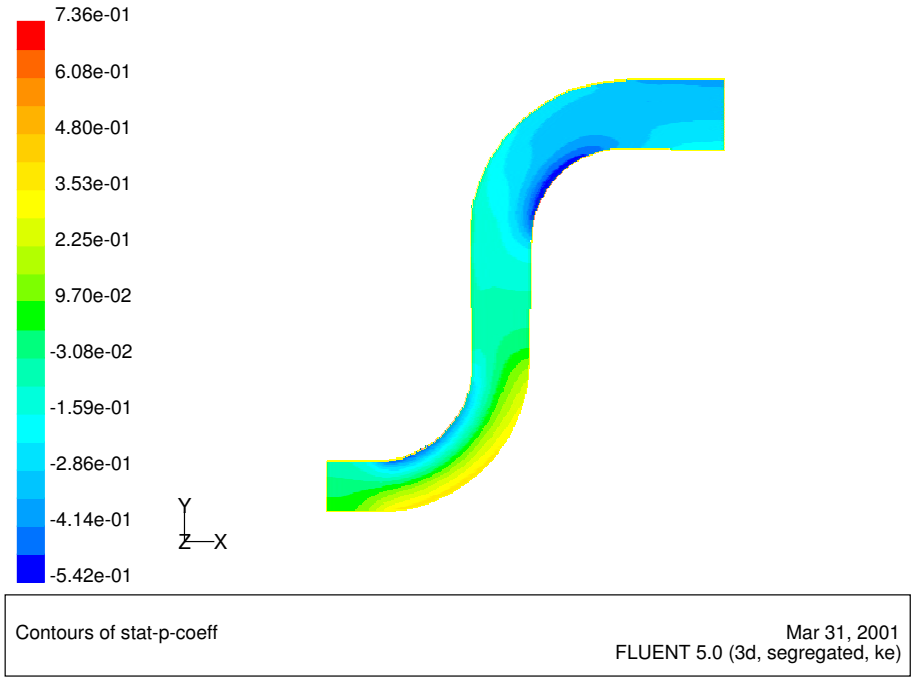


Figure 5.18: Distribution of C_p over symmetry plane, initial duct run 1.

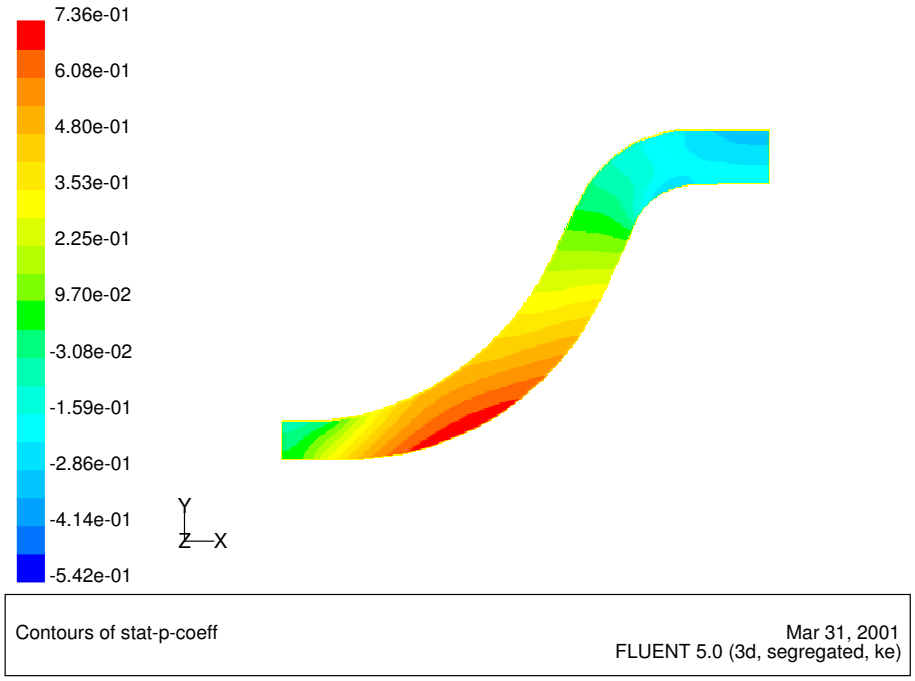


Figure 5.19: Distribution of C_p over symmetry plane, optimal duct run 1.

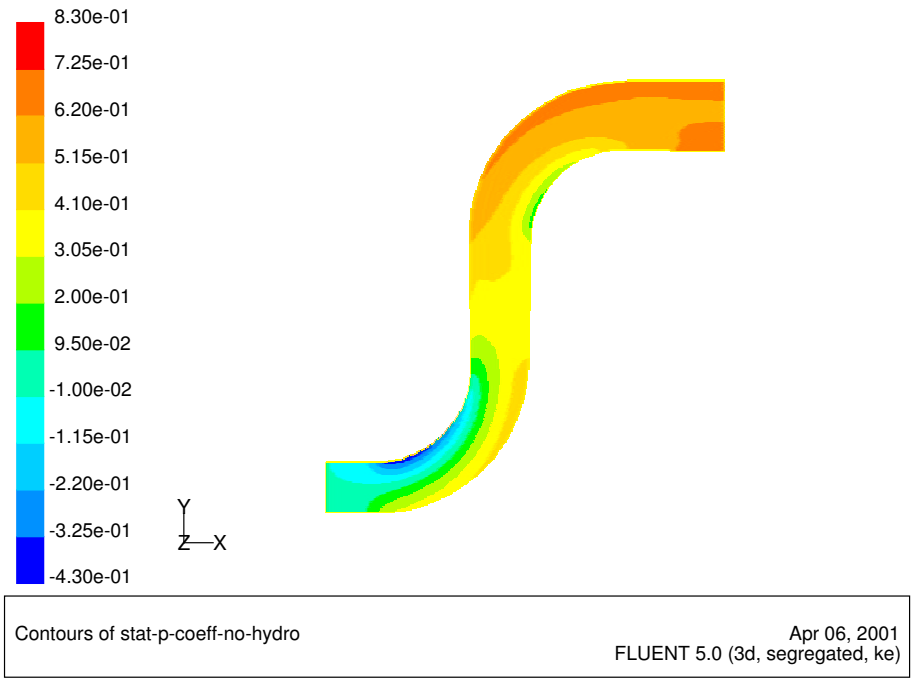


Figure 5.20: Distribution of $(p - p_{ref})/(0.5\rho U_{ref}^2)$ (C_p without hydrostatic pressure) over symmetry plane, initial duct run 1.

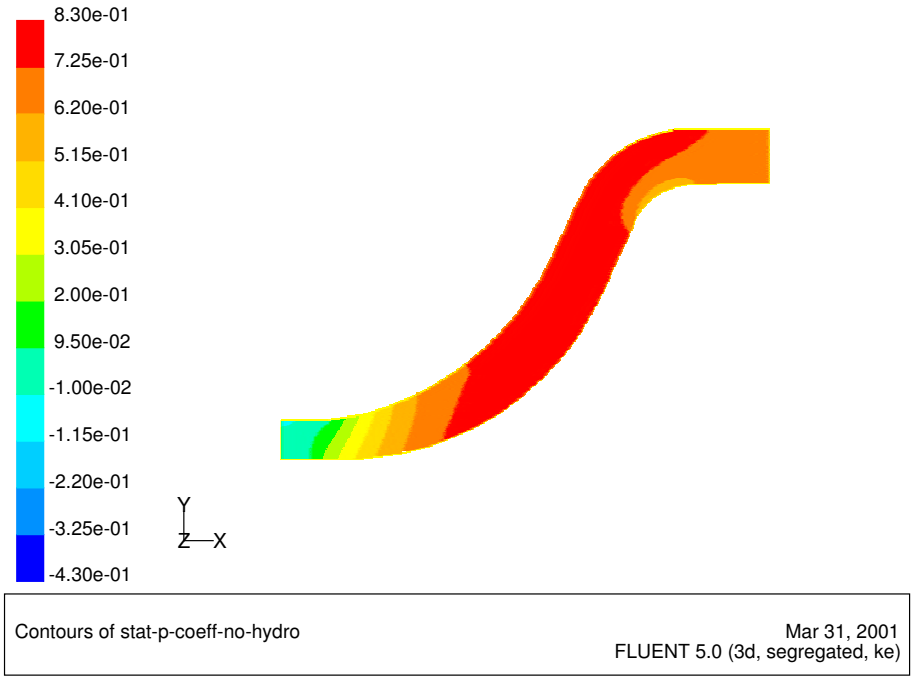


Figure 5.21: Distribution of $(p - p_{ref})/(0.5\rho U_{ref}^2)$ (C_p without hydrostatic pressure) over symmetry plane, optimal duct run 1.

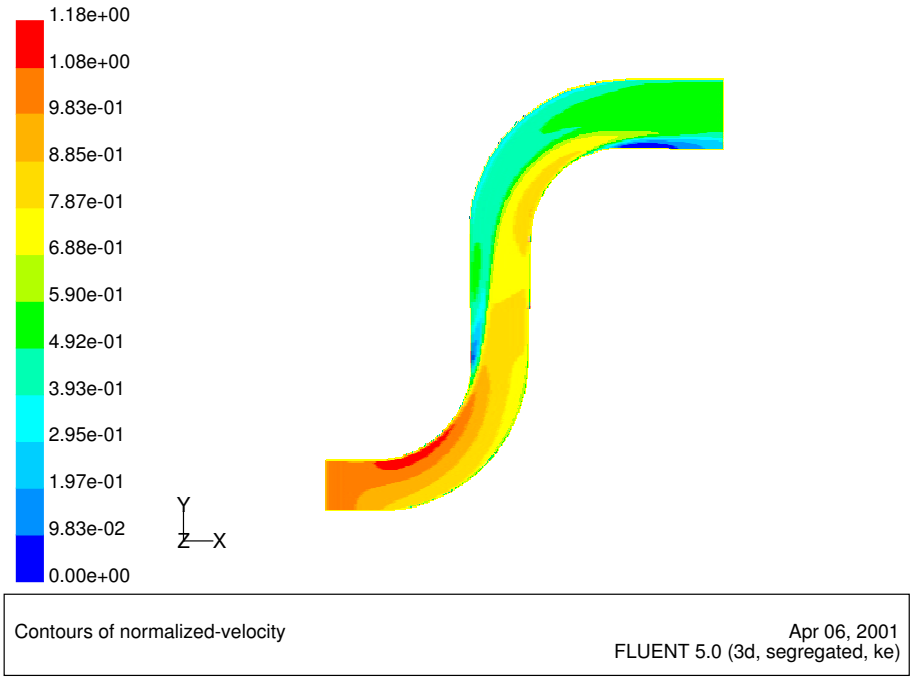


Figure 5.22: Distribution of U/U_{ref} over symmetry plane, initial duct run 1.

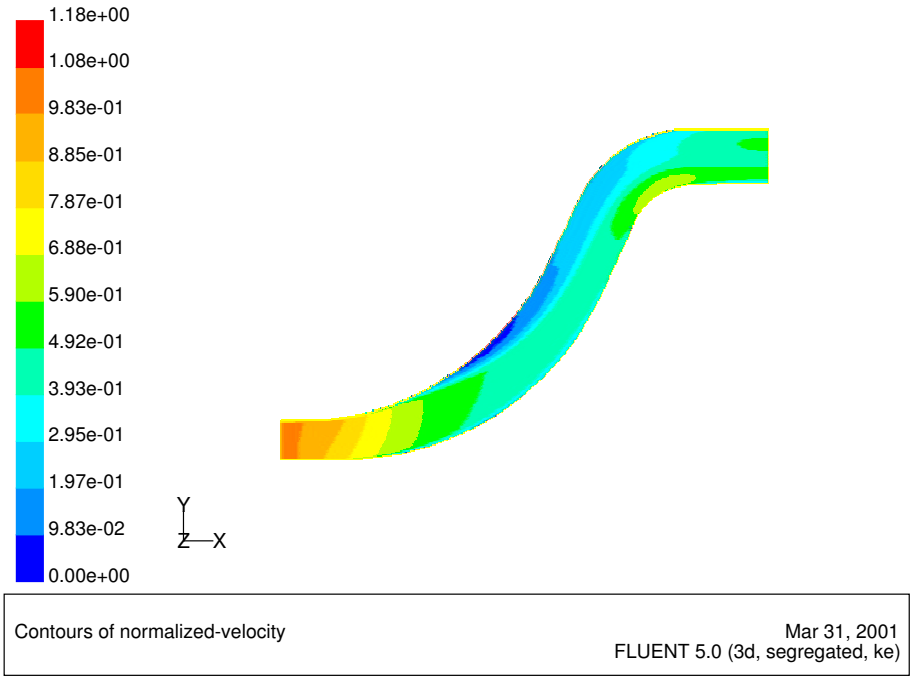


Figure 5.23: Distribution of U/U_{ref} over symmetry plane, optimal duct run 1.

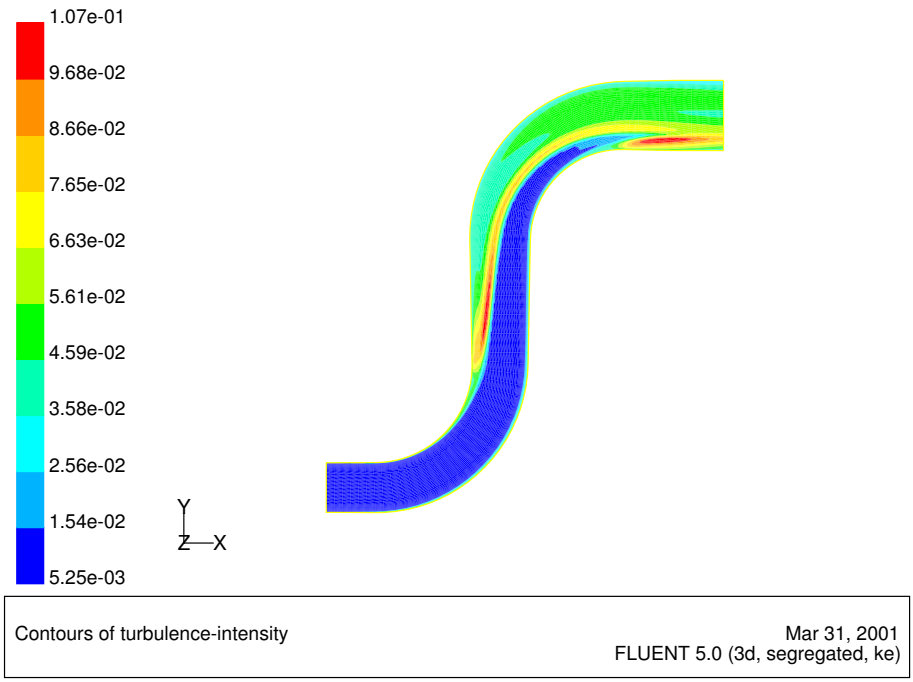


Figure 5.24: Distribution of turbulence intensity $\sqrt{2k/3}/U_{ref}$ over symmetry plane, initial duct run 1.

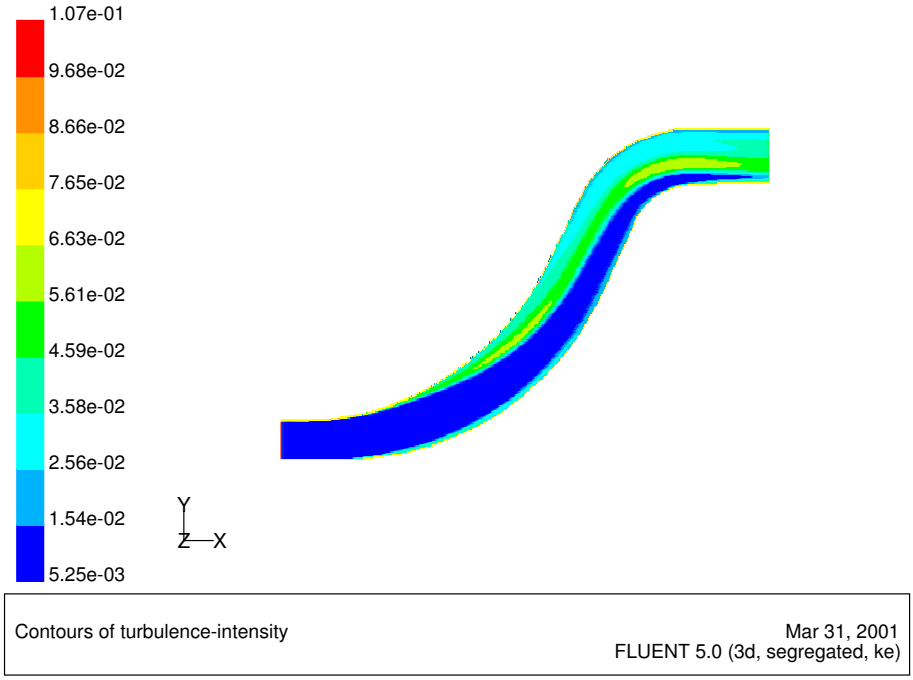


Figure 5.25: Distribution of turbulence intensity $\sqrt{2k/3}/U_{ref}$ over symmetry plane, optimal duct run 1.

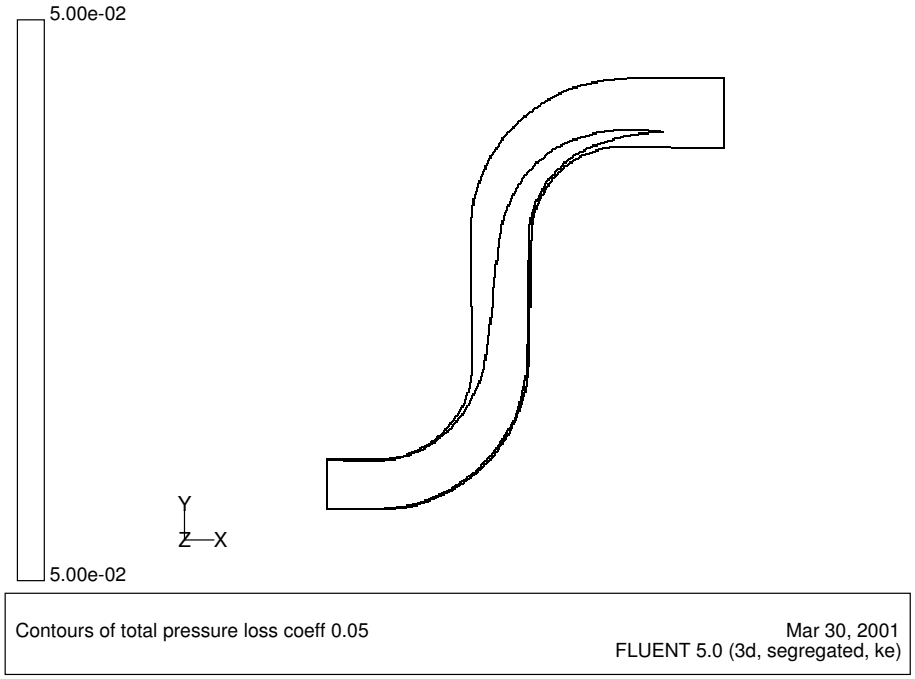


Figure 5.26: Boundary layer thickness represented by the contour of $C_P = 0.05$. Initial duct run 1.

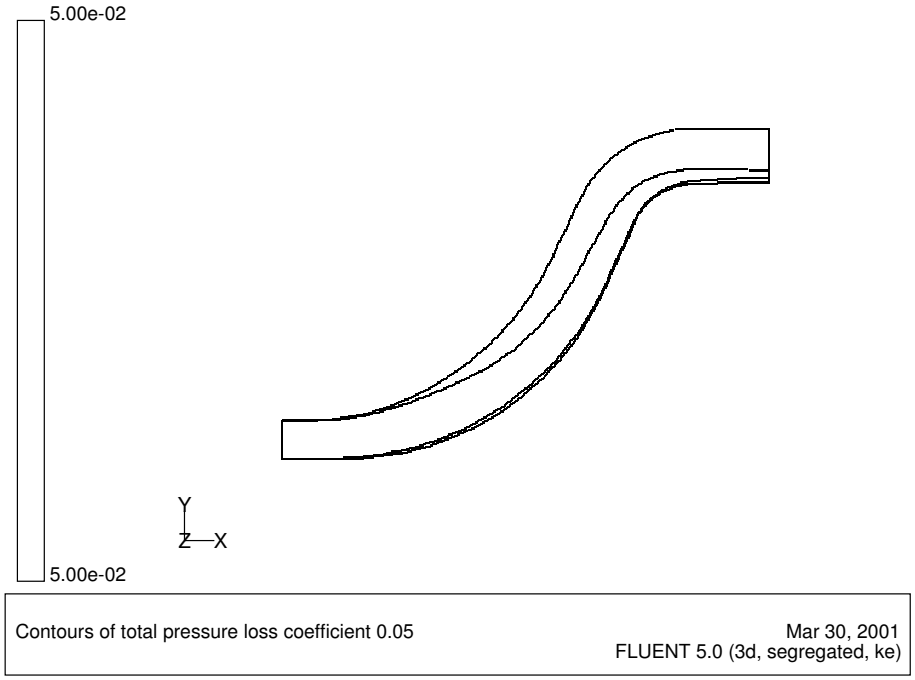


Figure 5.27: Boundary layer thickness represented by the contour of $C_P = 0.05$. Optimal duct run 1.

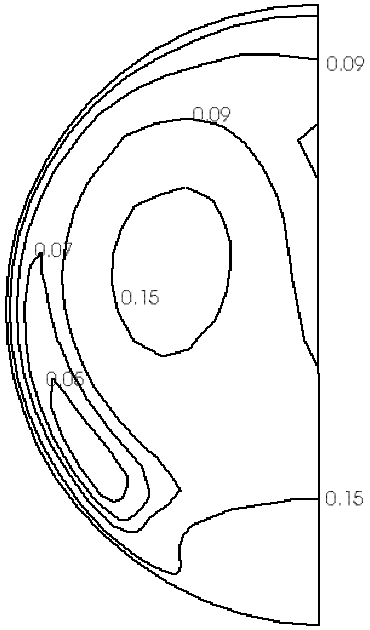


Figure 5.28: Contours of total pressure loss coefficient C_P at exit of initial duct run 1.

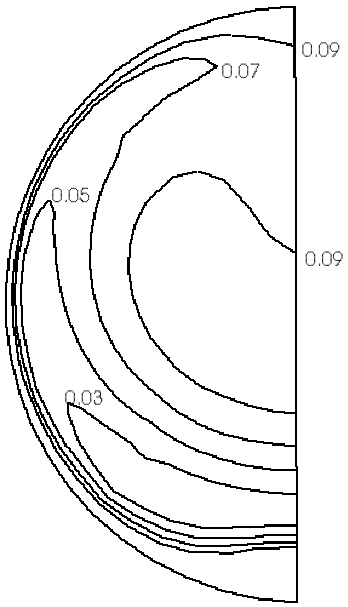


Figure 5.29: Contours of total pressure loss coefficient C_P at exit of optimal duct run 1.

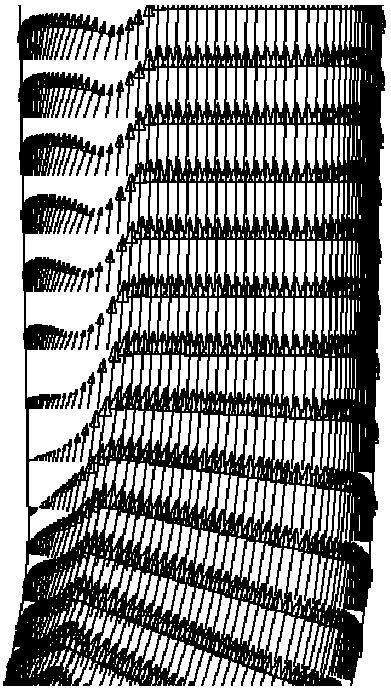


Figure 5.30: Velocity vectors after upstream bend, initial duct run 1.

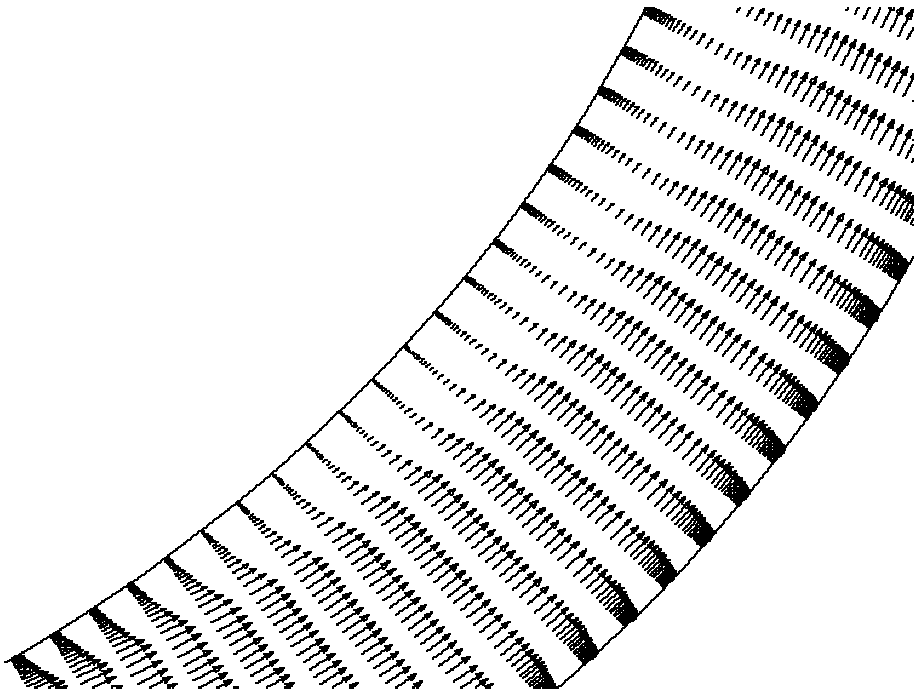


Figure 5.31: Velocity vectors in upstream bend, optimal duct run 1.

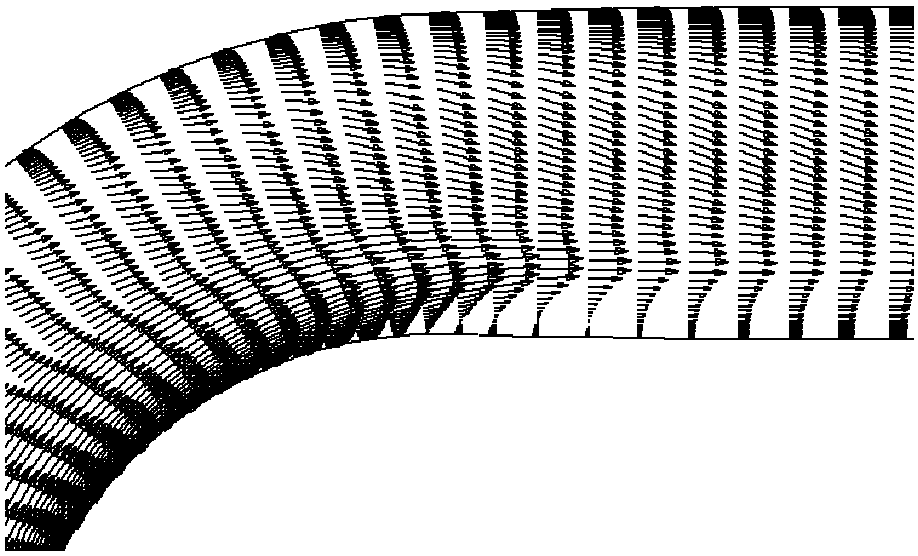


Figure 5.32: Velocity vectors in downstream bend, initial duct run 1.

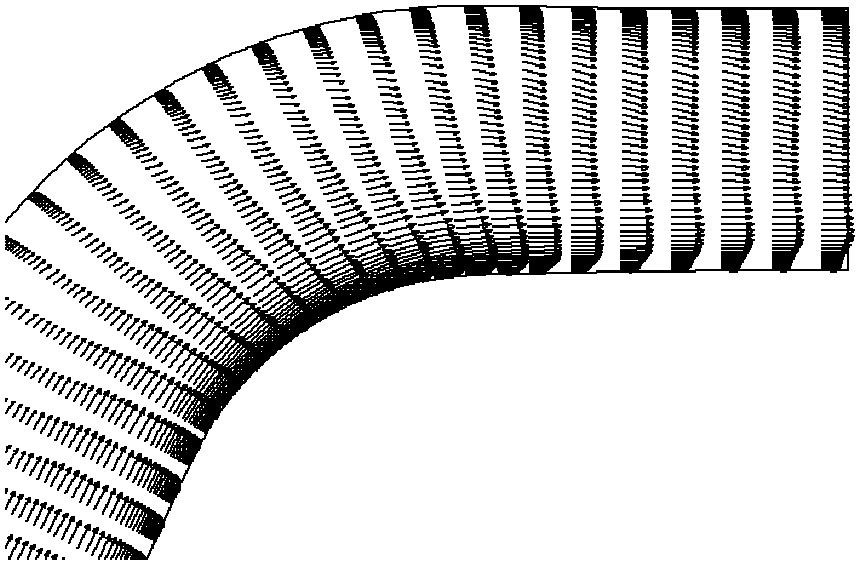


Figure 5.33: Velocity vectors in downstream bend, optimal duct run 1.

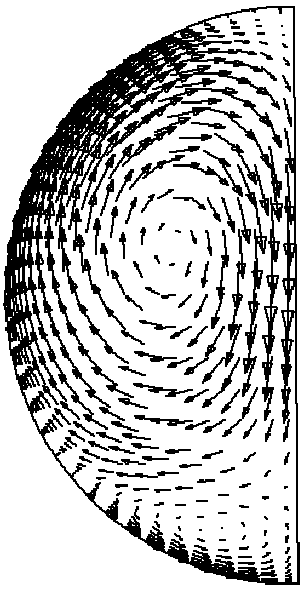


Figure 5.34: Secondary flow at exit of initial duct run 1.

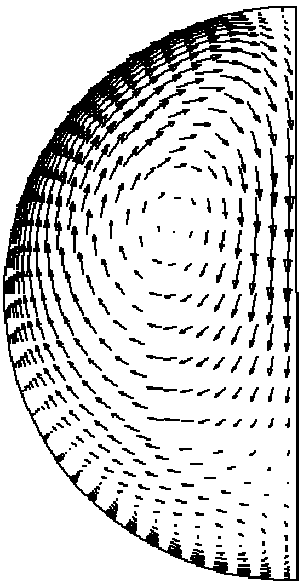


Figure 5.35: Secondary flow at exit of optimal duct run 1.

5.6 Discussion of the results

We discuss and continue the analysis of the results obtained in the previous sections.

5.6.1 Geometry and hydrodynamic performance

The risk of cavitation has been minimized in the optimal duct by increased static pressure due to a larger cross-sectional diameter, in combination with reduced cross-stream pressure gradients due to smaller bend angles.

The lower total pressure loss in the optimal duct is a consequence of the reduced secondary flow, the thinner boundary layer at the bottom of the duct after the downstream bend, and the fact that the flow does not separate. The reduced secondary flow is at least partly due to the smaller bend angles.

It is interesting to see how flow separation is avoided in the optimal duct by different mechanisms in the two bends. In the initial duct the flow separation in the lower bend was caused by the thickening of the boundary layer in combination with an adverse pressure gradient. The adverse pressure gradient results from the large bend angle and curvature, as well as the expansion of the duct. The thickening of the boundary layer in the lower bend is still present in the optimal duct (since it is mainly caused by secondary flow effects), but the size of the adverse pressure gradient at the critical location mid-way through the bend is much smaller. This is due to the decreased bend angle and curvature. Moreover, ***the most rapid increase in the cross-sectional diameter takes place in the beginning of the upstream bend, before the evolution of the boundary layer takes place.*** There, the boundary layer is still very thin and the adverse pressure gradient caused by the rapid expansion of the duct does not cause flow separation.

The separation of the flow in the upper bend in the initial duct is probably mainly caused by an adverse pressure gradient and not so much by secondary flow effects. The argument for this is that the two vortices that convect low-momentum fluid to the bottom of the duct in the upper bend are much weaker than the corresponding vortices in the lower bend. Their strengths are reduced by the two upper vortices that were induced by the lower bend (Fig. 5.34). Therefore, the reduction of the size of the adverse pressure gradient due to the contraction of the duct has a radical impact on the flow over the upper bend as can be seen by comparing Fig. 5.32 to Fig. 5.31. We may conclude that ***the reduced boundary layer thickness at the inside of the upper bend is primarily a consequence of the contraction of the duct.***

5.6.2 Relation between the objectives

In Section 5.5.3 we noted that the objective functions were dependent of each other. We explained this by the fact that the minimum static pressure was attained at the (top of the) outlet. Therefore, the dependency is at least partly a consequence of the specific flow conditions and the expansion of the duct. The reason is that the static pressure at the outlet can only be lower than the static pressure at the inlet for certain inlet velocities.

To see this, we use the definitions of P and p^* in Eq. (4.13) and (4.9) to

write

$$p^* = P - \rho g y - \frac{1}{2}\rho U^2.$$

Let IN and OUT refer to the top of the inlet and outlet opening, respectively. These are the locations on the inlet and outlet with the lowest static pressure. Then we can write the difference between the outlet and inlet pressure as

$$p^*_{OUT} - p^*_{IN} = P_{OUT} - P_{IN} + \rho g(y_{IN} - y_{OUT}) + \frac{1}{2}\rho(U^2_{IN} - U^2_{OUT}).$$

Since the volumetric flow rate is constant, we approximately have

$$U_{OUT} = U_{IN}\frac{A_{IN}}{A_{OUT}}.$$

The difference between y_{OUT} and y_{IN} is close to the duct height H. If we ignore viscous losses, so that $P_{OUT} = P_{IN}$ by Bernoulli's Theorem, we may therefore simplify the above equation to

$$p^*_{OUT} - p^*_{IN} = -\rho g H + \frac{1}{2}\rho U^2_{IN}\Big(1 - \Big(\frac{A_{IN}}{A_{OUT}}\Big)^2\Big).$$

For an expanding duct we have $A_{IN} < A_{OUT}$ and therefore, as we increase the velocity, the pressure at the outlet will eventually be higher than at the inlet. To see for which velocity we have $p^*_{OUT} = p^*_{IN}$, we set the right-hand side of the above equation equal to zero. Solving for U_{IN} yields

$$U_{IN} = \sqrt{\frac{2gH}{1 - (A_{IN}/A_{OUT})^2}}.$$

For our s-duct we have $A_{IN}/A_{OUT} = 1/2$ and $H = 3$, which gives $U_{IN} = 8.9$ m/s. For higher velocities the location where the minimum pressure is attained must move from the outlet which presumably alters the relationship between the objective functions.

To confirm these arguments and investigate them further, we compute the flow in the optimal duct from run 1 for inlet velocities 9, 10 and 12 m/s, in addition to the original 8 m/s used during the optimization.

Table 5.5 summarizes the values of $-C_{p,min}$ and $\bar{C_P}$, which were computed with the new reference conditions at the inlet. We see that both $-C_{p,min}$ and $\bar{C_P}$ decrease with increasing velocity.

Fig. 5.36 shows the distribution of C_p along the top and bottom of the duct symmetry plane for the different inlet velocities. We first note that when the inlet velocity is increased, the static pressure at the top of the outlet increases relative to the pressure at the inlet, as predicted above. Let us consider the location of the minimum static pressure. For 8 m/s it is attained at the exit. For 9 m/s it has moved to the downstream bend. For higher velocities, the minimum static pressure is attained at the top of the inlet. This confirms the theory above. The fact that the static pressure at the exit is not exactly equal to that at the inlet for 9 m/s can be attributed to viscous losses.

We also note in Fig. 5.36 how *all* values of C_p are increased with the velocity. We may conclude that ***in absence of sharp bends, the risk for internal***

	Inlet velocities			
	8 m/s	9 m/s	10 m/s	12 m/s
C_P	0.0741	0.0729	0.0718	0.0670
$-C_{p,min}$	0.3258	0.1175	0.0587	0.0469

Table 5.5: Effects of changing the inlet velocity in the optimal duct from run 1.

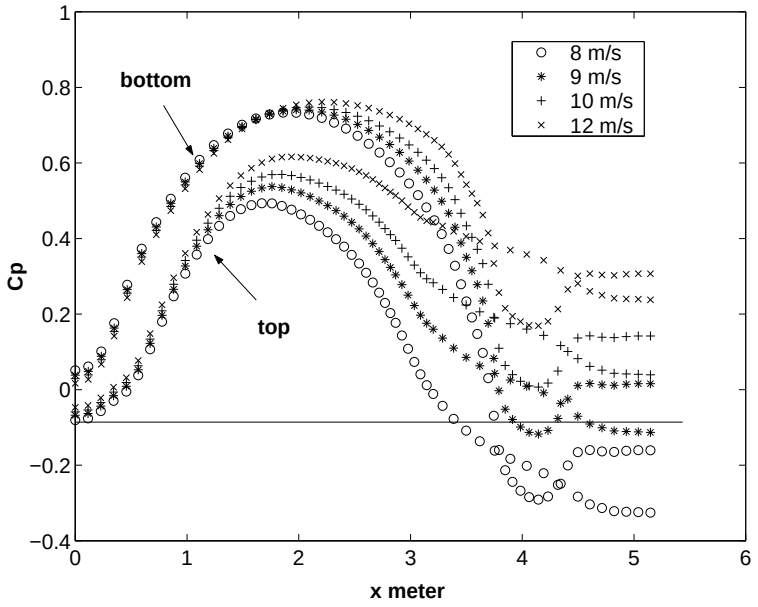


Figure 5.36: Distribution of C_p along the top and bottom of the optimal duct from run 1 for different inlet velocities. The straight line indicates the value of C_p at the top of the duct inlet.

cavitation is small for high velocities. (With "internal" we mean inside the duct. In practice, cavitation may occur at the entrance of the duct, but this scenario is not included in our model.)

Let us denote the value of C_p at the inlet by C_p^*. It would not be true to state that C_p^* is completely independent of $\mathbf{p}$, because it is influenced by the conditions in the upstream bend. However, this influence is limited, and mainly, C_p^* depends on the inlet radius and the inlet velocity. Consequently, for any duct, C_p^* is close to the theoretical minimum of $C_{p,min}$. When we have found a duct in which $C_{p,min}$ is attained at the inlet ($C_{p,min} = C_p^*$), changes in the downstream geometry that does not move the location of $C_{p,min}$ from the inlet will only have small effects on $C_{p,min}$ inasmuch as they affect C_p^*.

To see that this is indeed the case, we consider the ducts $\mathbf{p}_1$ and $\mathbf{p}_1^*$ in Fig. 5.37. The first is a long, smooth duct specified by

$$\mathbf{p}_1 = (30, 3.2, 4, 0.305, 0.32),$$

and the second is the optimal duct from run 1. The ducts have almost identical

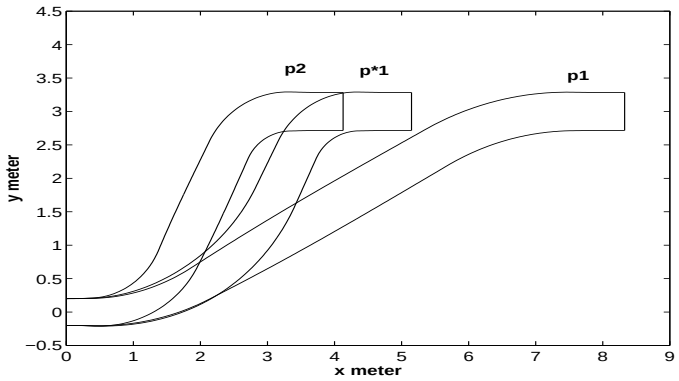


Figure 5.37: Geometry of three different ducts for the investigation of the dependency between the objective functions.

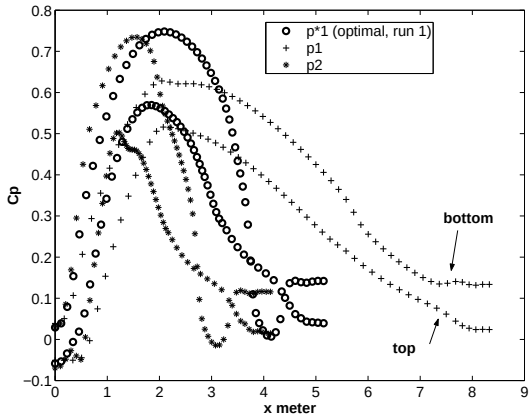


Figure 5.38: Distribution of C_p along top and bottom of the duct symmetry plane for the three ducts in Fig. 5.37.

values of $-C_{p,min}$, as seen in Table 5.6. Fig. 5.38 shows that the minimum static pressure is attained at the outlet. In fact, we have again examined ten ducts on a line in parameter space between $\mathbf{p}_1$ and $\mathbf{p}_1^*$, and all of them have values of $-C_{p,min}$ that are between those for $\mathbf{p}_1$ and $\mathbf{p}_1^*$. We conclude that ***as soon as $-C_{p,min}$ is attained at the duct inlet, we have reached a large basin of ducts that are near-optimal with respect to $-C_{p,min}$.***

To get more insight in the relationship between the objective functions, let us again consider Fig. 5.37 and 5.38. The duct $\mathbf{p}_2$ is given by

$$\mathbf{p}_2 = (65, 1.4, 1, 0.305, 0.32).$$

It is the same as $\mathbf{p}_1^*$ apart from a sharper upstream bend. Fig. 5.38 shows that for all ducts, the minimum static pressure is attained at the inlet. Consequently they have all approximately equal, near-optimal values of $-C_{p,min}$. However, in Table 5.6 we can see that they differ with respect to $\bar{C}_P$. $\mathbf{p}_2$ have largest losses, which can be attributed to the sharp upstream bend where heavy flow

	$\mathbf{p}_1^*$	$\mathbf{p}_1$	$\mathbf{p}_2$
C_P	0.0718	0.0794	0.0983
$-C_{p,min}$	0.0587	0.0509	0.0695

Table 5.6: Data for three different ducts for the investigation of the dependency between the objective functions.

	Opt. radius	Non-opt.
C_P	0.0741	0.1312
$-C_{p,min}$	0.3258	0.5919

Table 5.7: Effects of different radius curves. The centre curve is the one in the optimal duct from run 1.

separation actually occurs. $\mathbf{p}_1$ has lower losses than $\mathbf{p}_2$, because it is smoother, but it has higher losses than $\mathbf{p}_1^*$, which may be attributed to its length.

We have thus found that the objectives are not dependent in general. Their relation depends on the critical velocity where the outlet pressure can be higher than at the inlet. Furthermore, sharp bends may increase losses even if they do not affect the cavitation performance. Losses may also increase if the duct is unecessary long.

5.6.3 Effects of changing the radius curve

A carefully selected cross-sectional shape of the duct is essential to obtain cavitation- and loss-optimal designs. To convince the reader that this is indeed the case, we have compared the flow through the optimal duct from run 1 with the flow through duct that have the same center curve but a different, non-optimal radius curve, see Fig. 5.39.

Table 5.7 contains the values of $-C_{p,min}$ and $\bar{C_P}$. The values for the duct with the non-optimal radius curve is significantly higher than those for the optimal duct.

In Fig. 5.40 we can see how the static pressure distribution along the top and the bottom of the duct is affected by the change of radius curve. We note that the static pressure at the end points (inlet and outlet) remains more or less the same, but in between the shape of the curve is changed considerably. The optimal radius curve yields much higher static pressure over the upstream bend than the non-optimal. The influence of the sharp downstream bend (seen as the "dip" in the curve before the exit) is much more marked for the non-optimal radius curve. The location of the minimum static pressure has moved from the duct exit in the optimal duct to the mid-part of the downstream bend in the non-optimal duct.

The conclusion is that the effect of optimizing the radius curve is to smooth out the distribution of C_p, so that the effects of sharp bends become less marked. This should allow the centre curve to have sharper bends.

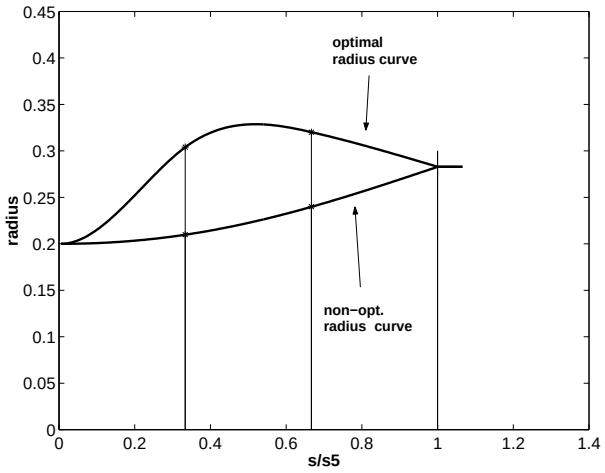


Figure 5.39: Different duct radius curves.

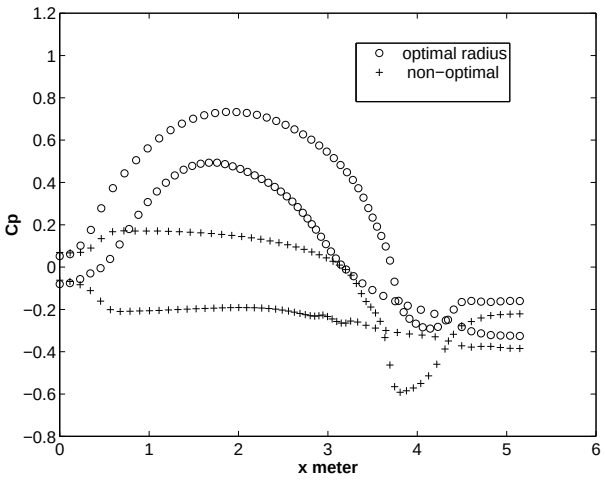


Figure 5.40: Effects of different radius curves on the distribution of C_p along the top and bottom of the duct symmetry plane. The centre curve is the one in the optimal duct from run 1.

5.6.4 Behaviour of the optimization algorithm

In all runs, the objective function values decreased rapidly during the first 20-30 evaluations (iterations), whereafter not much happened. Even so, it took many more iterations for the algorithm to converge. For example, in spite of the small improvement in the objective function made during step II in runs 2 to 5, the algorithm still required approximately another 100 evaluations in order to converge ($\Delta < \Delta_{min} = 0.01$). This can be attributed to the following property of the algorithm: each reduction (halving) of the step length requires 10 (2*n, where n is the dimension of the optimization problem) function evaluations. Thus, to reduce the step length from 1 to less than 0.01 requires at least 70 evaluations (even if the current iterate is optimal).

In Section 5.3 we noted that the non-linear constraints and the possibly low regularity of the cavitation objective function could cause convergence problems, at least in theory. Numerical investigations show that the optimal vectors are quite far from the boundaries of the constraint domain Ω. Therefore, the constraints should not have caused any false convergence in practice. It is harder to say anything about the influence of the regularity of $-C_{p,min}$ on convergence.

5.6.5 The surrogate functions

In one way we can say that the coarse mesh functions $-C^{\alpha}_{p,min}$ and $\bar{C_P}^{\alpha}$ were very good surrogate functions because when they were optimized, the values of $-C^{\alpha}_{p,min}$ and $\bar{C_P}$ decreased to almost the same extent. The surrogates provided very good initial guesses for the optimization in step II, where only very small improvements in the objective were made. On the other hand, the surrogates did not give significantly faster convergence, except for run 3.

Consider run 4 in Fig. 5.8. Let $\mathbf{p}_0$ denote the initial vector in step I. We have $\bar{C_P}^{\alpha}(\mathbf{p}_0) = 0.1336$ and $\bar{C_P}(\mathbf{p}_0) = 0.1172$, which is a difference of 14%. Thus, the surrogate loss function is only a crude approximation of the loss function, at least in some regions of the parameter space. Nevertheless, the optimum of step I provides such a good initial guess in step II that almost no further improvement in the objective is made. We conclude again (as in Section 5.4.1) that the most important property of a surrogate for optimization is not that it approximates the function itself, but rather that it approximates the differences in function values between any two points.

5.7 Summary and conclusions

We have used the Sherif-Boice algorithm [35] to optimize the waterjet inlet performance model described in Chapter 4 for maximal static pressure (minimal risk of cavitation) as well as minimum loss. The implemented procedure is fully automatic and terminates within a reasonable amount of time. The credibility of the procedure is shown by examples where the optimization is started from a "poor" initial design (one that is known to be less efficient) and gives an optimized design with significantly improved hydrodynamic performance - higher static pressure, lower loss, no flow separation. Furthermore, the inlet model was used as a tool for investigating the interaction between the inlet geometry and its hydrodynamic performance under different flow conditions. We have also

discussed how to speed up the optimization process by using coarse grids and a premature termination of the flow solver.

It was found that, under the given flow conditions, the objective functions were dependent of each other, that is, when we optimized for maximal static pressure we obtained low losses as well, and vice versa. We also found a *range* of geometries which were near-optimal with respect to both objective functions at the same time, so the geometry of the optimal ducts could be altered quite significantly with a preserved good hydrodynamic performance. The main geometric features of the optimal ducts are

1. a long sweeping lower bend radius,
2. a sharper upper bend,
3. and a contraction of the duct over the upper bend.
4. The maximum cross-sectional radius is attained approximately mid-way through the duct.

As is discussed in the following section, the modelling of the inlet duct has been done under considerable simplifications. Therefore, perhaps more interesting than the result of the optimization is an understanding of how the hydrodynamics relate to the underlying geometry. We have analyzed the hydrodynamics of one optimal duct as compared to an initial "poor" geometry. The hydrodynamics of the optimal duct are summarized by:

1. Higher static pressure at the inside of the bends,
2. reduced adverse pressure gradients at the inside of the bends so that the flow does not separate,
3. a more uniform flow over the second half of the upper bend, with a relatively thin boundary layer at the inside of that bend,
4. reduced secondary flow.

The contraction of the duct over the upper (relatively sharp) bend thins out the boundary layer at the inside of that bend, thereby reducing viscous losses. The large upstream bend radius minimizes the adverse pressure gradients and thus the risk of flow separation. The large radius in that bend can also be seen as necessary for the cross-sectional radius to be able to increase and reach its maximum mid-way through the duct without causing flow separation. It is interesting to note how the most rapid increase in the cross-sectional radius takes place in the first part of the lower bend, before the critical point where the boundary layer starts to grow. This is probably essential in order to avoid that the flow separates.

We can conclude that the parametric model of the inlet geometry described in Section 4.2 worked well. It was a simple model, consisting of only five parameters; nevertheless it could be used to generate a relevant range of geometries.

It was found that the observed dependency between the objectives does not hold in general, but is a consequence of the given inlet velocity U_{ref}. In our case the dependency could be attributed to the fact that the minimum static pressure was attained at the top of the outlet in all optimized ducts. However,

analysis and numerical experiments show that, due to the expansion of the duct, the downstream pressure will increase relative to the pressure at the inlet when the inlet velocity is increased. Typically, it is only possible for the minimum static pressure to be attained at the outlet when $U_{ref} < 9$ m/s.

The risk for internal cavitation should be reduced when the inlet velocity is increased, at least in the absence of sharp bends. A duct that is cavitation-optimal for one velocity should be cavitation-optimal for higher inlet velocities as well. This is due to the just-mentioned fact that the downstream pressure increases relative to the pressure at the inlet when the inlet velocity is increased.

For $U_{ref} > 9$ m/s the minimum static pressure may be attained at the inlet. We found that when this happens we have reached a large basin in the parameter space where all ducts give values close to the theoretical minimum of the cavitation performance measure ($-C_{p,min}$). We used this to construct two examples that show that the objectives are indeed not dependent in general. Both ducts had the minimum static pressure at the inlet opening and thus near-optimal cavitation performance. But both ducts had relatively large losses; one due to a sharp lower bend where the flow separated and the other because it was very long. This also shows that the cavitation measure $-C_{p,min}$ can not be used as the only flow performance measure.

Our experience with the Sherif-Boice algorithm (SBA) can be summarized as follows:

- It gave near-optimal function values within only 20-30 function evaluations.
- It worked well together with the surrogate functions.
- It required many iterations to converge. This is explained by the fact that the algorithm must perform $2n$ (where n is the number of variables) evaluations each time the step length is halved.

The surrogate functions were constructed by performing the flow computation in Fluent on a relatively coarse grid. The surrogates were approximately three times as fast to evaluate as the real functions. We summarize our conclusions about the surrogates as follows:

- The surrogates preserved enough of the nature of the problem to drive the optimization process in the right direction. Since the SBA is a rank-order method, the desirable property of the surrogates is that they maintain the rank between different function values.
- In a short period of time the optimization on the surrogates produced designs that were near-optimal also when measured with the real functions.
- They can be used to get quick, preliminary results about the correspondence between geometry and flow performance.

We conclude that the SBA, although it is a very simple algorithm, worked well on our problem. It is not believed that other algorithms would perform significantly better on problems like this, as long as the number of variables is as low as in our case, but perhaps it is possible to find a better stopping criterion.

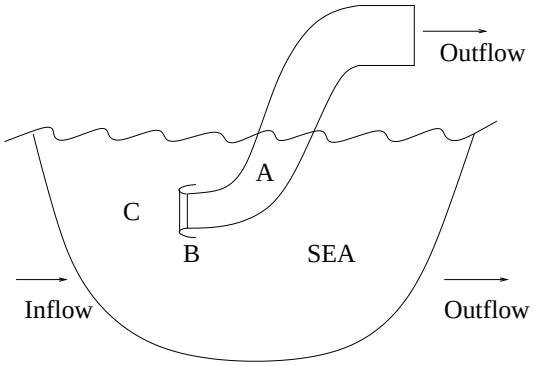


Figure 5.41: Extended flow domain.

5.8 Future research

As pointed out in the Introduction to the thesis, there are some shortcomings in our waterjet inlet modelling. In particular, we have only considered internal aspects of the flow. This might affect the direct applicability of our conclusions concerning the inlet design. For example, the optimized ducts expand rapidly before the lower bend and become the thickest mid-way through the duct. In practice, this will increase the drag as the inlet ploughs through the sea. Moreover, the analysis of the losses at the end of Section 4.5.2 indicates that the loss depends on the conditions at the inlet opening, but these are more or less constant in our model since we specify a uniform inlet velocity condition.

Therefore, it would be desirable to extend the model by extending the flow domain to include a volume of water surrounding the inlet opening as shown in Fig. 5.41. Such an arrangement was used by Seil [34] to model the flow into a flush-type inlet. With this extended flow domain it would be possible to include more aspects of the hydrodynamic performance of the duct into the optimization, for example:

- external drag around A in Fig. 5.41,
- drag due to differences between the free-stream pressure (at C) and the pressure inside the duct (A),
- losses associated with how the flow enters into the duct,
- cavitation performance around the inlet lip (B) and at the inlet opening.

Seil [34] notes that "... an actual waterjet inlet must reflect a compromise between the cruise and manoeuvring conditions,... ". This suggests the challenging problem of optimizing the inlet for a range of operating conditions simultanously. However, for such an optimization to be really interesting, it would be necessary to extend the flow domain as indicated above. The reason is that it allows us to include the important effects from different inlet velocity ratios (IVR), which can be defined as the ratio between the volumetrically-averaged velocity at the duct exit and the ship speed [34]. The flow may be very different

for different IVR and the IVR can range from 0.5 for a cruise condition to ∞ for a starting or manoeuvring condition.

A possible extension of the geometric model used in this work could be to allow for different cross-sectional shapes. For example, the cross-section could be rectangular in the beginning and elliptic at the exit with a successive transformation between these two shapes along the duct.

More studies can be done on how to make use of knowledge about the CFD model to reduce the amount of time required for the optimization. Suggestions for further research on this issue are:

1. improve the efficiency of using good initial flow solutions in the flow solver;
2. construct surrogates that are faster to evaluate;
3. consider to use a range of surrogates in the optimization;
4. investigate the possibility of *progressive optimization* [11], by which is meant a continuous update of the flow solver - finer grid, more iterations in the flow solver - as the optimization proceeds.

The value of surrogates should become more marked when the number of variables in the optimization problem is increased.

Appendix A

Positive linear dependence and the regular simplex in optimization

The purpose of this chapter is to investigate some properties of the regular simplex from the point of view of optimization.

Simplices are geometric constructions consisting of $n+1$ vectors, called vertices, in $\mathbb{R}^n$. They are used in some nonlinear optimization methods, such as the classical simplex method [16] and some pattern search methods [22]. However, they are also of interest to the analysis of general pattern search methods, because, as we will show, simplices are minimal positive bases. Positive bases are used in pattern search methods to define the search directions. Hence, it is interesting to know how well a positive basis can approximate the gradient (see also discussion in Section 2.5.2). We show that the cosine of the angle between an arbitrary vector and the closest vector in a minimal positive basis consisting of a regular simplex is bounded below by $1/n$.

We also investigate the geometric properties of the iterates generated by the classical simplex method by Spendley, Hext and Himsworth [16]. This method proceeds by generating a sequence of simplices in $\mathbb{R}^n$, where the next simplex is found by reflecting one vertex of the current simplex in the plane through the remaining vertices. Inspired by Powell [32] we show that, at least when $n > 3$ is not a power of 2, repeated use of such reflections may result in an infinite sequence of points in a bounded domain. Hence the iterates generated by the classical simplex method do not stay on a lattice as do the iterates generated by pattern search methods. The lattice structure of the iterates was essential to the convergence analysis of the last mentioned methods [36].

Our analysis uses some basic theory of positive linear dependence, which we now present.

A.1 Positive linear dependence and positive bases in $\mathbb{R}^n$

The *positive span* of $\{a_1, \cdots, a_r\} \subset \mathbb{R}^n$ is the cone

$$\{a \in \mathbb{R}^n : a = c_1 a_1 + \cdots + c_r a_r, c_i \in \mathbb{R}, c_i \geq 0, \forall i\}.$$

The set $\{a_1, \cdots, a_r\}$ is called *positively dependent* if some a_i is in the positive span of the others, otherwise positively independent. A *positive basis* is a positively independent set whose positive span is $\mathbb{R}^n$. A positive basis must contain at least $n+1$ elements and such a basis is called *minimal*. Also, it contains at most $2n$ elements and such a basis is called *maximal*.

From [12] we have the following two theorems of which the second characterizes positive spans.

Theorem 3. *Suppose $\{a_1, \cdots, a_r\}$ positively spans $\mathbb{R}^n$. Then $\{a_2, \cdots, a_r\}$ linearly spans $\mathbb{R}^n$.*

Theorem 4. *Suppose $\{a_1, \cdots, a_r\}, a_i \neq 0$, linearly spans $\mathbb{R}^n$. Then the following are equivalent:*

1. *$\{a_1, \cdots, a_r\}$ positively spans $\mathbb{R}^n$.*
2. *For every $b \neq 0$, there exists an i such that $b \bullet a_i > 0$.*
3. *For every i, $-a_i$ is in the positive span of the remaining a_i:s.*

For what follows, it is also convenient to have

Lemma 5. *Let $\{a_1, \cdots, a_{n+1}\}$ be a minimal positive basis for $\mathbb{R}^n$. For any given $x \in \mathbb{R}^n$ we may choose n of the a_i:s so that x is in their positive span.*

Proof. From Theorem 3 we have $x = \sum_{i=1}^{n} b_i a_i$, $b_i \in \mathbb{R}$. If $b_j < 0$ we can use 3 in Theorem 4 to replace $b_j a_j$ with a positive linear combination of the remaining a_i:s. This gives x as a linear combination of n of the a_i:s where one more coefficient is positive than in the first linear combination. Repeating this procedure eventually gives the desired result. □

A.2 The regular simplex in $\mathbb{R}^n$

Simplices may be defined in different ways. In some parts of mathematics, they are defined as n-dimensional sets in $(n+1)$-dimensional space. In simplex methods for nonlinear optimization (originally described in [16]), however, they are given by $n+1$ vectors in $\mathbb{R}^n$, whose convex hull is n-dimensional.

A.2.1 The regular simplex as a minimal positive basis

In our definition of a regular simplex, we only use the property that all the edges are of equal (non-zero) length.

Definition 1. *The set of vectors $\{v_1, \cdots, v_{n+1}\}$ in $\mathbb{R}^n$ is called a* regular simplex *if $v_i \neq v_j$ and $\|v_i - v_j\| = constant > 0$, $\forall i \neq j$. The v_i:s are called the* vertices *of the regular simplex. If $\|v_i\| = 1, \forall i$, the regular simplex is said to be* normalized.

Intuitively, a normalized regular simplex constitutes a minimal positive basis. In order to show this, we first prove the following proposition.

Proposition 6. *Any n vectors in a normalized regular simplex are linearly independent.*

Proof. Let $\{v_1, \cdots, v_{n+1}\}$ denote a normalized regular simplex. Suppose that

$$y := \sum_{i=1}^{n} c_i v_i = 0.$$

Definition 1 implies that $v_i \bullet v_j$ is equal to some constant α for all $i \neq j$. More precisely,

$$\begin{aligned} \text{constant} &= \|v_i - v_j\|^2 = \|v_i\|^2 + \|v_j\|^2 - 2\, v_i \bullet v_j \\ &= 2 - 2\, v_i \bullet v_j, \forall i \neq j. \end{aligned} \tag{A.1}$$

We then have

$$y \bullet v_{n+1} = \alpha \sum_{i=1}^{n} c_i = 0,$$

$$y \bullet v_j = c_j + \alpha \sum_{i=1, i\neq j}^{n} c_i = 0,$$

so that $c_j = \alpha c_j$. Then either $\alpha = 1$ or $c_j = 0$, but $\alpha = 1$ implies that all v_i are the same, which is contrary to the definition of a regular simplex. Thus $c_j = 0$, $\forall j$, which shows the linear independence of $v_1, \cdots, v_n$. It easily follows that any n of the v_i:s are linearly independent. □

Theorem 7. *Let $\{v_1, \cdots, v_{n+1}\}$ be a normalized regular simplex in $\mathbb{R}^n$. Then it is also a minimal positive basis.*

Proof. We will use theorem 4. From Proposition 6 it follows that $\{v_2, \cdots, v_{n+1}\}$ linearly spans $\mathbb{R}^n$. Let

$$-v_1 = \sum_{i=2}^{n+1} c_i v_i.$$

If $j > 1$ then

$$v_j - v_1 = (1 + c_j) v_j + \sum_{i=2, i\neq j}^{n+1} c_i v_i$$

We now take the scalar product of both sides in this equation with $v_j - v_1$ and, using equation (A.1) and $v_i \bullet v_j = \alpha (\neq 0)$, $\forall i \neq j$, so obtain

$$2(1 - \alpha) = \|v_j - v_1\|^2 = (1 + c_j)(1 - \alpha).$$

Hence $c_j = 1$, $\forall j$, and

$$\sum_{i=1}^{n+1} v_i = 0. \tag{A.2}$$

Theorem 4 combined with the fact that a positive basis contains at least $n + 1$ elements now gives that the v_i:s form a minimal positive basis. □

To show that Definition 1 is meaningful, we next prove that normalized regular simplices exist in all dimensions. To this end, it is convenient to have the following lemma.

Lemma 8. *Let $\{v_1, \cdots, v_{n+1}\}$ be a normalized regular simplex. Then*

$$v_i \bullet v_j = -\frac{1}{n}.$$

Proof. From (A.1) we have $v_i \bullet v_j = \alpha, \forall i \neq j$. We then obtain the desired result by taking the scalar product with v_1 in (A.2). □

Theorem 9. *There exists a normalized regular simplex in $\mathbb{R}^n$, for any n.*

Proof. The proof is constructive and we use induction on the dimension n. The existence is clear for $n = 2$. Now, suppose $\{v_1, \cdots, v_n\}$ is a normalized regular simplex in $\mathbb{R}^{n-1}$ for some $n - 1 \geq 2$. Equation (A.1) and Lemma 8 implies that

$$\|v_i - v_j\|^2 = 2\Big(1 + \frac{1}{n-1}\Big). \tag{A.3}$$

Denoting the elements of the vector v_i by v_{ij}, $j = 1, \ldots, n$, we define the vectors $w_i \in \mathbb{R}^n$ by

$$w_i = \begin{cases} (v_{i1}/\gamma, \ldots, v_{in}/\gamma, \beta), & \text{for } i = 1, \ldots, n, \\ (0, \ldots, 0, 1), & \text{when } i = n + 1. \end{cases}$$

where β and γ are real numbers. In order for $\{w_i\}$ to be a normalized regular simplex we must have

$$\frac{2\Big(1 + \frac{1}{n-1}\Big)}{\gamma^2} = \frac{1}{\gamma^2} + (1 - \beta)^2$$

and

$$1 = \frac{1}{\gamma^2} + \beta^2,$$

where the first equation comes from the condition $\|w_i - w_j\|^2 = $constant, $\forall i \neq j$, and the second from $\|w_i\|^2 = 1, \forall i$. These two equations have the simultaneous solutions

$$\beta = -\frac{1}{n},$$
$$\gamma = \frac{n}{\sqrt{n^2 - 1}},$$

which ends the proof. □

The performance of a pattern search algorithm (Section 2.3.1) depends, among other things, on how well the search directions approximate the direction of steepest descent. The search directions are required to constitute at least a positive basis in $\mathbb{R}^n$. Therefore, it is of interest to find an upper bound on the angle, or, equivalently, a lower bound on the cosine of the angle, between an arbitrary vector and any vector in a positive basis. For a maximal positive basis consisting of the standard unit vectors and their opposites, the lower bound for the cosine is $1/\sqrt{n}$ (see Figure 2.3 and [36]). The following theorem states the corresponding result for a normalized regular simplex (which, according to Theorem 7, is a minimal positive basis).

Theorem 10. *For a minimum positive basis consisting of a normalized regular simplex* $\{v_1, \cdots, v_{n+1}\}$ *in* $\mathbb{R}^n$ *we have*

$$\min_{\|x\|=1} \max_i (x \bullet v_i) = \frac{1}{n},$$

and the minimum is attained when $x = -v_i$*, for any* i*.*

Proof. From Lemma 8 we have $(-v_i) \bullet v_j = 1/n$, $\forall i \neq j$, and it only remains to show that $\max_i x \bullet v_i \geq 1/n$ for any unit vector x. For a contradiction, suppose $\max_i x \bullet v_i < 1/n$ for some x of unit length. Lemma 5 assures that (after possibly a reordering of the v_i:s) we may write $x = \sum_{i=2}^{n+1} c_i v_i$, $c_i \geq 0$. Let

$$k = \arg \max_{i=2,\ldots,n+1} c_i.$$

Then, using the (contradictive) assumption that $x \bullet v_i < 1/n$ for every i, we get

$$\begin{aligned} \|x\|^2 &= x \bullet \Big(\sum_{i=2}^{n+1} c_i v_i \Big) \\ &< \frac{1}{n} \sum_{i=2}^{n+1} c_i \leq \frac{c_k}{n} \sum_{i=2}^{n+1} = c_k \end{aligned}$$

Hence $c_k > 1$ and it follows that

$$\begin{aligned} x \bullet v_k &= c_k - \frac{1}{n} \sum_{i=2, i\neq k}^{n+1} c_i \\ &\geq c_k \Big(1 - \frac{1}{n} \sum_{i=2, i\neq k}^{n+1} \Big) > \frac{1}{n} \end{aligned}$$

which contradicts the assumption on x. This concludes the proof.

□

A.2.2 On a question of Powell

Finally, we consider the classical simplex method in [16]. This method proceeds by generating a sequence of simplices, where each simplex has all but one vertex in common with the preceding simplex. The objective function is evaluated at all vertices $\{v_0, \ldots, v_n\}$ of the current simplex. The new vertex v is found through the reflexion of the vertex with the highest function value, say v_m, in the plane through the others, i.e. ,

$$v = -v_m + \frac{2}{n} \sum_{i\neq m}^{n} v_i. \tag{A.4}$$

For $n = 2$, this can be seen in Figure A.1, and the simplices obtained from such reflexions form a regular pattern (Figure A.2).

Hence, the method can only produce a finite sequence of different simplices in a bounded domain of $\mathbb{R}^2$. In [32], Powell shows that repeated reflexions may

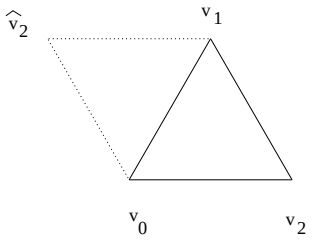


Figure A.1: Reflexion of a regular simplex.

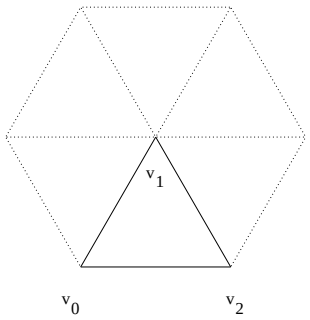


Figure A.2: In $\mathbb{R}^2$ the sequence of points generated by repeated reflexions of any of the vertices of a simplex stays on a lattice. Hence, if one vertex, v_1 in the figure, is kept fixed, reflexion of the remaining two vertices produce only a finite number of new points (namely 4).

produce an infinite sequence of different simplices when $n = 3$ and asks whether the same thing could happen in higher dimensions. We show that it indeed can, at least when n is not a power of 2. As mentioned in the introduction to this chapter, this result indicates that a simplex method may behave quite differently from a pattern search method.

Let $\{v_0, \ldots, v_n\}$ be a normalized regular simplex in $\mathbb{R}^n$ (see Definition 1). Let $n = ab > 3$, where a is any positive integer and b is any prime number different from 2 (i.e., n is not a power of 2). We will consider the sequence of simplices generated by keeping all but two of the vertices in the simplex fixed and repeatedly apply (A.4) to the other two. More precisely, the k:th simplex ($k > 1$) has one vertex w_k different from the $(k-1)$:th simplex, where

$$\begin{cases} w_0 = v_0,\ w_1 = v_1, \\ w_k = -w_{k-2} + \frac{2}{n}(w_{k-1} + \sum_{i=2}^{n} v_i), \quad k > 1. \end{cases} \tag{A.5}$$

All of the simplices in the sequence generated in this way will be different if all of the vertices $\{w_k\}_{k=0}^{\infty}$ are different. If we let

$$c = \frac{1}{n-1}(v_2 + v_3 + \ldots + v_3)$$

and define

$$u_k = w_k - c,\ k = 0, 1, \ldots,$$

the sequence in Equation (A.5) transforms to

$$u_0 = v_0 - c,\ u_1 = v_1 - c \qquad \text{and} \qquad u_k = \frac{2}{n}u_{k-1} - u_{k-2}, k > 1.$$

We see that the points u_k stay in the two dimensional subspace spanned by u_0, u_1. From Lemma 8 we have $v_i \bullet v_j = -1/n$, $\forall i \neq j$, and it is then easy to compute

$$\|u_0\|^2 = \|u_1\|^2 = \frac{n+1}{n-1}, \qquad \text{and} \qquad u_0 \bullet u_1 = \frac{n+1}{n(n-1)}.$$

It follows that u_1 is the rotation R of u_0 by an angle θ in the plane spanned by u_0, u_1, where

$$\cos(\theta) = \frac{u_0 \bullet u_1}{\|u_0\|\,\|u_1\|} = \frac{1}{n}.$$

Furthermore,

$$u_2 = \frac{2}{n}u_1 - u_0 = 2\frac{u_0 \bullet u_1}{\|u_1\|^2}u_1 - u_0,$$

so that u_2 is the reflexion of u_0 in the line through u_1. This means that u_2 is the rotation R of u_1. By induction we get $u_k = Ru_{k-1}$.

Now, the elements in $\{w_k\}$ are distinct if and only if the elements in $\{u_k\}$ are distinct. If two elements in $\{u_k\}$ are equal, we must have $R^m u_0 = u_0$ for some positive integer m. This happens exactly when $m\theta/2\pi$ is an integer. Suppose this is so. We expand $\cos(m\theta)$ in powers of $\cos(\theta)$, the highest power being of order m,

$$\begin{aligned} 1 = \cos(m\theta) &= 2^{m-1}\cos^m(\theta) + \alpha_{m-2}\cos^{m-2}(\theta) + \dots \\ &= 2^{m-1}\frac{1}{n^m} + \alpha_{m-2}\frac{1}{n^{m-2}} + \dots, \end{aligned}$$

where all α_i are integers. Multiplying both sides by n^m gives

$$2^{m-1} + n^2\alpha_{m-2} + n^4\alpha_{m-4} + \dots = n^m,$$

where all but the first term are divisible by b. We have thus reached a contradiction under the assumption that $m\theta/2\pi$ is an integer, and so we have an infinite sequence of distinct simplices in a bounded region of $\mathbb{R}^n$.

Appendix B

NURB curves

NURB curves are piecewise defined rational polynomials. They provide a unified mathematical basis for the representation of both analytic shapes, such as conic sections, as well as complex engineering shapes. Furthermore, they allow for efficient implementations and fast algorithms. NURB curves include B-splines and rational and nonrational Bézier curves as special cases.

We first define the B-spline basis functions and discuss some of their properties. We then define the NURB curves, and to show their use we give three examples of applications. These include the representation of conic sections, the connection of two curves, and interpolation. Most of the material below is taken from [30] and [33].

B.1 B-spline basis functions

Let $[a, b]$ be a finite interval and $p \geq 0$ an integer. Let $\{v_i\}_{i=0}^{k}$ be a strictly increasing sequence of real numbers, called *breakpoints*, such that

$$a = v_0 < v_1 < \ldots < v_k = b.$$

With each v_i we associate a positive integer m_i and define the *knot vector* U by

$$U = \{u_0, \ldots, u_m\} = \{\underbrace{v_0, \ldots, v_0}_{m_0}, \ldots, \underbrace{v_k, \ldots, v_k}_{m_k}\},$$

where $m_0 = m_k = p + 1$ and $1 \leq m_i \leq p$ for $i = 1, \ldots, k-1$. m_i is called the *multiplicity* of the breakpoint v_i and also of the knot u_j, if $u_j = v_i$.

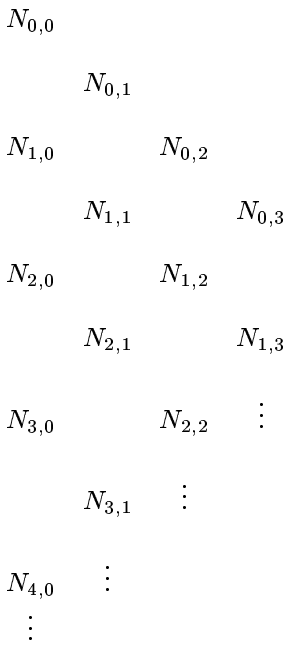
We recursively define the $m - p$ B-spline basis functions $N_{i,p}$ on the knot vector U by

$$N_{i,0}(u) = \begin{cases} 1, & \text{if } u \in [u_i, u_{i+1}), \\ 0, & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u).$$

If a denominator involving knot differences becomes zero, the quotient is defined to be zero.

Many important properties for the B-spline basis functions can be derived by considering their recursive structure, which is illustrated below.



We now list some of the most important properties of the B-spline basis functions.

Proposition 11. *(Properties of B-spline basis functions.)*
P1. $N_{i,p}$ is a piecewise polynomial of degree p. It is a polynomial of degree p in the interior of a breakpoint interval (v_j, v_{j+1}).
P2. $N_{i,p}$ is at least $p - m_j$ times continuously differentiable at a breakpoint v_j.
P3. Compact support and non-negativity: $N_{i,p}(u) = 0$ outside $[u_i, u_{i+p+1}]$ and $N_{i,p}(u) > 0$ on (u_i, u_{i+p+1}). Note that $u_i < u_{i+p+1}$ since $m_j < p + 1$ for $j = 1, \ldots, k-1$, whence $N_{i,p} \neq 0$.
P4. Partition of unity:

$$\sum_{i=0}^{m-p-1} N_{i,p}(u) = 1, \ u \in [a, b].$$

P5. $\{N_{i,p}\}_{i=0}^{m-p-1}$ forms a basis for the linear space V of all piecewise polynomial functions of degree p which are $p - m_j$ times continuously differentiable at the interior breakpoints v_j, $j = 1, \ldots, k-1$.

Proof. P1 and P2 corresponds to Theorem 4.14 in [33]. P3 is proved in Theorem 4.17 in the same reference. P4 follows easily from Theorem 4.20. To prove P5, we first note that the basis functions are indeed members of V (P1 and P2), and that $dim(V) = m - p$. It only remains to prove that the $N_{i,p}$:s are linearly independent. To this end, suppose that

$$\sum_{i=0}^{m-p-1} c_i N_{i,p}(u) = 0, \ u \in [a, b].$$

P3 ensures that the support of $N_{i,p}$ contains a non-empty knot interval (u_j, u_{j+1}) and that $N_{i,p}(u) > 0$ there. Considering the non-negativity of the basis functions (P3 again), it follows that $c_i = 0$. □

B.2 NURB curves

A NURB (*non-rational uniform B-spline*) curve, of degree p on the interval $[0, 1]$ is defined by (see [30])

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}, \qquad 0 \le u \le 1, \tag{B.1}$$

where $\mathbf{P}_i$ are the *control points*, $w_i > 0$ the *weights* and $N_{i,p}$ the p:th degree B-spline basis function (described in the previous section) defined on some knot vector U.

We now list some relevant properties of a NURB curve $\mathbf{C}$. Most of them can be deduced directly from Proposition 11, or else found in [30].

P0. Algorithms for manipulating NURB curves are fast and numerically stable.

P1. The degree, p, number of control points, $n+1$, and number of knots, $m+1$, are related by

$$m = n + p + 1.$$

P2. Though a NURB curve does not in general interpolate the control points, one always has end point interpolation, $\mathbf{C}(0) = \mathbf{P}_0$, $\mathbf{C}(1) = \mathbf{P}_n$.

P3. $\mathbf{C}$ is infinitely differentiable in the interior of a knot interval and at least $p - m_j$ times differentiable at a knot of multiplicity m_j.

P4. It lies in the convex hull of the control polygon.

P5. The NURB curves contain B-spline curves (all $w_i = 1$, see P4 in Proposition 11), rational Bézier curves ($U = \{0, \ldots, 0, 1, \ldots, 1\}$) and Bézier curves ($w_i = 1$ and $U = \{0, \ldots, 0, 1, \ldots, 1\}$) as special cases.

P6. If the control point $\mathbf{P}_i$ is moved or the weight w_i changed, it affects only the part of the curve corresponding to the interval $[u_i, u_{i+p+1}]$.

P7. The effect of increasing one weight w_i in proportion to the others is to pull the curve towards the control point $\mathbf{P}_i$.

Because of these properties, NURB curves provide an excellent and flexible tool for the representation of geometric shapes. In particular, property P6 allows for local control and variation of the curve, which makes it possible to use NURB curves as a design tool or in a shape optimization procedure.

B.3 Applications

We discuss three applications of NURB curves. First we discuss how conic sections and in particular circular arcs can be represented by NURB curves. We then describe how to represent two piecewise defined curves by one curve. Finally, we outline how to use NURB curves for interpolation in general and for cubic spline interpolation in particular.

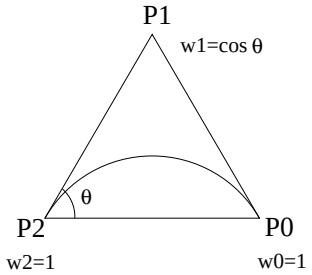


Figure B.1: A Bézier arc.

B.3.1 Circular arcs

In [30] it is shown how the NURB curves can be used to represent circular arcs. More generally, it is shown that the quadratic rational Bézier curve,

$$\mathbf{C}(u) = \frac{(1-u)^2 w_0 \mathbf{P}_0 + 2u(1-u)w_1\mathbf{P}_1 + u^2 w_2 \mathbf{P}_2}{(1-u)^2 w_0 + 2u(1-u)w_1 + u^2 w_2}, \quad u \in [0,1], \tag{B.2}$$

represents a conic (ellips, parabola or hyperbola). This is done by means of a change of coordinates. Equation B.2 is obtained from the NURB curve definition by taking $p = 2$ and $U = \{0,0,0,1,1,1\}$.

To represent a circular arc, let $\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2$ be an isosceles triangle such that $\mathbf{P}_0\mathbf{P}_1 = \mathbf{P}_1\mathbf{P}_2$. Let $\theta = \angle\mathbf{P}_1\mathbf{P}_2\mathbf{P}_0$ and take $w_0 = w_2 = 1$, $w_1 = \cos\theta$. Then $\mathbf{C}(u)$ is a parameterization of the circular arc from $\mathbf{P}_0$ to $\mathbf{P}_2$ that passes through the inside of the triangle $\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2$ and has $\mathbf{P}_0\mathbf{P}_1$ and $\mathbf{P}_1\mathbf{P}_2$ as tangents (Figure B.1).

B.3.2 Connecting two curves

Given two NURB curves,

$$\mathbf{C}_j(u) = \frac{\sum_{i=0}^{n_j} N_{i,p}^j w_i^j \mathbf{P}_i^j}{\sum_{k=0}^{n_j} N_{k,p}^j w_k^j}, \quad u \in [a_j, b_j],\ j = 1,2,$$

such that $\mathbf{C}_1(b_1) = \mathbf{C}(a_2)$ and $w_{n_1}^1 = w_0^2$, we can connect them into one curve. More precisely, we can construct a continuous curve $\mathbf{C}(u)$ on $[a_1, b_1 + (b_2 - a_2)]$ such that $\mathbf{C}(u) = \mathbf{C}_1(u)$ on $[a_1, b_1]$ and $\mathbf{C}(u + b_1 - a_2) = \mathbf{C}_2(u)$ on $[a_2, b_2]$.

Suppose first that $b_1 = a_2$. To define $\mathbf{C}$ we have to define the control points, the weights and a knot vector. We choose

$$\begin{array}{llll} \mathbf{P}_i = \mathbf{P}_i^1, & w_i = w_i^1, & i = 0, \ldots, n_1 \\ \mathbf{P}_{i+n_1} = \mathbf{P}_i^2, & w_{i+n_1} = w_i^2, & i = 1, \ldots, n_2. \end{array}$$

If the knot vector for $\mathbf{C}_j$ is $U_j = \{u_0^j, \ldots, u_{m_j}^j\}$, we define the knot vector for $\mathbf{C}$ by

$$U = \{u_0^1, \ldots, u_{m_1-1}^1, u_{p+1}^2, \ldots, u_{m_2}^2\}.$$

Hence U contains $m + 1 = m_1 + m_2 - p$ knots.

We now show that $\mathbf{C}$ has the desired property. Note that, by the definition of a knot vector, $u^1_{m_1-p-1} < u^1_{m_1-p} = \ldots = u^1_{m_1-1} < u^2_{p+1}$ and hence (note that $m_1 - p - 1 = n_1$)

$$u_{n_1} < u_{n_1+1} = \ldots = u_{n_1+p} < u_{n_1+p+1}.$$

Using the B-spline basis function property P3, we find that if $i < n_1$ then $N_{i,p} = N^1_{i,p}$ and its support belongs to $[a_1, b_1]$. Similarily, if $i > 0$ then $N_{n_1+i,p} = N^2_{i,p}$ and its support is in $[a_2, b_2]$. For $i = n_1$ we have

$$N_{n_1,p} = N^1_{n_1,p} + N^2_{0,p},$$

and $N_{n_1,p}$ equals $N^1_{n_1,p}$ when restricted to $[a_1, b_1]$, and $N^2_{0,p}$ when restricted to $[b_1, b_2]$. Thus, for u in $[a_1, b_1]$ we have

$$\begin{aligned}
\mathbf{C}(u) &= \frac{\sum_{i=0}^{n_1+n_2} N_{i,p} w_i \mathbf{P}_i}{\sum_{j=0}^{n_1+n_2} N_{j,p} w_j} \\
&= \frac{\sum_{i=0}^{n_1} N_{i,p} w_i \mathbf{P}_i}{\sum_{j=0}^{n_1} N_{j,p} w_j} \\
&= \frac{\sum_{i=0}^{n_1} N^1_{i,p} w^1_i \mathbf{P}^1_i}{\sum_{j=0}^{n_1} N^1_{j,p} w^1_j} \\
&= \mathbf{C}_1(u), \quad u \in [a_1, b_1].
\end{aligned}$$

Analogously, it is shown that $\mathbf{C}$ restricted to $[a_2, b_2]$ equals $\mathbf{C}_2$.

If $b_1 \neq a_2$ we define a translated knot vector $\tilde{U} = U_2 + b_1 - a_2$. It is then easy to show that the corresponding curve $\tilde{\mathbf{C}}_2$ defined on $[b_1, b_2 + b_1 - a_2]$ is such that $\tilde{\mathbf{C}}_2(u) = \mathbf{C}_2(u - b_1 + a_2)$, and we may proceed as above.

B.3.3 Interpolation and parameterizations

The NURB curve does not in general interpolate its control points. If we want to interpolate a given set of points $\{\mathbf{Q}_k\}, k = 0, \ldots, n$ with a pth-degree B-spline curve $\mathbf{C}$ at parameter values $\bar{u}_k$, we can select a knot vector U and solve the system of linear equations given by

$$\mathbf{Q}_k = \mathbf{C}(\bar{u}_k) = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k)\mathbf{P}_i, \quad k = 0, \ldots, n, \tag{B.3}$$

where the control points $\mathbf{P}_i$ are the unknowns. The choice of U and $\bar{u}_k$ will affect both the shape and the parameterization of the curve, as noted in [30]. How to choose U in order to avoid a singular system of equations is discussed in [30], section 9.2.

Clearly, the choice of $\bar{u}_k$ affects the parameterization. To approximate the arc length (uniform) parameterization, it is common to choose $\bar{u}_k$ as the chord length along the polygon defined by the $\mathbf{Q}_k$:s, as explained in [30], section 9.2.

In particular, we can use NURB curves for the traditional cubic spline interpolation. Take $\bar{u}_k$ as above, $p = 3$ and the knot vector

$$U = \{\bar{u}_0, \bar{u}_0, \bar{u}_0, \bar{u}_0, \bar{u}_1, \ldots, \bar{u}_{n-1}, \bar{u}_n, \bar{u}_n, \bar{u}_n, \bar{u}_n, \}.$$

This means that the interpolation occurs at the knots. The corresponding $n+3$ B-spline basis function constitute, according to P5 in Proposition 11, a basis for the space of cubic spline functions on $[\bar{u}_0, \bar{u}_n]$ with knots (nodes) $\bar{u}_k$. Setting up the system (B.3) leaves two degrees of freedom that can be used to specify the end derivatives.

Bibliography

[1] D.J. Acheson, *Elementary Fluid Dynamics*, Oxford University Press, Oxford, 1990.

[2] N. Alexandrov, J.E. Dennis, M.L. Lewis, V. Torczon, A trust region framework for managing the use of approximation models in optimization, *Structural Optimization*, 15:1, pp. 16–23, 1998.

[3] P. Bansod, P. Bradshaw, The flow in s-shaped ducts, *Aeronautical Quarterly*, The Royal Aeronautical Society, 23, pp. 131–140, May 1972.

[4] J.-F. M. Barthelemy, R. T. Haftka, Approximation concepts for optimum structural design - a review, *Structural Optimization*, 5, pp. 129–144, 1993.

[5] A.J. Booker, J.E. Dennis, P.D. Frank, D.B. Serafini, V. Torczon, M.W. Trosset, A rigorous framework for optimization of expensive functions by surrogates, *Structural Optimization*, 17:1, pp. 1–13, 1999.

[6] A.J. Booker, J.E. Dennis, P.D. Frank, D.B. Serafini, V. Torczon, Optimization using surrogate objectives on a helicopter test example, *Computational Methods in Optimal Design and Control*, Birkhauser, Boston, pp. 49–58, 1998.

[7] G.E.P. Box, K.B. Wilson, On the experimental attainment of optimum conditions, *J. Roy. Statist. Soc. Ser. B XIII*, pp. 1–45, 1951.

[8] A.R. Conn, Ph.L. Toint, An algorithm using quadratic interpolation for unconstrained derivative free optimization, *Nonlinear Optimization and Applications*, Plenum Publishing, New York, pp. 27–47, 1996.

[9] A.R. Conn, K. Scheinberg, Ph.L. Toint, A derivative free optimization algorithm in practice, Proceedings of the AIAA St Louis Conference, 1998.

[10] A.J. Chorin, J.E. Marsden, *A Mathematical Introduction to Fluid Mechanics*, 3rd ed., Springer-Verlag, New York, 1993.

[11] A. Dadone, B. Grossman, Progressive optimization of inverse fluid dynamic design problems, *Computers & Fluids*, 29, pp. 1–32, 2000.

[12] O.L. Davies, *The Design and Analysis of Industrial Experiments*, Hafner Publishing Company, New York, 1954.

[13] J.E. Dennis, V. Torczon, Derivative-free pattern search methods for multidisciplinary design problems, *Proceedings of the AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, Florida, 1994.

[14] J.E. Dennis, V. Torczon, Managing approximation models in optimization, *Multidisciplinary Design Optimization: State of the Art*, SIAM, Philadelphia, 1996.

[15] *Fluent5 User's Guide Volume 1–4*, Fluent Incorporated, 1998.

[16] G.R Hext, F.R. Himsworth, W. Spendley, Sequential application of simplex designs in optimisation and evolutionary operation, *Technometrics*, 4, pp. 441–461, 1962.

[17] J.O. Hinze, *Turbulence*, 2nd ed., McGraw-Hill, 1975.

[18] W. Hock, K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, 1981.

[19] R. Hooke, T.A. Jeeves, Direct search solution of numerical and statistical problems, *Journal of the Association for the Computing Machinery*, 8, pp. 212–229, 1961.

[20] W.M. Lai, D. Rubin, E. Krempl, *Introduction to Continuum Mechanics*, 3d ed., Butterworth-Heinemann Ltd, Oxford, 1993.

[21] B.E. Launder, D.B. Spalding, The numerical computation of turbulent flows, *Computer Methods in Applied Mechanics and Engineering*, 3, pp. 269–289, 1974.

[22] R.M. Lewis, V. Torczon, Rank ordering and positive bases in pattern search algorithms, Technical Report 96-71, Institute for Computer Applications in Science and Engineering, Nasa Langley Research Center, Hampton, VA, 1996.

[23] R.M. Lewis, V. Torczon, Pattern search algorithms for bound constrained minimization, Technical Report 96-20, Institute for Computer Applications in Science and Engineering, Nasa Langley Research Center, Hampton, VA, 1996.

[24] R.M. Lewis, V. Torczon, Pattern search methods for linearly constrained minimization, *SIAM Journal of Optimization*, 10, pp. 917–941, 2000.

[25] R.M. Lewis, V. Torczon, M.W. Trosset, Direct search methods: then and now, *Journal of Computational and Applied Mathematics*, 124, pp. 191–207, 2000.

[26] W. Malalasekera, H.K. Versteeg, *An Introduction to Computational Fluid Dynamics. The Finite Volume Method*, Addison Wesley Longman Limited, 1995.

[27] R.R Mankbadi, Turbulence models for CFD, in *Computational Fluid Dynamics Techniques*, edited by W.G. Habashi, M.M. Hafez, Gordon and Breach Publishers, 1995.

[28] K.I.M. McKinnon, Convergence of the nelder-mead simplex method to a nonstationary point, *SIAM Journal on Optimization*, 9, pp. 148–158, 1998.

[29] J.A. Nelder, R.Mead, A simplex method for function minimization, *Computer Journal*, 7, pp. 308–313, 1965.

[30] L. Piegl, W. Tiller, *The NURBs Book*, 2nd ed., Springer-Verlag Berlin Heidelberg New York, 1997.

[31] M.J.D. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, *Advances in Optimization and Numerical Analysis*, Kluwer Academic, Dordrecht, pp. 51–67, 1994.

[32] M.J.D. Powell, Direct search algorithms for optimization calculations, *Acta Numerica*, 7, pp. 287–336, 1998.

[33] L. L. Schumaker, *Spline Functions: Basic Theory*, John Wiley and Sons, 1981.

[34] G.J. Seil, *Computational Fluid Dynamics Investigations and Optimisation of Marine Waterjet Propulsion Unit Inlet Design*, PhD Thesis, Department of Mechanical and Manufacturing Engineering, The University of New South Wales, 1997.

[35] Y.S. Sherif, B.A. Boice, An efficient algorithm for solving the unconstrained nonlinear multivariable minimization problems, *Advances in Engineering Software*, 17, pp. 29–37, 1993.

[36] V. Torczon, On the convergence of pattern search algorithms, *SIAM Journal on Optimization*, 7, pp. 1–25, 1997.

[37] D.J. Tritton, *Physical Fluid Dynamics*, 2nd ed., Oxford University Press, Oxford, 1988.

[38] M.W. Trosset, V. Torczon, Numerical optimization using computer experiments, Technical Report 97-38, ICASE, NASA Langley Research Center, Hampton, VA, 1997.

[39] F.M. White, *Viscous Flow*, McGraw-Hill, 1974.

[40] S. Ågren, *Parametric Geometry Representation of a Waterjet Duct and Initialisation of the Related CFD Problem*, Master Thesis, Department of Mathematics, Chalmers University of Technology, Göteborg, 1999.