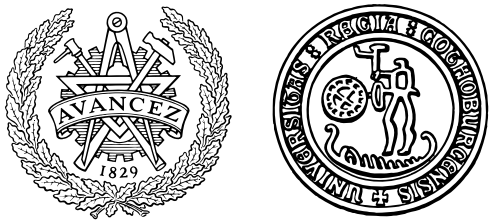
Thesis for the Degree of Licentiate of Philosophy

Derivative-Free Optimization of a Waterjet Inlet Duct Model

Jakob Hultén



Department of Mathematics
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
Göteborg, October 2000

Derivative-Free Optimization of a Waterjet Inlet Duct Model
Jakob Hultén

ISSN 0347-2809/No 2000:61
Department of Mathematics
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-7721000

Matematiskt Centrum
Göteborg, Sweden 2000

Abstract

We apply computational fluid dynamics (CFD) techniques and numerical optimization methods to the engineering problem of optimal design of a waterjet inlet duct model. The objective function is very time-consuming to evaluate because it depends on the numerical solution of a turbulent flow problem. No explicit information is given about the gradient. Such objective functions demand optimization methods that do not calculate derivatives.

We describe a well-known class of derivative-free methods called positive basis pattern search methods. Their general convergence analysis requires the objective function to be continuously differentiable. The necessity of this condition is shown by an example of a function that is Lipschitz continuous and for which coordinate search with a fixed step length converges to a non-stationary point.

A natural question is how many search directions one should have in each iteration with a pattern search algorithm in $\mathbb{R}^n$. Using few directions reduces the worst number of function evaluations per iteration whereas using many directions leads to a better approximation of the direction of steepest descent. We analyze the cases when the search directions constitute a minimal and a maximal positive basis, and find the latter to be preferable.

A solution strategy, based on computer simulations, is presented for the waterjet inlet duct design problem. Numerical tests show how a significant decrease in the objective function is obtained after 10 to 20 function evaluations. We also investigate how the sizing of the mesh can be used to construct surrogate functions (inexpensive models of the objective function), and discuss how they can be incorporated into the optimization procedure.

Finally, we use some basic theory of positive linear dependence to derive results about simplices and minimal positive bases.

Keywords: optimal design, derivative-free optimization, direct search, pattern search, positive linear dependence, positive basis, regular simplex, surrogate function, waterjet propulsion, fluid dynamics, parametric model

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 supervisor Michael Patriksson at the Department of Mathematics at Chalmers for always taking his time to discuss problems and ideas and giving me many helpful comments on the manuscript. I would also like to express my gratitude to Johan Lennblad at Caran/VM-data who has been my "industrial" supervisor and a great support through his advises, discussions and kind encouragement. He was also the initiator of the project. Many thanks also to Gregory Seil at Kamewa, for patiently sharing with me his broad knowledge of waterjets and computational fluid dynamics, and Lennart Berghult, at Kamewa as well, for supporting the project. I thank Sara Ågren for lending me a C++ code she had done which was important for the geometric modelling of the problem.

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

First and last I am forever grateful to 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

Chapter 1

Introduction

In this work we consider derivative-free optimization methods, fluid dynamics and their application to the problem of optimal shape design of a waterjet propulsion unit inlet model. The model flow performance is determined by computational fluid dynamics (CFD) techniques. In this chapter, we first give a background to waterjet propulsion. We then give a preliminary formulation of the problem and finally a short overview of the report.

1.1 Background

Marine waterjet propulsion is used for the propulsion of many kinds of marine vehicles, ranging from pleasure boats to hydrofoil ferries. A waterjet drive principally consists of an inlet duct and a pump. Sea water enters the duct, the pump increases its momentum and gives a thrust when the water exits the duct as a jet. This thrust propels the boat. Steering is accomplished through the deflection of the exiting jet of water in different directions.

In Figure 1.1 is shown a waterjet propulsion unit with a so called ram-type inlet [28]. It is typically used on hydrofoil crafts but has also been used on surface-effect ships. There are also units with what is called a flush-type inlet, which is used on displacement, semi-displacement and planning hulls.

Waterjet propulsion offers, according to [28], some benefits not present with conventional propeller-driven units. Among these benefits are

- Safety: no externally turning propellers implies a higher safety for divers etc. With a flush-type inlet there is small risk of damage to the boat from floating debris.

- Manoeuvrability: a large manoeuvring thrust can be developed even at low speeds. Jet driven vessels are hence appropriate for use in recovery

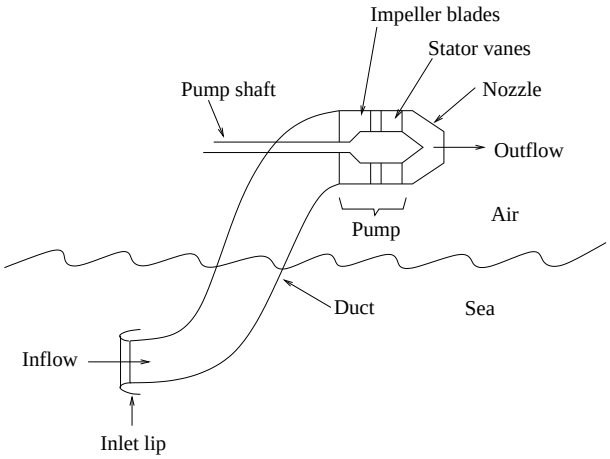


Figure 1.1: A waterjet propulsion unit with ram-type inlet duct.

operations.

- Environment: reduced underwater noise and on-board noise and vibrations.

- Reduced appendage drag.

Also some disadvantages can be noted. There is a risk for plugging of the inlet opening because of floating debris. Biological growth within the inlet duct may reduce the efficiency because of pressure losses. Also, the ingestion of air into the inlet leads to reduced performance. This is a problem when designing waterjet propulsion systems for planning hulls.

When designing waterjet propulsion systems one must avoid *cavitation* since this leads to reduced thrust or even a stop. Cavitation means that the internal pressure in the water flowing through the waterjet inlet duct locally is below the vapour pressure. In an experimental setup, cavitation can be seen as small bubbles, produced close to the inlet duct walls and spread by the flowing water into the pump. Cavitation may occur at any speed. A waterjet for high-speed cruise conditions is designed so that the flow diffuses into the inlet duct and the pressure in the inlet duct is higher than the free-stream pressure. Under these conditions cavitation is unlikely to occur unless the inlet duct has sharp bends. At low speeds, the inlet duct pressure is lower than the free-stream pressure and this pressure drop increases the risk of cavitation. The issue of cavitation at low speeds is particularly important if the maximum in the resistance curve of the ship occurs at a low speed.

Since low speeds may correspond to the ship manoeuvring in a harbour, it is important to avoid cavitation also at these conditions even for a ship with a high-speed cruise condition.

As already mentioned, one source to the problem of cavitation is the bends of the inlet ducting. The purpose of the inlet duct is to lead sea water to the pump. For technical reasons concerning power transmission, it is preferable to have the pump above the sea level (as a benefit, the appendage drag is reduced since a reduced part of the inlet duct will be in the sea). Thus the inlet duct *has to* bend; the question is *how*.

1.2 The problem

This report is about a problem posed by Kamewa, a company in Kristinehamn, Sweden, that manufactures, among other things, waterjet propulsion systems. To design a waterjet propulsion system one has to take many factors into account such as the shape of the inlet duct, the pump, waterjet-hull interactions etc ([28]). However, Kamewa formulated the following simplified problem:

> **Aim 1:** Find a shape of a waterjet inlet duct model that minimizes the risk for cavitation.

The flow performance of the waterjet inlet duct was to be determined by the commercial computational fluid dynamics (CFD) program Fluent5.0.

We may divide the problem into three subproblems:

- Geometric modelling: Construct a simplified parametric pipe model of the waterjet inlet duct geometry.
- Flow modelling: Use CFD techniques to determine the flow performance of the pipe model.
- Optimization: Apply an appropriate optimization algorithm to find the optimal shape of the parametric geometry.

The problem is very complex, mostly because the objective function depends on the solution to the pipe flow problem which, in practice, has to be found by computationally intensive (time-consuming) computer simulations.

The following aim is therefore more realistic than the one above.

> **Aim 2:** Find a robust and efficient procedure that starts from an initial design and automatically and iteratively searches for a better one.

By "robust" we mean a procedure that does not break down once in a while (for example because of instabilities in the computational geometric or flow modelling) and that gives reliable results. By "efficient" we mean a procedure that gets close to a local minimum within either a reasonable number of function evaluations (less than, say, 100) or a reasonable amount of time (a couple of hours).

1.3 Design optimization

What we have just seen is a typical example of the common engineering problem of optimal design. Computer simulations is an increasingly important and popular tool for solving such problems. 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 many different designs in order to choose the best one. This possibility leads to an optimization problem where the objective function, because of the inherent computer simulation, becomes computationally expensive to evaluate, has almost unknown structure, gives no explicit information about derivatives and contains high frequency distortions. Hence these problems demand for *derivative-free*, or *direct*, search methods which are optimization methods that make no explicit use of derivatives.

Engineering practise [2] for tackling these problems 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 build a model, or *surrogate*, function. Then the optimization is performed on this surrogate function, which, of course, is comparatively much less expansive and also provides explicit derivatives. The main question is what to do next, 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 on derivative-free methods and on how to use surrogate functions in the optimization process. For example, J.E. Dennis and V. Torczon [31] suggest how to use surrogate functions in the

pattern search algorithms developed by Torczon in [30]. There is no explicit discussion of the choice of surrogate function in [31] 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 surrogate function frameworks for optimization, also built upon the pattern search method, are suggested in [33] and [3]. The objective function is regarded as the outcome of a stochastic process, a fiction that enables the construction of algebraic approximation models from samples using statistical tools. In [3], some of the function evaluations are done in order to update the model, a procedure referred to as a *balanced search*.

A different approach is found in [5], 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 the considerations of strict geometric properties of the set of model-building points.

An application of design optimization using surrogate functions can be found in [4] and [6]. There, the algorithms presented in [3] and [5], are applied to a helicopter test problem with 31 design variables. With the best methods, considerably improved designs where found after about $100 - 200$ function evaluations. This can be compared with a genetic algorithm tested in [3] that, even though it used many more evaluations, gave a more moderate decrease in the objective. As already mentioned, the surrogate functions used are algebraic models constructed from samples of the objective function, but in [3] it is pointed out that simplified simulations are used since qualitatively they are enough to reflect the real physical process. It is also noted that a motivation for using direct methods instead of gradient methods is that the latter tends to result in refined designs and not radically new ones.

The above mentioned research on how to use surrogate functions in the optimization procedure has inspired this work. However, instead of using algebraically constructed models we investigate other kinds of surrogate functions. The reason is that the approach using algebraic models treats the objective as a "black box" whereas we actually do know something about its structure. Our aim is to construct surrogates based on simplifications of the computer simulations implicit in the objective.

1.4 Contributions and outline of the report

The rest of the report consists of four chapters. The following two chapters and the last one are mainly theoretical, while Chapter 4 is more practical.

Chapter 2 is about derivative-free optimization. We discuss a class of direct methods called *positive basis pattern search methods* ([18], [30]).

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

In Chapter 4, which contains the main part of the work done, we use the theory discussed in Chapters 2 and 3 to model and numerically solve the problem described above. We also investigate how the sizing of the mesh can be used to construct surrogate functions. The chapter has been written as a report to be read by people at Kamewa. We therefore present not only the final results but also the details about the steps along the way.

Chapter 5, finally, is quite separate from the rest of the report. It is the most "mathematical" chapter. Inspired by a couple of questions arising in connection to optimization we apply some theory of positive linear dependence to regular simplices and minimal positive bases.

There are three main mathematical contributions in this report. The first is the example in Section 2.1.3 which shows the necessity of the objective function to be C^1 for the convergence of pattern search methods. The second is the solution to the minmax problem in Theorem 9 in Chapter 5. This problem arises as a natural question concerning the efficiency of a positive basis in a pattern search method. The third is the construction (at the end of Chapter 5) of a bounded infinite sequence of distinct regular simplices of constant size. This shows that the iterates generated by the classical simplex method [12] do not have the same favourable algebraic structure as those generated by pattern search methods.

Chapter 2

Optimization without calculating derivatives

In this chapter we consider so called *derivative-free*, or *direct search*, methods, which are optimization algorithms that do not make explicit use of derivatives. These algorithms are suitable for the kind of problem given in the Introduction. We present a class of direct methods called *pattern search methods* ([30],[18]), discuss their properties and present a basic convergence result. We also discuss so called *minimal* and *maximal positive bases* and, finally, how to use approximations, *surrogates*, of the objective function in the optimization.

Consider the general finite dimensional optimization problem

$$\min_{x \in X} f(x) \tag{2.1}$$

where $X \subseteq \mathbb{R}^n$. Many methods for solving this problem have been presented. A certain method is often constructed with a certain kind of objective function in mind, such as the simplex method for linear functions, the conjugate gradient method for quadratic functions etc. In engineering applications such as optimal design mentioned in the Introduction, the objective function is often based on the outcome of complicated computer simulations, which causes it to have the following properties:

i) The derivatives of f, even if they do exist, are often not available.

ii) f is expensive to evaluate at any point x. (By expensive we mean for example time-consuming.)

iii) The idealized function f is contaminated with high-frequency, low-amplitude distortions, originating from small variations and numerical errors in the underlying computer simulations.

Most methods for (2.1) make explicit use of the gradient of f. Thus these methods do not seem well suited for our problem because of i) but we still have the possibility to compute some numerical approximation to the gradient. This, however, 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.

Direct search (or derivative-free) methods proceed just by evaluating f at certain points. We will see that even so, they can be what is called *gradient-related* (Section 2.1.2). Apart from being well suited for our problem these procedures are attractive also because they are often easy to understand and to implement. Following the nice overview of direct search methods made by Powell [27], we may distinguish at least six different kinds of direct search methods: approximation methods, simplex methods, random methods, discrete grid or pattern search methods, line search methods and conjugate direction methods. Without going into any details 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 [26] or quadratic as in [5]. 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 method by Hext, Himsworth and Spendley [12] and the very popular algorithm by Nelder and Mead [24]. 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 [23] showed how the Nelder-Mead algorithm converges to a nonstationary point in a case when the objective is convex (and hence continuous).

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 [4]. 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 [27] 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 many well known algorithms, for example coordinate search and the algorithm by Hooke and Jeeves [15], are easy to understand and implement and have recently been given a general form by Torczon in [30], where she also presents a general convergence theory. This general formulation leaves a great freedom in the design of specific algorithms. For example, one may easily construct algorithms that make use of surrogate functions ([3],[4],[10]).

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

2.1 Positive basis pattern search algorithms

In the following we present a simplified version of the *positive basis pattern search algorithm* given in [18]. It is a class of algorithms developed for the unconstrained problem

$$\min_{x \in \mathbb{R}^n} f(x) \tag{2.2}$$

where f has the properties i), ii) and iii) listed in the beginning of the chapter. In [20] they are extended to linearly constrained problems. The positive basis pattern search algorithms are natural extensions of those in [30] 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* (Appendix). It leads to a possible decrease in the maximal number of function evaluations per iteration, which might be desirable because of the property ii).

2.1.1 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 goes as follows:

Algorithm 1. *(The general pattern search algorithm for unconstrained optimization.)*
Let $x_0 \in \mathbb{R}^n$ and Δ_0 be given. 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 define a specific pattern search algorithm one has to define a specific pattern P_k, an exploratory moves algorithm and how to update the pattern and the step length. We now give necessary conditions for how this should be done in order to guarantee convergence according to Theorem 1.

The pattern P_k is the matrix product of what the *basis matrix* B and the *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

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

and Γ is required to be a positive basis for $\mathbb{R}^n$. The 0 just means a column of zeros. We have $P_k = BC_k = [B\Gamma \; BL_k \; 0]$ and we call $B\Gamma$ *the core pattern.*

The conditions on the exploratory moves algorithm are that $s_k \in \Delta_k P_k$ and if there exists a core pattern step $y \in \Delta B\Gamma$ such that $f(x_k + y) < f(x_k)$ then $f(x_k + s_k) < f(x_k)$.

We either leave the step length unchanged or half it. In [18] 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".

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, $\Delta_{k+1} = \Delta_k$.

2.1.2 Some properties of the general algorithm

The general pattern search algorithm 1 is a *gradient-related adaptive grid method* which only require *simple decrease* in the objective function, i.e. $f(x_{k+1}) < f(x_k)$.

The simple decrease condition, which follows directly from the definition of the algorithm, 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 ([30]) that the iterates x_k lay on differently scaled versions of the lattice 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. In this sense pattern search methods are adaptive grid methods. Let us consider the following example in $\mathbb{R}^2$:

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

This pattern is illustrated in Figure 2.1. We see how $\Delta_k B$ generates a lattice on which to search for the next iterate. C determines the possible steps (indicated by arrows) to take from the current iterate.

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 consist of an arbitrarily large (but bounded) part of the lattice.

The fact that Γ is a positive basis guarantees the existence of a lower bound on the angle θ between the negative gradient and the best search direction. If we for an arbitrary fixed vector x define

$$\cos(\theta) = \max_{y \in \Gamma} \frac{x \bullet y}{\|x\|\|y\|},$$

we have the bound ([18])

$$\cos(\theta) \geq K \frac{1}{n\sqrt{n}}, \tag{2.3}$$

where the constant $K > 0$ depends on Γ. Hence the pattern search methods are gradient-related: there will always exist a search direction $d \in \Gamma$ that captures a certain part of the steepest descent direction.

We note how the bound (2.3) gets worse with n, indicating a loss of efficiency for the pattern search algorithms with increasing dimension. If

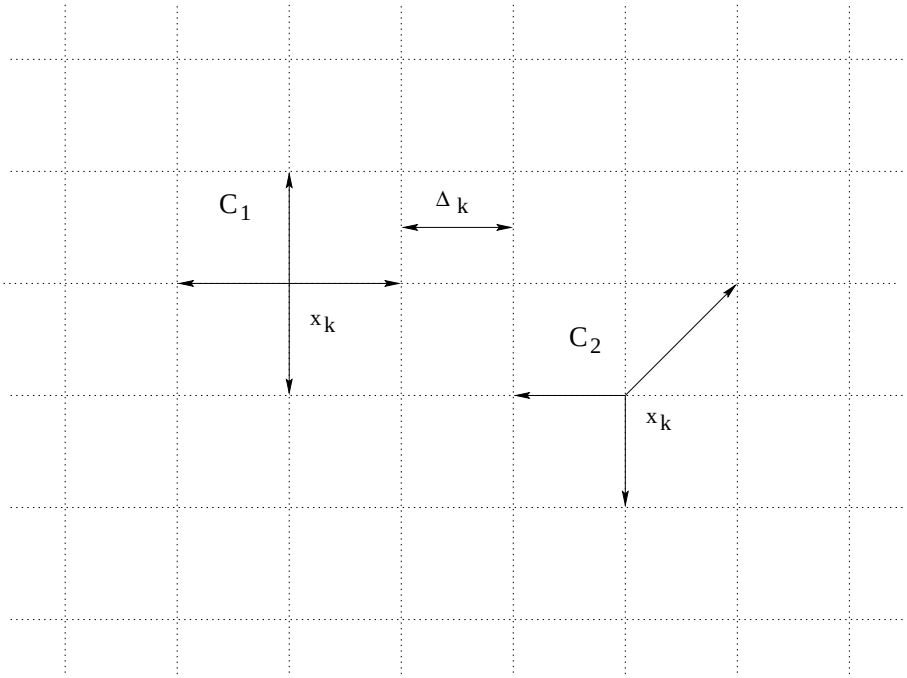


Figure 2.1: The dashed lines represent the lattice generated by $\Delta_k B$ when $B = I$. The generating matrix C determines the possible steps from the current iterate x_k.

$\Gamma = [I \ -I]$, we have from [30] the sharp bound $\cos(\theta) \geq 1/\sqrt{n}$. Whether the bound in (2.3) is sharp or not (regarding the power of n) I do not know. However, in Theorem 9 in Chapter 5 we show that when Γ is the minimal positive basis consisting of a regular simplex, the sharp bound is $1/n$.

2.1.3 Convergence

Let $\{x_k\}_{k=0}^{\infty}$ be the sequence of points generated by the generalized pattern search algorithm 1 applied to problem (2.2). From [18] we then have the following convergence result.

Theorem 1. *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 on the pattern search algorithm the lim inf in this theorem can be replaced by lim ([18]). 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

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

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

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

This means that if for example simple coordinate search, which in [30] is shown to be an instance of the generalized pattern search algorithm, starts off from the origin it will stay there in spite the existence of a descent direction.

2.1.4 Bound constraints

The shape optimization problem motivating this thesis is, by its nature, constrained (see Section 4.7). We therefore note the extension made in [19] of pattern search algorithms to bound constrained problems,

$$\min_{l \leq x \leq u} f(x), \tag{2.4}$$

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 (2.4), see [19]. 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 1 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 (2.4).

2.1.5 Examples: coordinate search and the SBA

We now describe the exploratory moves algorithms for coordinate search with fixed step length and a variant of the Hooke and Jeeves algorithm [15] called the Sherif-Boice algorithm [29], or SBA.

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 has 3^n columns that contains all possible combinations of $\{-1, 0, 1\}$. For $n = 2$ we have

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

Algorithm 3. *(The exploratory moves algorithm for coordinate search with fixed step length.)*
Let x, Δ and $f(x)$ be given. Let $s = 0$ and $min = f(x)$.
For $i = 1, \cdots, n$ do
    $s^i = s + \Delta e_i$
    If $f(x + s^i) < min$ then $min = f(x + s^i)$ and $s = s^i$,
    else $s^i = s - \Delta e_i$
        If $f(x + s^i) < min$ then $min = f(x + s^i)$ and $s = s^i$
Return $s, f(x + s)$.

We denote this algorithm by $(s, f(x + s)) = cs(x, f(x), \Delta)$. The reason for returning the function value at the new point and not only the step is to reflect that we do not want to compute the function at a point more than once since it is supposed to be very expensive to do so.

The SBA

The SBA [29] is a variant of the Hooke and Jeeves algorithm [14]. It alternately performs coordinate searches and *pattern steps*, which is an attempt to further investigate promising directions built up by a preceding coordinate search. Also, some of the currently best points are marked as *base points*. We say that a search or step *succeeds* if it leads to a lower function value and otherwise that it *fails*.

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

In contrast to the Hooke and Jeeves algorithm the SBA only performs the coordinate search after the pattern step *if the pattern step was succesful*. The formulation above differs from the original one in [29] because it reduces the step length also when a failed coordinate search follows after a successful

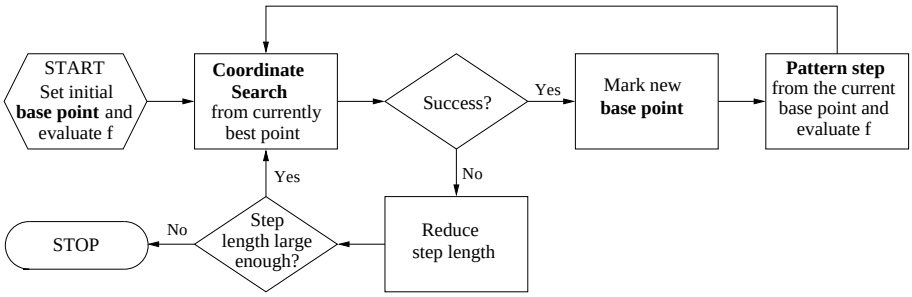


Figure 2.2: Flowchart of the SBA.

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.

The pattern matrix is the same as for the Hooke and Jeeves algorithm and can be found in [30]. As before, $(s, f(x+s)) = cs(x, f(x), \Delta)$ is a short hand for Algorithm 3. Furthermore, *bp* denotes "base point", *pbp* " previous base point". *do_ps*, "do pattern step", is either 1 or 0.

Algorithm 4. *(Exploratory moves for the SBA.)*
Let $x_k, f(x_k), \Delta_k, bp, pbp$ *and* do_ps *be given. Let* $s_k = 0$.
If $do\_ps = 1$ *then* $x_p = x_k + (bp - pbp)$
    If $f(x_p) < f(x_k)$ *then* $s_k = bp - pbp$.
    $do\_ps = 0$.
else $(s_k, f(x_k + s_k)) = cs(x_k, f(x_k), \Delta_k)$
    If $s_k \neq 0$ *then* $do\_ps = 1$, $pbp = bp$, $bp = x_k + s_k$.
Return $s_k, f(x_k + s), bp, pbp$ *and* do_ps.

2.2 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.

2.2.1 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 2 in Chapter 5). 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 5. *(Exploratory moves for a minimal positive basis search algorithm.)*
Let $x, f(x), \Delta$ *be given. Let* $s = 0, min = f(x), i = 0, j = 0$.
While $i \leq n+1$ *and* $j < n$
    $s^i = s + \Delta\Gamma(i)$
    If $f(x + s^i) < min$ *then* $min = f(x + s^i), s = s^i, j = j + 1$.
    $i = i + 1$
Return $s, f(x + s)$.

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.

2.2.2 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 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 6). From [30] we have $d(\Gamma^1) = 1/\sqrt{n}$ and Theorem 9 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

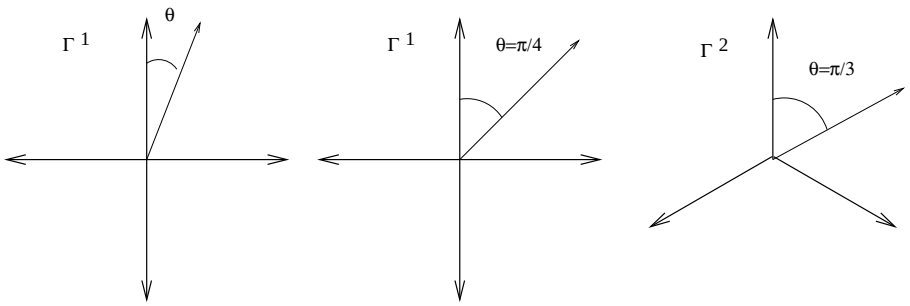


Figure 2.3: Illustration of how a positive basis Γ approximates an arbitrary vector. The basis to the left is maximal and the basis on the right is minimal. θ is the angle between some vector and the closest vector in the basis. In the middle and to the left is seen a vector that is furthest possible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

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.

2.3 Pattern search, oracles and surrogate functions

In this section we suppose that we have access to a *surrogate* (or *model*) *function* $\tilde{f}$ for f, i.e., $\tilde{f} \approx f$ and $\tilde{f}$ is less expensive to evaluate than f. We discuss, inspired by [3], [31], [33], how a surrogate can be used in a systematic way in order to optimize f. We first discuss possible choices of surrogate functions.

A well known method that uses a surrogate function for optimization is the *trust region method* which uses a quadratic model that is the second order Taylor expansion of f around the current iterate.

If we have no information about the derivatives, the surrogate could be an algebraic model constructed from a sampling of the objective([5], [26]). As the optimization proceeds, the model can be updated whenever f is evaluated at a new point. If the objective is like a "black box", i.e., given a point it returns a value and we have no knowledge at all of how this is done, one is forced to use algebraic models. If f is based on some experiment or complicated computer simulation, $\tilde{f}$ can be the result of performing a simplified experiment or simulation. In the case studied in this report, a point x determines six pde:s that have to be solved numerically in order to

determine $f(x)$. We can take $\tilde{f}$ to be the result of using a larger mesh size in the numerical pde solver, or approximate the equations with something that is easier to solve.

Sometimes the notation $\tilde{f}_k$ is used to indicate that the surrogate is updated during the optimization. Since we will not discuss any updating procedures we simply write $\tilde{f}$.

How should $\tilde{f}$ be used in the optimization of f? Most straightforward would be to optimize on $\tilde{f}$ until convergence and then use the result as a start for the optimization of f. However, if $\tilde{f}$ is not a reliable model of f the result could be that we start the optimization of f even further from the optimum and have to spend even more time to get close to the optimum than if we had not used a model at all. This is illustrated by the numerical results in Section 4.8.2. The reason for restricting the search on the surrogate may also be, as for trust region method, that we have a reason to believe that $\tilde{f} \approx f$ only holds in a small neighbourhood of the current iterate. It is thus clear that surrogate functions should be used in some kind of framework that takes into account the reliability of the model.

The main reason why the general pattern search algorithm 1 is well suited for the construction of such a framework is the freedom of choice of the exploratory moves algorithm. It may consist of only 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 2.1.1). Then we can write the following simple exploratory moves algorithm.

Algorithm 6. *(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 $s^1, \ldots, s^m$ *on the current grid.*
If $f(x_k + s^i) < f(x_k)$ *for some* $i \in \{1, \ldots, m\}$ *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* $s_k = s$.
Return $s_k, f(x_k + s_k)$.

The oracle could consist of some iterations with an arbitrary optimization procedure on the surrogate $\tilde{f}$.

We now give a coordinate search algorithm that uses a surrogate function. We write $(s, f(x+s)) = cs(x, f(x), f, \Delta)$ 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 7. *(Exploratory moves for coordinate search that uses a model for the objective function.)*
Let $x, f(x), \Delta$ *be given and let* $s = 0$.
$(\tilde{s}, \tilde{f}(x + \tilde{s})) = cs(x, \tilde{f}(x), \tilde{f}, \Delta)$
If $\tilde{s} \neq 0$ *then*
    If $f(x + \tilde{s}) < f(x)$ *then* $s = \tilde{s}$,
    else $(s, f(x + s)) = cs(x, f(x), f, \Delta)$
else $(s, f(x + s)) = cs(x, f(x), f, \Delta)$.
Return $s, f(x + s)$.

We get versions of the Hooke and Jeeves algorithm and SBA that can use a surrogate for the objective function simply by changing from the ordinary coordinate search in these algorithms to the one just described. 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

The problem described in the Introduction has two main theoretical aspects, namely, optimization and fluid dynamics. We discussed the first part in the previous chapter and now continue with the second. Everything in this chapter can be found in most introductory textbooks on fluid dynamics.

We consider the motion in $\mathbb{R}^3$ of a viscous, incompressible fluid of constant density. Such a motion may be described by the physical quantities

$\mathbf{u}(\mathbf{x}, t) =$ the velocity of a fluid particle at the position $\mathbf{x}$ in space at time t

$p(\mathbf{x}, t) =$ the static, or internal, pressure at the position $\mathbf{x}$ at time t

$\rho =$ the (constant) density of the fluid

$\mu =$ coefficient of viscosity

$\nu = \dfrac{\mu}{\rho}$ the kinematic viscosity,

The parameters ρ and μ are known quantities, depending only on the fluid under consideration. $\mathbf{u}$ and p are typically unknown functions of both $\mathbf{x}$ and t. They satisfy four partial differential equations, the so called Navier-Stokes equations.

3.1 The 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.

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 $V \subseteq \mathbf{R}^3$ 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. The surface stresses of the volume element are pressure and internal friction and may be represented by the *stress tensor*[1] $\sigma(\mathbf{x}, t) \in \mathbb{R}^3 \times \mathbb{R}^3$ 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 [7] 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

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

[1] 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 [16].

where μ is the *coefficient of viscosity* and

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

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.

3.2 The 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 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.

3.2.1 Flows with high and low Reynolds numbers

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 one has to reduce the number of boundary conditions.

3.3 The 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.)

3.4 Turbulent flow and the Reynolds equations

We briefly present the fundamental problems of turbulent flow modelling, and state and motivate the Reynolds equations.

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 ([22]). 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 ([22]). 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"([17]). Hence, even though the Navier-Stokes equations are the general equations of fluid dynamics, they are not appropriate to be used in a computer model for turbulent flows since such a model would require an impractical number of computational nodes in order to resolve the small-scale (stochastic) effects on the large-scale (average) flow behaviour.

The stochastic behaviour of turbulent flows require a statistical, averaged description. To this end, we divide each flow quantity q into

$$q = Q + q',$$

where Q is a time-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 ([32], p. 300).

If we put $u_i = U_i + u_i', p = P + p'$ into the Navier-Stokes equations and then take the average, we get the so called *Reynolds equations* for the mean

quantities U, P,

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

The six quantities $\tau_{ij} = -\rho\overline{u_i' u_j'}$ behave as extra 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 Turbulence modelling

We describe one way of closing the Reynolds equations, namely the *Standard* $k-\epsilon$ *model* ([17], [11]). It consists of two transport equations for the turbulent kinetic energy k and its dissipation rate ϵ. The model 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 independent of the direction: "perfect disorder reigns" ([13], p.3). Hinze [13] 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. According to Fluent [11] the popularity of the $k-\epsilon$ model for industrial flow calculations is due to its "robustness, economy, and reasonable accuracy for a wide range of turbulent flows". They also state that for certain flows a Reynolds stress model (RSM) is a must in order to model the nonisotropic effects on the flow, but, for the time being, the RSM is computationally more expensive and less robust then the $k-\epsilon$ model. Malalasekera and Versteeg [21] remark that the $k-\epsilon$ model is the most widely used and validated turbulence model.

In the Reynolds equations (3.11), the Reynolds stresses appear together with the viscous stresses. Therefore, it is natural 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 lead to the *modified* Boussinesq hypothesis [13]:

$$-\rho\overline{u_i' u_j'} = 2\mu_t \bar{D}_{ij} - \frac{2}{3}\rho k \delta_{ij}. \tag{3.12}$$

Here δ_{ij} is the Kronecker delta function, $\overline{D}_{ij}$ the average rate of deformation and μ_t a new parameter called the *turbulent viscosity*. k, called the turbulent kinetic energy, is defined by

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

We also define the rate of dissipation of turbulent kinetic energy,

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

The way in which kinetic energy is dissipated may briefly be explained as follows ([32]). Turbulent regions of the flow are made up of eddies at different scales. Energy passes from the mean flow to the largest eddies. These in turn pass the energy down to smaller and smaller eddies until the length scale is such that viscosity becomes important. (This phenomenon is known as the *energy cascade*.) Energy is then dissipated through the work of the smallest eddies against viscous stresses.

Arguments based on dimensional analysis under the assumption that one length scale and one velocity scale suffice to describe the effects of turbulence yields that μ_t can be related to k and ϵ as ([21], [28])

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon}. \tag{3.13}$$

where $C_\mu = 0.09$.

By an averaging procedure similar to the one used to derive the Reynolds equations from the Navier-Stokes equations, it is possible to derive two transport equations for k and ϵ. Modelling some of the terms in these equations (see for example [21]) gives, for high Reynolds numbers, the following two equations used in the Standard $k-\epsilon$ model ([17]):

$$\begin{cases} \rho\dfrac{Dk}{Dt} = \dfrac{\partial}{\partial x_i}(\mu_t \dfrac{\partial k}{\partial x_i}) + 2\mu_t \bar{D}_{ij}\bar{D}_{ij} - \rho\epsilon \\ \rho\dfrac{D\epsilon}{Dt} = \dfrac{\partial}{\partial x_i}(\dfrac{\mu_t}{\sigma_\epsilon}\dfrac{\partial \epsilon}{\partial x_i}) + C_{1\epsilon}\dfrac{\epsilon}{k}2\mu_t \bar{D}_{ij}\bar{D}_{ij} - C_{2\epsilon}\rho\dfrac{\epsilon^2}{k}, \end{cases} \tag{3.14}$$

where

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

Once Equation (3.14) has been solved, k and ϵ determine the Reynolds stresses through the modified Boussinesq hypothesis (3.12) and Equation (3.13).

To sum up, the Standard $k-\epsilon$ model models the six Reynolds stresses 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 (3.14), together with the seven algebraic relations (3.12) and (3.13).

3.5.1 Near walls

The presence of solid boundaries, "walls", significantly affects turbulent flows. The no-slip condition enforces large velocity gradients close to the wall and these give rise to the production of turbulent energy. Gregory Seil [28] states that solid boundaries "act as sources of turbulence and energy loss". Fluent [11] notes that the near-wall modelling is important in order to get reliable numerical solutions. Since the Standard $k-\epsilon$ model described above was developed for high Reynolds number flows, it has to be modified in order to account for the low Reynolds effects always present in the near-wall region. In general, there are at least two methods for numerical modelling of the turbulent boundary layer ([17]): the *wall-function-method* and the *low-Reynolds-number-modelling method*. In the latter, the turbulence model equations themselves take viscous effects into account and 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 [11].

In the following we discuss the structure of a turbulent boundary layer and present a set of wall functions that can be used in the Standard $k-\epsilon$ model. We denote by $U(>0)$ and u' the mean and fluctuating flow velocities tangential to the wall, by v' the fluctuating velocity normal to the wall and by y the distance from the wall. Often, the dimensionless quantities

$$U^+ = \frac{U}{u_\tau} \qquad \text{and} \qquad y^+ = \frac{u_\tau y}{\nu}$$

are used instead of U and y. Here, $u_\tau = (\tau_w/\rho)^{1/2}$ is called the *friction velocity* and τ_w is the *wall shear stress*.

According to Hinze ([13], 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 ($0 \leq y/\delta \lesssim 0.1$) the shear stress is approximately constant and equal to the wall shear stress τ_w ([21]). 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 an outer layer ($0.01 \lesssim y/\delta \lesssim 0.1$), sometimes called the *log-law layer*, where turbulent stresses dominate.

In the wall function approach to the modelling of turbulent boundary layers, the values of U, k and ϵ at the wall-adjacent nodes are required to fulfil three relations. First, we have the *log-law*,

$$\frac{U}{u_\tau} = \frac{1}{\kappa} \ln \left(\frac{E\, u_\tau\, y}{\nu} \right), \tag{3.15}$$

that can be derived for the turbulent part of the inner region. That is why this region is also called the log-law layer. The derivation is accomplished in [13] using the Boussinesq hypothesis, the assumption of a constant shear stress and an assumption concerning the length scale of the near-wall turbulence. From measurements the numerical values of the constants are found to be $\kappa = 0.41$ (von Karman's constant) and $E = 9.8$ for a smooth wall ([21]). Experimental measurements also show the log-law to be valid for $30 < y^+ < 500 \sim 1000$ ([13]).

In the near-wall region the production term in the transport equation for k becomes ([28])

$$G_k = -\rho\, \overline{u'v'}\, \frac{\partial U}{\partial y}. \tag{3.16}$$

In the inner region, the shear stress is approximately constant and equal to the wall shear stress. 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

$$\frac{\partial U}{\partial y} = \frac{u_\tau}{\kappa y}. \tag{3.17}$$

The last two equations inserted into Equation (3.16) result in

$$G_k = \rho \frac{u_\tau^3}{\kappa y}. \tag{3.18}$$

We now assume that the production G_k of turbulent kinetic energy equals the rate of dissipation $\rho\epsilon$ (see [13], p.649). Hence,

$$\epsilon = \frac{u_\tau^3}{\kappa y}. \tag{3.19}$$

Inserting this expression for ϵ into Equation (3.13) and solving for k gives

$$k^2 = \frac{\mu_t u_\tau^3}{C_\mu \rho \kappa y}. \tag{3.20}$$

The Boussinesq hypothesis,

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

together with Equation (3.17) allow us to write Equation (3.16) as

$$G_k = \mu_t \frac{u_\tau^2}{\kappa^2 y^2}. \tag{3.21}$$

Equating the two expressions (3.18) and (3.21) for G_k and solving for μ_t results in

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

Inserting this expression for μ_t into Equation (3.20) we finally arrive at

$$k = \frac{u_\tau^2}{C_\mu^{1/2}}. \tag{3.22}$$

When using the Standard $k-\epsilon$ model with wall functions, the wall-adjacent nodes should be well inside the log-law layer, but not closer; $30 < y^+ < 200$, say. The values of U, k and ϵ in these nodes are determined using the relations derived above, i.e.,

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

It should be noted that the friction velocity u_τ is not known a priori, but rather calculated from the gradient of the mean velocity near the wall. Hence, it is not obvious how to employ these formulae in practice; the interested reader is referred to [21].

3.6 Cavitation

In the Introduction we discussed the consequences of *cavitation* in a waterjet duct. We saw that it must be avoided since it leads to a less efficient propulsion of the ship. We now define the *cavitation number* and discuss what is meant by cavitation.

In a domain with flowing water the internal pressure p of the fluid will vary with the position. We say that the water *cavitates* if the internal pressure at some position is less than the vapour pressure p_v so that the water starts to boil. The local risk of cavitation may be measured by the so called *cavitation number* σ which, in the case of a stationary flow, is defined by

$$\sigma(\mathbf{x}) = \frac{p(\mathbf{x}) - p_v}{p_d}, \tag{3.23}$$

where $p_d = 1/2\rho|\mathbf{u}|^2$ is the dynamic pressure. The larger σ is, the less is the risk of cavitation. We note that σ, being a function of the position, is a local measure of the risk of cavitation and to get a global measure in $\mathbf{x}$ we define the real number

$$\sigma_{min} = \min_{\mathbf{x} \in D \subseteq \mathbb{R}^3} \sigma(\mathbf{x}). \tag{3.24}$$

This quantity, *the least cavitation number*, will be our objective function in the shape optimization of the waterjet inlet duct model in the next chapter. It may seem to be quite a pessimistic measure of the cavitation properties. Also, taking the minimum implies that we cannot *a priori* assume higher regularity than C^0 for f. However, it is the measure proposed by Kamewa. We finally note that with fixed boundary conditions, σ_{min} depends only on the geometry of the pipe flow domain D.

Chapter 4

Shape optimization of a waterjet inlet duct model

We present a solution strategy, based on computer simulations, for the problem of the optimal shape design of a waterjet inlet duct model with respect to cavitation. For a more thorough background discussion and motivation to this problem, we refer the reader to the Introduction.

The structure of the chapter follows the structure of the problem. We first sketch a model of the problem in order to discuss some basic properties of it. We then make a precise problem formulation. Then follows a presentation of how it has been solved. We describe our model of the waterjet inlet duct geometry, the computational grid generation and the flow model that has been used. We discuss the properties and the implementation of the objective function, followed by a description of the optimization procedure and the numerical results. We conclude with some remarks and suggestions for future research.

4.1 Preliminairies

We shortly describe how we intend to model the waterjet inlet duct. We also discuss what to do about the fact that our optimization problem is infinite dimensional.

4.1.1 Water flow in s-shaped pipes

In general the geometry of a flow problem may be described by an open subset $D \subseteq \mathbb{R}^3$ with its boundary ∂D partitioned into the wall W, the inlet I and the outlet O. Let $\mathbf{n}$ be the outward unit normal to ∂D, where it is

defined.

Loosely speaking, we model the waterjet inlet duct with an s-shaped pipe having two bends as in Figure 4.1. The inlet and outlet openings of the pipe are plane surfaces parallel to each other and the pipe bends smoothly in between.

The essential purpose of such an *s-pipe* is to accomplish a parallel transport of the water flowing through it. This means that the water velocity at the outlet should be approximately parallel to that at the inlet, but the outlet should be "higher" than the inlet.

Water is viscous and incompressible, and the flow through the pipe is stationary but turbulent. Appropriate boundary conditions are a no-slip condition on W, a velocity condition at I and a pressure condition at O.

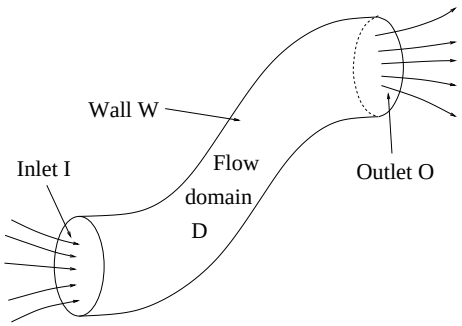


Figure 4.1: The pipe flow domain D with boundary partition into walls, inlet and outlet.

4.1.2 From infinite to finite dimensions by parameterization

Our problem is to find the shape of the s-duct that minimizes the risk of cavitation in the water flowing through it. The risk of cavitation is measured by the least cavitation number (Section 3.6) and hence the objective function is

$$f(D) = -\sigma_{min}(D), \tag{4.1}$$

where D is the pipe flow domain. Among what set, or class, of pipes D should we look for an optimal solution? The pipe flow domain is determined by its boundary surface which, since it is two-dimensional, may be thought of as a function from a subset of $\mathbf{R}^2$ into $\mathbf{R}^3$. But such functions have an infinite

number of degrees of freedom and we hence have an *infinite dimensional* optimization problem

$$\min_{D\in\mathbb{D}} -\sigma_{min}(D), \tag{4.2}$$

where $\mathbb{D}$ is some set of admissible domains. There are infinite optimization problems that are (analytically) solvable but this one is not. In order to solve it numerically we have to reduce the number of degrees of freedom to finitely many. This can be done by a *parameterization*. The details are left until later on, but it means that we describe the pipe with some variables, or parameters, each one capturing some aspect of the geometry considered to be important for the flow and the objective function. These parameters are described by a vector $\xi = (\xi_1, ..., \xi_n)$ in some subset Ω of $\mathbb{R}^n$. Each value of ξ uniquely determines a pipe flow domain $D(\xi)$. In this way we get the finite dimensional optimization problem

$$\min_{\xi\in\Omega} -\sigma_{min}(D(\xi)). \tag{4.3}$$

(Since the flow domain is completely determined by ξ we may simply write $\sigma(\xi)$.) The price we have to pay is a drastic reduction of the set where we look for the optimal pipe. We can not be sure that there is no other class where there are (much) better pipe. Thus the parameterization has to be done very carefully in order to preserve the main features of the problem. The challenge is to get a quite general class of pipes described by only a small, finite, number of parameters.

4.2 Problem formulation with some comments

Advised by Kamewa, we propose the following simplified model of the waterjet inlet duct (Figure 4.2). The inlet duct is described by a pipe having a circular cross section, i.e., there is a smooth curve through the pipe D so that if we take the intersection of D and a plane perpendicular to that curve we get a circular disc. The total length of the pipe is fixed to $8\,m$ and its height to $3\,m$. The radius of the inlet opening is set to $0.2\,m$ and the radius of the outlet opening to $0.2\sqrt{2}\,m$. For the waterjet propulsion of the ship to be in the intended direction we would like to have the outlet water velocity parallel to the inlet water velocity. We approximate this by requiring the last $0.6\,m$ of the pipe to be straight (cylindric).

First we want to find a finite dimensional parametric pipe model (explained in the previous section), which generates an appropriate class of

pipes with the properties just mentioned. Then this class of pipes can be described by all vectors ξ in $\Omega \in \mathbb{R}^n$, and our optimization problem becomes

$$\min_{\xi \in \Omega} -\sigma_{min}(\xi). \tag{4.4}$$

σ_{min} (Equation (3.24)) depends on the solution to a viscous, incompressible, stationary and turbulent flow problem on the domain D determined by ξ. Kamewa suggests the water flow through the pipe to be $1\,m^3/s$. Consequently, since the radius of the inlet opening is $0.2\,m$, we put the inlet velocity at I to $8\,m/s$ normal to I.

Because of the objective function, further discussed in Section 4.6, the problem (4.4) is to be considered a very hard one. After having given up all hope of finding anything like an analytic solution, we stick to computer simulations and numerical algorithms. Since the problem is complex and contains a lot to model in proportion to the time available for a post graduate student, we first strive to reach Aim 2 in the Introduction, i.e., to implement an iterative process that numerically searches for approximate solutions to the problem.

4.3 Geometric modelling

This section contains the construction of a parametric pipe model according to the simplifications of the waterjet inlet duct geometry done in the previous section. We begin with a description of the pipe by two curves. We then parameterize these two curves to obtain a parameterization of the pipe. Finally, we present the generic generation of the grid for the computational analysis of the flow.

4.3.1 The generic pipe geometry

We represent the pipe geometry by two curves (or functions).

The *centre curve* is a plane (the xy-plane) curve from the pipe inlet opening to the outlet opening. It is piecewise defined by five segments according to Figure 4.5. The first and last segments are straight lines parallel with the x-axis and the last one is at least $0.6\,m$ long. Between them are two circular arcs, connected by a straight line. The centre curve can be described by a smooth mapping $\gamma_c : [0,1] \to \mathbb{R}^3$ with $\gamma_c'(s) \neq 0$. Note that γ_c is a vector valued function. We fix the coordinate system by letting $\gamma_c(0) = (0,0,0)$,

$\gamma_c(1) = (8, 3, 0)$. We also require that

$$\frac{\gamma_c'(0)}{\gamma_c'(0)} = (1, 0, 0) = \frac{\gamma_c'(1)}{\gamma_c'(1)}.$$

The *radius curve* contains the information about the radius of each circular cross section along the pipe. It can be seen as a smooth real-valued function $\gamma_r : [0, 1] \to \mathbb{R}$ such that $\gamma_r > 0$, $\gamma_r(0) = 0.2$ and $\gamma_r(1) = 0.2\sqrt{2}$.

If we denote by $\mathbf{t}_c(s)$ the unit tangent to γ_c and by $\mathbf{n}_c(s)$ a unit vector in the xy-plane which is perpendicular to $\mathbf{t}_c(s)$, the pipe domain D is defined by

$$D = \{\gamma_c(s) + r \cdot \mathbf{n}_c(s) : s \in (0, 1), |r| < \gamma_r(s)\} \tag{4.5}$$

To prepare for the meshing of the pipe we partition D into one inner and four outer volumes, giving a cross section like the one in Figure 4.3, and divide the pipe into five segments along its length as shown in Figure 4.4. However, we leave the details of this and continue with the parameterization.

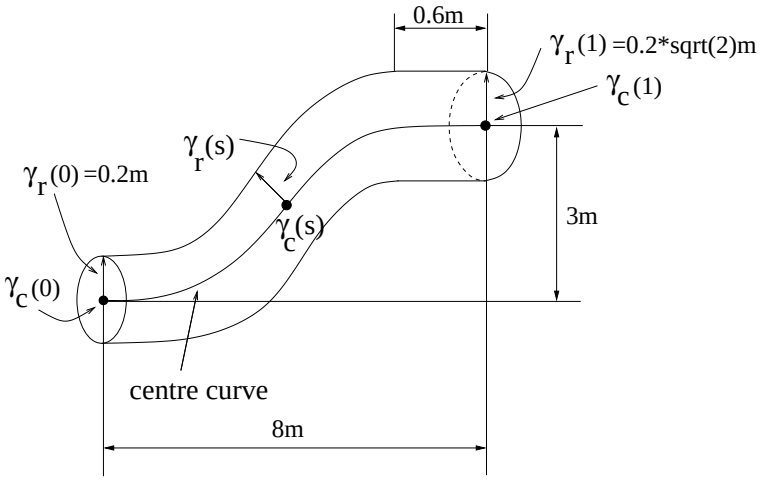


Figure 4.2: The centre and radius curves building up the pipe.

4.3.2 Parameterization of the geometry

With the generic pipe model just described at hand we parameterize the pipe geometry by parameterizing the centre and radius curves.

Under the condition that the centre curve should be continuously differentiable (a continuously changing tangent) it is uniquely determined by the

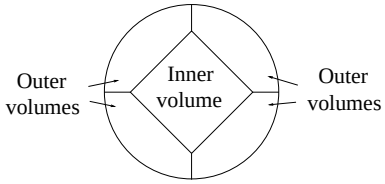


Figure 4.3: Partition into outer and inner volumes to prepare for a good mesh.

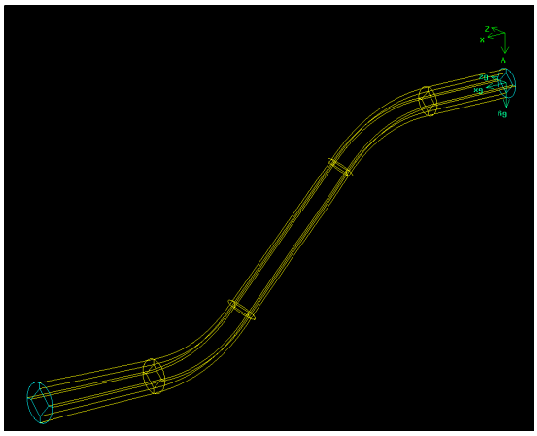


Figure 4.4: Partition of the pipe into five segments along its length to prepare for a good mesh. The cross sectional partitioning into inner and outer volumes can also be seen.

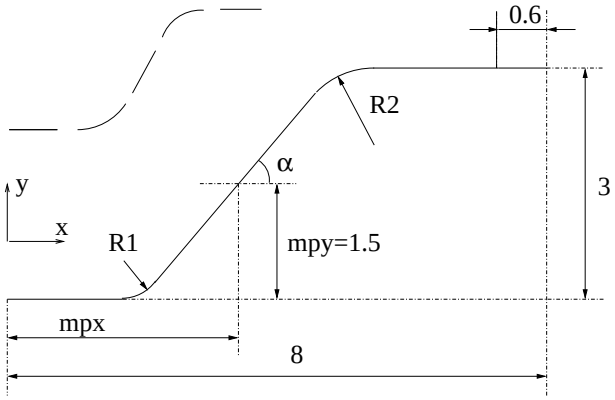


Figure 4.5: The parameterization of the centre curve. The upper part of the figure shows how the centre curve is built up by straight lines and circular arcs.

four parameters $mp_x, R1, R2$ and α described in the following. $R1$ and $R2$ are the radii in the first and the second arc, respectively. The curve has to pass through the point $(mp_x, 1.5, 0)$ and α is the angle between this segment and the x-axis (Figure 4.5). We thus have a parameterization of the centre curve γ_c. (In the Appendix it is shown how it can be represented by a NURB curve, which facilitates its computer implementation.)

We now define the radius curve γ_r as the first coordinate in a B-spline curve:

$$\mathbf{C}(s) = (\gamma_r(s), c_2(s), c_3(s)) = \sum_{i=0}^{3} N_{i,2}\mathbf{P}_i.$$

This is a NURB curve with all $w_i = 1$ (Appendix). We take $\mathbf{P}_0 = (0.2, 0, 0)$, $\mathbf{P}_0 = (0.2, 0, 0)$, $\mathbf{P}_0 = (r, 0, 0)$, $\mathbf{P}_0 = (0.2 \cdot \sqrt{2}, 0, 0)$ and the knot vector $U = \{0, 0, 0, u, 1, 1, 1\}$. It follows from the properties of NURB curves listed in the Appendix that γ_r is a continuously differentiable curve from 0.2 to $0.2\sqrt{2}$ with $\gamma_r'(0) = 0$. r and u are the two parameters. The first changes the thickness of the pipe while the second determines where along the pipe the first should have effect. This is best explained by Figure 4.6.

4.3.3 Constraints on the chosen parameters

In the following we give sufficient constraints on the chosen parameters in order for them to define the centre and radius curves which in turn determines the pipe.

We restrict our attention to centre curves where

$$0 < \alpha < \frac{\pi}{2}. \tag{4.6}$$

Since the centre curve passes through $(mp_x, 1.5, 0)$, we get

$$0 < mp_x < 7.4. \tag{4.7}$$

For the two arcs to be bended appropriately (like an "s" and not like an "inverted s")we get another two conditions on α from the first and the second bend respectively:

$$\tan(\alpha) > \frac{1.5}{mp_x}, \tag{4.8}$$

$$\tan(\alpha) > \frac{1.5}{7.4 - mp_x}. \tag{4.9}$$

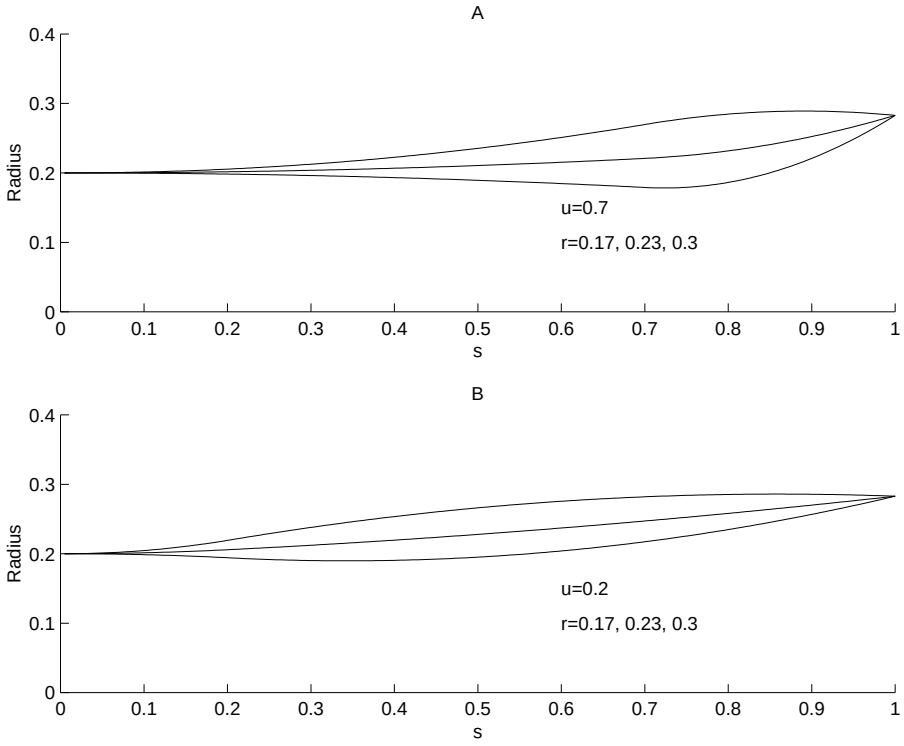


Figure 4.6: The effect of the radius curve parameters r and u. Each plot shows the radius curve with tree different values for r. In plot A, $u = 0.7$ and in plot B $u = 0.2$. We see how u controls whether the effect of r should come in the beginning or the end of the pipe.

To be able to connect the straight segments with arcs so that the curve is continuously differentiable, the radii can not be too large:

$$R1 < \left(mp_x - \frac{1.5}{\tan(\alpha)} \right) \frac{1}{\tan(\alpha/2)}, \tag{4.10}$$

$$R1 < \frac{1.5}{\sin(\alpha)} \frac{1}{\tan(\alpha/2)}, \tag{4.11}$$

$$R2 < \left(7.4 - mp_x - \frac{1.5}{\tan(\alpha)} \right) \frac{1}{\tan(\alpha/2)}, \tag{4.12}$$

$$R2 < \frac{1.5}{\sin(\alpha)} \frac{1}{\tan(\alpha/2)}. \tag{4.13}$$

(These last conditions follow by considering the following problem: given two line segments AB, BC, find the circle with the largest radius with tangential points on the segments, see Figure 4.7.)

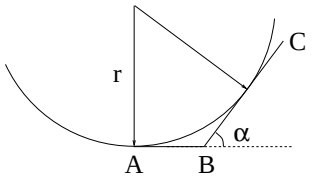


Figure 4.7: The circle with the largest radius r having tangential points on the line segments AB and BC.

The two natural conditions on the radius curve parameters are

$$0 < u < 1, \tag{4.14}$$

$$r > 0. \tag{4.15}$$

$r > 0$ implies that the radius curve is positive everywhere, which follows from the properties of NURBs listed in the Appendix.

There may be one more natural condition which involves both the centre and radius curve parameters. If the radius of the pipe is too large compared to the radius of the centre curve, the pipe wall will intersect itself. This is no problem in the abstract definition in Equation (4.5) but poses a problem when it comes to the implementation.

4.4 Grid generation

The CFD (Computational Fluid Dynamics) analysis of a flow is performed on a computational grid, a *mesh*. We describe the generation of such a grid for the generic pipe model given above.

To generate high quality meshes, or grids, for numerical flow computations is, in general, hard. Different geometries require different meshes and the size of the mesh may have to change a lot between different parts of the geometry. Large meshes (having many elements) lead to time-consuming computations and storage problems.

The challenge with meshing in the context of shape optimization is to find out how to mesh automatically and still be (quite) sure that the quality of the mesh is high enough for all geometries in the class over which we are optimizing. We note that we only need a grid fine enough for giving reliable *cavitation values* and not for resolving *all* aspects of the flow such as secondary flows etc.

4.4.1 Computer representation of geometry and mesh with Gambit

The geometry is constructed and meshed with a commercial computer software called Gambit which is a preprocessor to the computational fluid dynamics program Fluent5. In Gambit the geometry is built in a hierarchical way starting with points, or vertices, connecting them to edges, making surfaces from the edges and volumes from the surfaces. Then the geometry is meshed, starting, in the case of a structured mesh, with the edges and continuing with surfaces and volumes. Finally, boundaries are defined and the mesh is exported to a file which can be imported into Fluent5.

Gambit admits automatic running by a so called *journal file*, which is printed outside Gambit. Since we have a generic pipe model, which is well suited for constructing the pipe in the way Gambit does, the structure of the journal files for different pipes are the same and so these files can be generated automatically by a computer program. I have had access to such a code, written in C++ by Sara Ågren ([34]).

4.4.2 Generic meshing procedure

We have chosen a structured mesh, and this for two reasons. Firstly, experience tells us that structured grids render, compared to unstructured grids, higher stability and faster convergence. Secondly, with structured meshes it is easier to map data between pipes which may be helpful for speeding up the CFD computations. The main drawback is that structured grids are more difficult to generate and less flexible than structured grids.

The pipe domain D has been partitioned into 25 parts according to Figure 4.4. Hence it consists of five sections of the sort shown in Figure 4.8. Three natural numbers (n_1, n_2, n_3) are given as input to the meshing procedure. n_1 is the number of grid nodes along the pipe. It determines the number of mesh cross sections along the duct. The number of mesh cross sections of each section (i.e., the number of (uniformly distributed) grid nodes on edge A in Figure 4.8) is determined by the length of the section compared with the length of the entire duct. n_2 grid nodes are uniformly distributed on edge C, which corresponds to one fourth of the circular end of each section. Finally, n_3 is the number of grid nodes at edge B in each section. The grid points on B are stretched to get a good near-wall mesh. The stretching is described by the successive ratio SR which is the ratio between two adjacent mesh intervals. $SR = 1$ gives a uniformly distributed edge mesh.

This meshing procedure reveals the purpose of splitting the pipe into 25 parts. The outer volumes (Figure 4.3) together with the stretching at

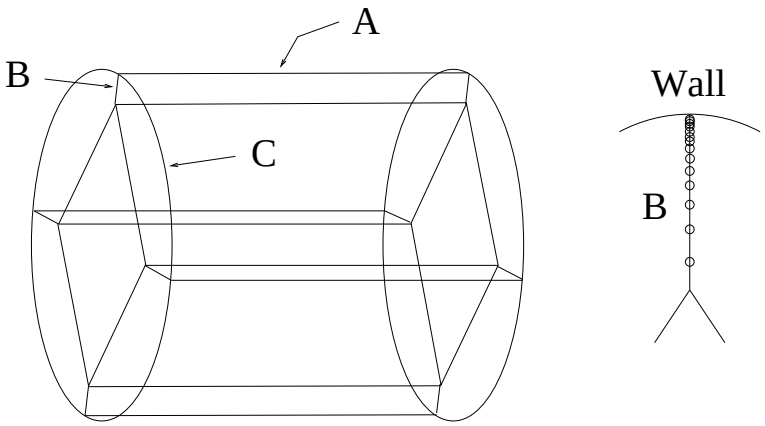


Figure 4.8: One of the five sections constituting the pipe. To the right it is shown how the mesh points are stretched at B so that they are closer together near the wall.

edge B gives the possibility of controlling the quality of the near-wall mesh without getting an excessively large overall mesh. The splitting along the pipe is made to get a good quality on the mesh elements themselves, i.e. to make them "orthogonal" enough. As already mentioned, the meshing is done starting with the edges but a uniform mesh on an edge along the whole pipe would result in skewed elements reducing the mesh quality, see Figure 4.9. Admittedly, this problem is quite related to the specific software used.

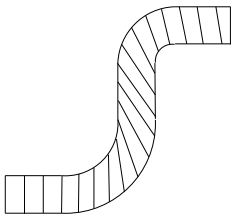


Figure 4.9: The mesh elements become skewed if the mesh is constructed from a uniformly distributed mesh on one edge along the whole duct.

To sum up, the generic meshing procedure just described gives a high quality mesh of moderate size, with stable and quite fast convergence in the fluid dynamic computations as a consequence, and this holds for a large set of different pipe geometries.

4.5 The flow model

We present the pipe flow model that has been used.

The flow in a pipe D is viscous, incompressible, stationary and turbulent. Thus it is best described by the Reynolds equations (3.11). The viscosity of the fluid is set to $\mu = 0.001003\,kg/(ms)$ and its density to $\rho = 998.2\,kg/m^3$, corresponding to water. The kinematic viscosity then becomes $\nu = 1.002 \cdot 10^{-6}\,m^2/s$. The inlet velocity is set to $8\,m/s$ in the x-direction. This gives the flow of $1\,m^3/s$ recommended by Kamewa. The outlet pressure is set to $p_{atm} = 1\,atm$ since the water leaves the waterjet inlet duct above the sea surface. At the pipe wall we have the no-slip condition.

If we choose the diameter of the inlet opening as the characteristic length scale and the inlet velocity as the characteristic velocity scale, the Reynolds number for our flow problem becomes

$$Re = \frac{8(m/s) \cdot 0.4(m)}{1.002 \cdot 10^{-6}(m^2/s)} \approx 3 \cdot 10^6.$$

The CFD analysis has been done with the commercial software Fluent5.0. Fluent uses the finite volume method for numerical solutions of both laminar and turbulent flows and the user can choose between different numerical schemes, turbulence models and wall functions. For a discussion of the effects of different turbulence models and wall functions on the numerical solutions of a pipe flow, we refer to [28]. We have chosen to start with a first order upwind scheme, followed by a second order upwind scheme and lower factors of under-relaxation. This gives a fairly stable and accurate solution. The pressure-velocity coupling is done with the SIMPLE algorithm. For explanations of the technical terms mentioned in this paragraph we refer to [21].

We have used the Standard $k - \epsilon$ model with standard wall functions (Section 3.5). For this model to be valid, the y^+-values at the wall-adjacent nodes should be between 30 and 200 (Section 3.5.1). Small y^+-values are obtained by a fine near-wall mesh, but how fine can not be computed a priori. The best way to solve this problem is to use, perhaps repeatedly, Fluent's built-in function for adaption of the near-wall mesh with respect to y^+.

The numerical solver is iterative and quite time-consuming. To get a converged solution for a pipe with an acceptable mesh size takes around 400 iterations or 30 to 60 minutes on a SUN Ultra workstation. The time depends a lot on the mesh size (Table 4.6). Consequently it is important not to choose a too fine (large) mesh.

4.6 The objective function

In this section we discuss some theoretical aspects of the objective function and also describe how it has been implemented.

The objective function, defined in Equation (4.1), depends on a turbulent flow problem. In practice, therefore, it is time-consuming to evaluate, does not have explicit derivatives and the complexity of its implementation (described below) introduces high frequency distortions of the idealized function. Hence it is an objective function of the kind described in Chapter 2.

Theoretically, not much can be said about the structure of the objective function. We know nothing *a priori* about the number of local minima or the behaviour of the function close to such a minimum. This is further discussed in connection to the numerical results at the end of this chapter. Also, since the function is given by taking the minimum over a region in space, it can be expected to be continuous (C^0), but not smoother. The reason is that we do not know that the location in the pipe where the minimum cavitation number occurs moves in a continuous way. It may jump, for example between the first and the second bend. As long as it stays in approximately one place, however, we can expect $-\sigma_{min}$ to be at least continuously differentiable (C^1). The question of regularity is essential. The convergence analysis for pattern search methods requires that the objective function is C^1 (Theorem 1 in Section 2.1.3) and this condition is necessary (as shown by the example in Section 2.1.3).

The implementation of the objective function has been written in Matlab but involves calls to functions written in UNIX and C++ as well as to the two commercial programs Fluent5.0 and Gambit1.2. A flow chart is found in Figure 4.10.

The function call is done with values for the parameters $mp_x, \alpha, R1, R2, r, u$ previously described. If they do not fulfil the constraints, infinity is returned as the objective value. Otherwise, a grid is generated for the pipe. First two files defining the centre and radius NURB curves are written. These files are read by a C++ program which writes a journal file with all the commands for constructing the geometry and its mesh in Gambit. Next, Gambit is run to give a mesh file. The CFD analysis in Fluent is performed on that mesh and the cavitation numbers are stored in a file. Finally, the least cavitation number is extracted from that file and its negative is returned as the function value.

As mentioned already, the function is time-consuming. Most of the time is spent on the CFD analysis but this depends heavily upon the mesh used. The geometric modelling and grid generation take approximately five minutes.

The implementation is fairly stable but it may happen that one step fails,

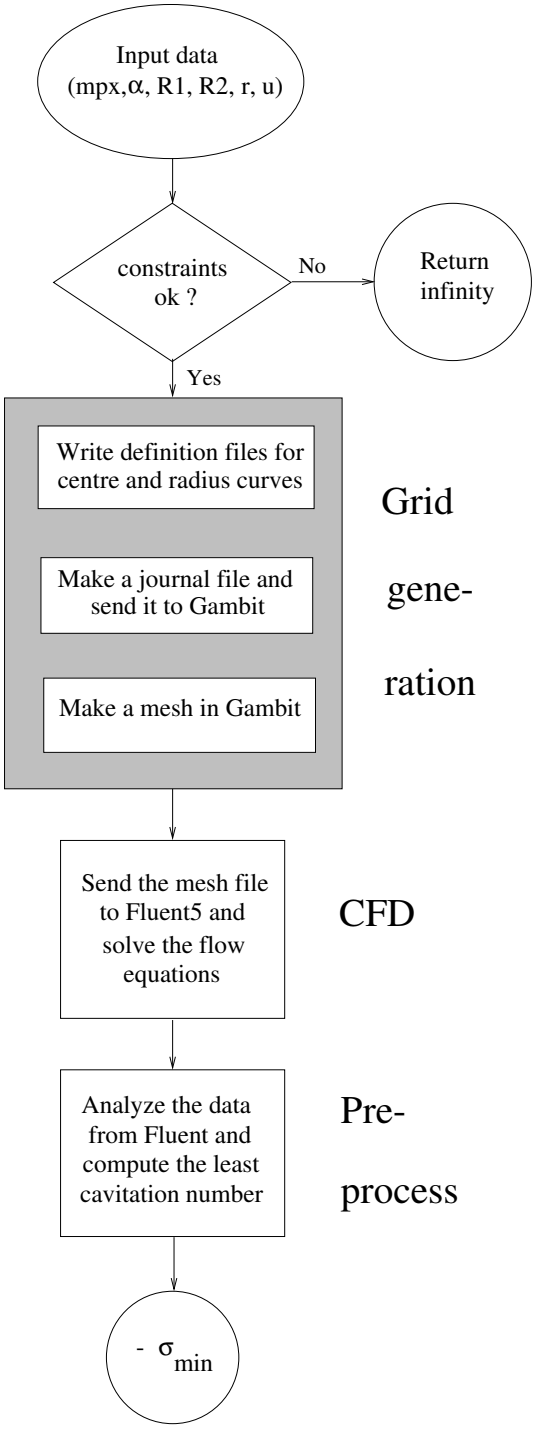


Figure 4.10: Flow chart of the implemented objective function.

in either the mesh generation or the flow solver. Then infinity is returned, possibly after the interrupted program has been terminated manually.

4.7 The optimization procedure

With a parameterized inlet duct model at hand we continue with the optimization problem (4.4), which we now restate for the sake of the reader's convenience:

$$\min_{\xi \in \Omega} f(\xi), \tag{4.16}$$

where

$$f(\xi) = -\sigma_{min}(D(\xi)).$$

The flow problem implicit in f is described in Section 4.5. The vector $\xi \in \mathbb{R}^6$ consists of the parameters for the generic pipe model (Section 4.3.1), i.e.,

$$\xi = (mp_x, \alpha, R1, R2, r, u).$$

With an eye to the geometric constraints in Section 4.3.3, to the possibility to have a structured mesh for all geometries generated from Ω and also to our choice of optimization algorithm in the following paragraph, we choose the following bound constraints for the parameters:

$$\begin{aligned} \Omega = \{\xi \in \mathbb{R}^6 : 2.6 \leq & mp_x \leq 4.8, \\ 31 \leq & \alpha \leq 91, \\ 0.49 \leq & R1, R2 \leq 1.51, \\ 0.15 \leq & r \leq 0.3, \\ 0.2 \leq & u \leq 0.8\}. \end{aligned}$$

Note that we now allow $\alpha > 90^o$ in contrast to the constraints in Section 4.3.3. These constraints, however, were only sufficient to guarantee the possibility to construct the pipe; this construction is still possible.

Because of the properties of the objective function, described in the previous section, we consider the pattern search method (Section 2.1) for our optimization problem. We have chosen the SBA (Section 2.1.5). This is a simple and robust direct method with, being a pattern search method, known convergence properties in the case of bound constraints (Section 2.1.3 and 2.1.4).

The SBA proceeds by taking steps $\Delta\xi_i$ in the coordinate directions and reducing the step size when necessary. The steps should be of different size

Parameter ξ	Initial step size $\Delta\xi$
mp_x	0.15
α	6^o
$R1, R2$	0.15
r	0.02
u	0.1

Table 4.1: Initial step sizes.

for different parameters ξ_i. Their initial sizes $\Delta\xi_i$, given in Table 4.1, are chosen to be approximately one tenth of the allowed interval in Ω for the corresponding parameter.

In order to make it possible to use the same step size in all directions we may change the variables to $\tilde{\xi}_i = \xi_i/\Delta\xi_i$. This is called *scaling* and corresponds to a multiplication $\tilde{\xi} = D\xi$ with the diagonal matrix $D = diag(1/\Delta\xi_i)$. Ω is then changed to $\tilde{\Omega}$ and the new objective function will be $\tilde{f}(\tilde{\xi}) = f(D^{-1}\tilde{\xi})$. In the following, though, we will only refer to f and ξ and just say that we choose a certain step length, the same in all directions, presuming that the scaling problem has been taken care of.

As convergence criteria we have used the step length Δ and the number of function evaluations. This means that the SBA algorithm stops as soon as the step length is reduced below a given minimum step length Δ_{min} or the number of objective function evaluations equals a given number n_{max} of allowed evaluations. The motivation for this second criterion is that f is so expensive to evaluate. (If the user has a certain period of time he may spend waiting for the optimization result he can then calculate n_{max} according to that.)

In Section 4.8 we investigate how the objective function depends on the mesh size $n_\alpha = (n_1, n_2, n_3)$ (Section 4.4.2). To indicate this dependence we write f_α. In that section we estimate the mesh size $n_1 = (80, 7, 12)$ to give mesh independent results, that is, $f = f_1$, and find that for coarser meshes we only have (see, for example, Table 4.3)

$$f \approx f_\alpha \tag{4.17}$$

to different degrees of accuracy. In Table 4.6 we see that f_α might be considerably cheaper to evaluate than f which means that we have found a *model*, or *surrogate function* (Section 2.3), for f.

In the following section we present numerical results from applying the SBA to the problem (4.16). We also investigate the properties of the functions

f_α, to see if they are suited as surrogate functions for f, by applying the SBA to the model problem

$$\min_{\xi \in \Omega} f_\alpha(\xi). \tag{4.18}$$

The stepwise procedure of the SBA makes it possible to use the solution of the currently best design (parameter) as an initial flow solution for the simulations on subsequent geometries. For example, with the mesh $(70, 10, 15)$ the solution for the pipe with parameters $(3.7, 55^o, 1, 1, 0.23, 0.5)$ converged after 432 iterations. When used as an initial solution for the pipe with parameters $(3.7, 55^o, 1, 1.1, 0.23, 0.5)$ a converged solution was reached after 267 iterations.

4.8 Numerical results

The numerical results is divided into two parts. In the first, we perform tests on three different pipes in order to find an appropriate mesh size. In the second, we present and discuss the results from some runs with the optimization algorithm chosen in the previous section.

4.8.1 Simulations for deciding the mesh size

The structure of the mesh is described in Section 4.4.2; what we have to decide now is its size, the successive ratio SR which specifies the stretching of the mesh near the wall, and also appropriate y^+-values. To get a mesh that makes the flow solution mesh independent would be optimal but we are satisfied if we manage to do that for the objective function.

We investigate the necessary mesh sizes for the pipes determined by the vectors $\mathbf{p}_1 = (3.7, 90, 0.5, 1, 0.23, 0.5)$, $\mathbf{p}_2 = (3.7, 90, 0.7, 1, 0.23, 0.5)$ and $\mathbf{p}_3 = (3.7, 55, 1, 1, 0.23, 0.5)$.

The value of y^+ in the wall-adjacent grid point should be between 30 and 500, approximately, in order for the log-law in the Standard $k - \epsilon$ model to be valid (see Section 3.5). In practise, however, it is often suggested that the upper limit for y^+ rather should be 200 in order to get more than one grid point inside the log-law layer. In table 4.2 we have examined the effect of reducing y^+, by a successive adaption of the near-wall mesh, on the objective function. There is no visible effect on neither pipe $\mathbf{p}_1$ nor $\mathbf{p}_3$. Thus $y^+ < 500$ should be enough for our purposes, since nothing seems to be gained from a further reduction. Let us just note that this adaption is a time-efficient method to get a good near-wall mesh. Even though a couple of adaptions

Test of y+ values					
Parameters	*Mesh*	*Size*	*No.it*	*y+*	σ_{min}
(3.7,90,0.5,1,0.23,0.5)	(100,10,15)	70000	283	395	0.64
- " -	Adaption1	98000	313	200	0.64
- " -	Adaption2	139500	340	141	0.64
(3.7,55,1,1,0.23,0.5)	(70,10,15)	49000	432	373	1.5
- " -	Adaption1	68600	465	201	1.5
- " -	Adaption2	146000	493	110	1.5
- " -	Adaption3	171000	510	81	1.5

Table 4.2: Test of y^+ values.

Parameters (3.7,90,0.5,1,0.23,0.5)				
Mesh	*Size*	*No.it*	*y+*	σ_{min}
(120,10,15)	84000	328	406	0.63
(100,10,15)	70000	283	395	0.64
(70,10,15)	49000	429	486	0.67
(100,8,13)	48000	237	543	0.66
(80,7,12)	30800	214	623	0.68
(60,6,7)	12240	190	1584	0.86
(50,4,5)	4800	167	2476	1.12
(40,3,4)	2280	115	3335	1.45

Table 4.3: Test of mesh size.

result in very large mesh sizes, the main part of the iterations in Fluent5.0 are done on a grid of moderate size.

In Table 4.3, 4.4 and 4.5 can be seen the effect of using different mesh sizes on the pipe $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$, respectively. For the meaning of the parameters in the first column, see section 4.4.2.

We see that at least a mesh of the size $(100, 10, 15)$ is necessary to give a mesh independent σ_{min} for all three pipes. For $\mathbf{p}_3$, the mesh size $(80, 6, 7)$ gives the same function value as the larger mesh sizes (Table 4.5). Also, the mesh of size $(80, 7, 12)$ gives approximately the same function value as $(100, 10, 15)$ for $\mathbf{p}_1$ and $\mathbf{p}_2$ and hence we regard that mesh to give a fairly mesh independent solution. For smaller mesh sizes, though, the objective function values deteriorates quickly.

Let us conclude this section with an interesting observation. Of the ex-

Parameters (3.7,90,0.7,1,0.23,0.5)				
Mesh	*Size*	*No.it*	*y+*	σ_{min}
(70,10,15)	49000	282	377	1.12
(100,8,13)	48000	371	534	1.13
(80,7,12))	30800	354	622	1.09
(60,6,7)	12240	216	1547	1.19
(50,4,5)	4800	156	2404	1.24
(40,3,4)	2280	135	3117	1.41

Table 4.4: Test of mesh size.

Parameters (3.7,55,1,1,0.23,0.5)				
Mesh	*Size*	*No.it*	*y+*	σ_{min}
(70,10,15)	49000	432	373	1.5
(80,6,9)	20160	191	972	1.5
(80,6,7)	16320	197	1465	1.5
(60,5,5)	7500	145	2311	1.56
(50,4,5)	4800	146	2245	1.59
(40,3,4)	2280	103	2885	1.66

Table 4.5: Test of mesh size.

Mesh	*Size*	*Approximate run-time in Fluent5*
(100,10,15)	84000	90 min
(80,7,12)	30800	30 min
(60,6,7)	12240	10 min

Table 4.6: Run-times for different meshes.

amined pipes, the minimum cavitation number for $\mathbf{p}_2$ is bigger than that for $\mathbf{p}_1$ and less than that for $\mathbf{p}_3$. This order is kept distinct by the meshes $(80, 7, 12)$ and $(60, 6, 7)$, though it is "weaker" with the latter. However, with the mesh size $(40, 3, 4)$ this order is lost. The run-times in Fluent5 for the different meshes can be seen in Table 4.6.

4.8.2 Optimization results

In this section we present some results from applying the SBA to the problems (4.18). We use two different meshes $n_1 = (80, 7, 12)$ and $n_2 = (60, 6, 7)$ and hence get two corresponding functions f_1 and f_2. We consider f_1 to be the idealized function f in problem (4.16). Most of the pictures and tables can be found in Appendix B.

Five runs with the SBA have been done. Runs $1, 2$ and 3 are done with function f_1 from three different initial vectors; the results are presented graphically in Figure 4.11. Runs 4 and 5 are done with function f_2 from two different initial vectors and the result from run 4 is presented in Figure B.1. The initial and final (optimized) radius and centre curves from runs 1,2 and 3 are shown in Figures B.2, B.3 and B.4, respectively. A comparison of the final centre and radius curves from runs 1, 2 and 3 can be seen in Figure B.5. The changes in each variable during the runs 1 to 4 are shown graphically in Figures B.6, B.7, B.8 and B.9. The essential numerical data from all the runs is found in Tables B.1 and B.2.

Analysis

We first note how the objective values in runs 1,2 and 3 decrease dramatically during the first 10-15 evaluations, after which the iterations no longer yield significant improvements (Figure 4.11). The final objective values in these runs are almost equal: -2.56, -2.56 and -2.55, respectively. The initial vectors, which were chosen quite arbitrarily, seem to be far from any local minima. The changes in the variables seem to be significant up to approximately 30 evaluations (Figures B.6, B.7 and B.8). At the end of the optimization changes occur in neither the variables nor the objective, which gives us a reason to believe that the final vectors are close to local minima.

Figures B.6, B.7 and B.8 indicate that all variables are important. We analyze the results of runs 1, 2 and 3 for each variable in turn.

mp_x : Takes final values between 4 and 4.7, which indicates that there are no unique optimal value for this variable. See also Figure B.5.

α : All final values are very close to 90^o.

$R1$: Takes final values between 0.8 and 0.96.

$R2$: Takes final values just above 0.6.

r : All final values are between 0.27 and 0.29.

u : The final values are quite dispersed (between 0.2 and 0.4).

It can be seen from Figure B.5 that the final values of the variables r and u in all the first three runs result in radius curves each one having an inflection point, in contrast to all initial radius curves.

Runs 4 and 5 are done with f_2 as the objective function. Run 4 starts from the same initial vector as run 1. Notably, the sequence of iterates from run 4 is exactly the same as the corresponding first part of the sequence of iterates from run 3 (which at least almost can be concluded from Tables B.1 and B.2 and Figures B.6 and B.9; the author can affirm that this is indeed the case). Run 5 starts from the final vector from run 2. As expected (since the final vector of run 2 should be close to a local minimum) only a small improvement in the objective are done in the course of 30 evaluations. However, it is interesting to note that the value of f_1 for the final vector of run 5 is actually *higher* than the value of f_1 for the initial vector of the same run (Table B.2).

Conclusions

From the results it is clear that the implemented procedure gives significantly improved inlet duct designs with respect to the cavitation property, at least when calculated as a percentage of the initial objective values.

The cavitation objective function appears to be quite "nice": all runs result in approximately the same objective value and the optimized variables from the different runs are not very far from each other. It would be interesting to know if we have found distinct local minima, or if they correspond to the "flat region " of one and the same local minimum. If we consider that significant changes occur in the variables for some iterations even when no marked decrease in the objective any longer can be seen, the latter seems plausible.

What conclusions can we draw concerning the optimal design of the pipe? All runs indicate that the angle of inclination α should be around 90^o; actually somewhat above that. The radius of the second bend should be around 0.6 m which is about tree quarters of what the radius of the first bend should be. The variable u is the one that differs the most between the runs. However, the general trend of the optimized radius curve is to have a point of

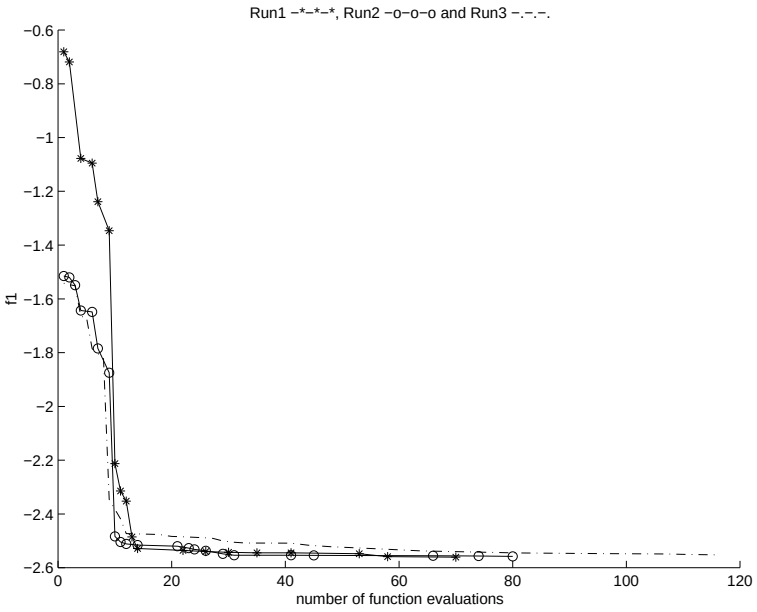


Figure 4.11: Optimization results from runs 1,2 and 3.

inflection (Figure B.5), which none of the initial radius curves had. Though the exact location of this point differs, it appears approximately after one third of the length of the centre curve (where the last $0.6\,m$ has not been taken into account).

The variables mp_x and u are obviously related to each other. We remember that u decides where along the duct the radius should increase or decrease (the effect of r). It is natural that this depends on mp_x which controls the position of the inclined part of the duct.

Run 4 shows that f_2 is a very good surrogate function for f_1, at least when the initial vector is far from a local optimum. f_2 is about one third as expensive to evaluate as f_1, see Table 4.6. On the other hand, the decrease of f_2 in run 5 actually corresponds to an *increase* in f_1, which we regard as the idealized function. Hence, results obtained from f_2 can be very helpful when analyzing f_1, since f_2 is less expensive and *sometimes* reflects the behaviour of f_1. However, as shown by run 5, information obtained from f_2 can only with great cautiousness be transferred to f_1.

4.9 Concluding remarks and future research

In this chapter we have described the construction of geometric and flow models for the problem formulated in Section 4.2. The SBA applied to this problem results in a dramatic decrease in the cavitation objective function within only 10-15 function evaluations. Moreover, we have found that the mesh size is an appropriate mean for constructing surrogate functions, though these surrogates must be used with care.

The objective function seems to behave quite "nicely". As mentioned earlier, not much can be said theoretically about the objective. The numerical results, however, indicate the existence of a minimum with a quite large region around it in the variable space where the function values are almost optimal. The existence of this kind of a minimum would make it suitable to optimize the pipe shape with respect also to other properties, such as the pressure drop etc. The cavitation objective function is related to the reliability of the waterjet propulsion, whereas a pressure drop objective function would rather measure the efficiency.

One thing that could be done to numerically investigate the question of regularity of the objective function, is to check the location in the pipe where the cavitation number attains its minimum. If this location does not "jump around" (see discussion in Section 4.6), we have a reason to believe that the objective function is continuously differentiable.

One important work still to be done is to validate the models and the numerical results with real data. Even though the overall model of the waterjet inlet duct was suggested by Kamewa, our choice of parameters has been done quite arbitrarily and the resulting pipe shapes are rather simple. For example, the cross section is circular, whereas in reality it might change from rectangular to elliptic. It would be easy to generalize our model to elliptic cross sections. Furthermore, another model of the inlet of the waterjet inlet ducting might be preferable. Instead of having a fixed inlet velocity one could have a box around the inlet where one sets a free-stream velocity and then computes the inlet velocity.

The parametric model has not been fully evaluated. It should be compared to other models and the relative importance of each parameter should be investigated with, for example, some statistical tool such as an analysis of variance (ANOVA). There is no obvious way to construct the generic pipe model. In a good parameterization the parameters are all equally important and more or less independent of each other. The parameters for the centre curve are physical and hence "natural", but this is not the case for the parameters r and u for the radius curve. The numerical results indicate that the radius curve should have an inflection point. Perhaps it would be

more "natural" to have a parameterization that controls the location of this inflection point. The problem with non-physical parameters is that it is more difficult to see the effect of a change in one parameter and to avoid redundancy, i.e. to avoid that different parameter values result in (almost) the same objective function value or even the same pipe geometry. (However, there is no safe way to avoid redundancy in this sense, since it might be an inherent property of the objective function.) Finally, the optimization result may be harder to interpret.

Actually, it seems as if the choice of parameters is a first intuitive optimization and in our case it can not be made by a machine since the full problem is infinite dimensional.

It would be interesting to know how the shape of our optimized pipes relates to the shapes of existing pipes. This question, the relevance of the geometric model and the possible addition of parameters should be discussed with an waterjet engineer expert.

We have used the Sherif-Boice optimization algorithm, which should be compared to other grid methods or some approximation method (see the introduction to Chapter 2). It would be desirable to implement an algorithm that allows for more general constraints than the bound constraints used here. (A pattern search algorithm for linearly constrained problems is presented in [20].) Furthermore, it would be interesting to implement and test an algorithm that uses the investigated surrogate functions f_α.

To conclude, we dare to say that we have reached Aim 2 set up in the Introduction. However, to even come close to Aim 1 a lot of validation and calibration of the models remain to be done. At the present stage not much can be said quantitatively about the results obtained.

It has been an interesting problem to work with, much because it concerns such a "practical" thing as a three-dimensional shape. Also, the problem combines the subjects of optimization and computational fluid dynamics in a challenging manner. To take the problem further is probably not just a question of improving the optimization algorithms and the CFD algorithms separate from each other, but rather combining the two. For example, the commercial software available is mainly developed for the analysis of a single geometry. There is not much effect of using the solution from one geometry as an initial guess to a neighbouring geometry (as it would be natural to do in connection to optimization). Probably, much remain to be done here, from a numerical as well as a fluid dynamics point of view.

Chapter 5

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 [12] and some pattern search methods [18]. 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 2.2.2). 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 [12]. 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 [27] 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 [30].

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

5.1 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 [8] we have the following two theorems of which the second characterizes positive spans.

Theorem 2. *Suppose $\{a_1, \cdots, a_r\}$ positively spans $\mathbb{R}^n$. Then $\{a_2, \cdots, a_r\}$ linearly spans $\mathbb{R}^n$.*

Theorem 3. *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 4. *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 2 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 3 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. □

5.2 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 [12]), however, they are given by $n+1$ vectors in $\mathbb{R}^n$, whose convex hull is n-dimensional. 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 5. *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,

$$\begin{aligned} \text{constant} &= \|v_i - v_j\|^2 = \|v_i\|^2 + \|v_j\|^2 - 2\, v_i \bullet v_j \\ &= 2 - 2\, v_i \bullet v_j, \forall i \neq j. \end{aligned} \tag{5.1}$$

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 6. *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 3. From Theorem 5 it follows that $\{v_1, \cdots, v_{n+1}\}$ linearly spans $\mathbb{R}^n$. Let

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

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 (5.1) and $v_i \bullet v_j = \alpha$, $\forall i \neq j$, so obtain

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

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

$$\sum_{i=1}^{n+1} v_i = 0. \tag{5.2}$$

Theorem 3 then 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 7. *Let $\{v_1, \cdots, v_{n+1}\}$ be a normalized regular simplex. Then*

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

Proof. From (5.1) 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 (5.2). □

Theorem 8. *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 (5.1) and Lemma 7 implies that

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

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

$$\begin{aligned} 2\Big(1 + \frac{1}{n-1}\Big) &= 1 + \gamma^2(1-\beta)^2, \\ 1 &= 1/\gamma^2 + \beta^2, \end{aligned}$$

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 solution

$$\begin{aligned} \beta &= -\frac{1}{n}, \\ \gamma &= \sqrt{\frac{1 + n + 1/(n-1)}{1+n}}, \end{aligned}$$

which ends the proof. □

The performance of a pattern search algorithm (Section 2.1.1) 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 [30]). The following theorem states the corresponding result for a normalized regular simplex (which, according to Theorem 6, is a minimal positive basis).

Theorem 9. *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 7 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 4 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.

□

Finally, we consider the classical simplex method in [12]. 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. ,

$$v = -v_m + \frac{2}{n} \sum_{i\neq m}^{n} v_i. \tag{5.3}$$

For $n = 2$, this can be seen in Figure 5.1, and the simplices obtained from such reflexions form a regular pattern (Figure 5.2).

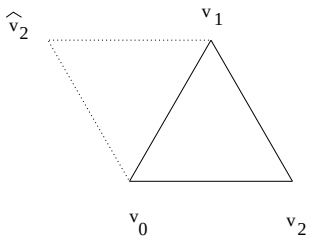


Figure 5.1: Reflexion of a regular simplex.

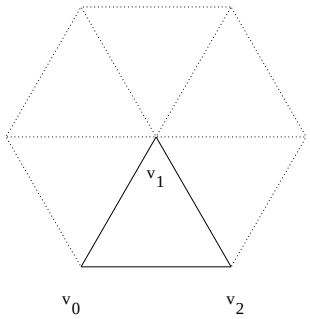


Figure 5.2: In $\mathbb{R}^2$ the sequence of points generated by repeated reflexions of any of the vertices of a simplex stays on a lattice. Hence, if one vertex, v_1 in the figure, is kept fixed, reflexion of the remaining two vertices produce only a finite number of new points (namely 4).

Hence, the method can only produce a finite sequence of different simplices in a bounded domain of $\mathbb{R}^2$. In [27], Powell shows that repeated reflexions may 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 (5.3) 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

$$\begin{cases} w_0 = v_0,\ w_1 = v_1, \\ w_k = -w_{k-2} + \frac{2}{n}(w_{k-1} + \sum_{i=2}^{n} v_i), \quad k > 1. \end{cases} \tag{5.4}$$

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 (5.4) 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 7 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,

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

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

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

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 A

NURB curves

We explain what a NURB curve is and state some elementary properties of it. We describe how a continuous curve consisting of circular arcs and straight line segments can be represented by a NURB curve. Everything in this appendix can be found in reference [25].

A.1 NURB curves

A *non-rational uniform B-spline curve*, in shorthand a NURB curve, of degree p is defined by (see [25])

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

where $\{\mathbf{P}_i\}$ are called the *control points* which together form a polygon called the *control polygon*, $w_i > 0$ are the *weights* and $N_{i,p}$ is the p:th B-spline basis function. For each $i \ge 0$ it is recursively defined by

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

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}, \qquad p > 0$$

where $U = (u_0, u_1, \cdots, u_m)$ is the *knot vector* with $u_0 = u_1 = \cdots = u_p = 0 \le u_{p+1} \le \cdots \le u_{m-p} = \cdots = u_m = 1$ and $m = n + p + 1$. Hence the first $p + 1$ knots equals 0 and the last $p + 1$ equals 1.

We now list some relevant properties of a NURB curve $\mathbf{C}$ ([25]).

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

- $\mathbf{C}$ is infinitely differentiable in the interior of a knot interval and at least $p-k$ times differentiable at a knot of multiplicity k.
- It lies in the convex hull of the control polygon.
- If all $w_i = 1$ then $\sum_{i=0}^{n} N_{i,p}(u)w_i = 1$ and the NURB curve is an ordinary B-spline curve.
- The effect of increasing one weight in proportion to the others is the NURB curve being pulled towards the corresponding control point.
- The derivatives at the end points are given by

$$\mathbf{C}'(0) = \frac{p}{u_{p+1}}\frac{w_1}{w_0}(\mathbf{P}_1 - \mathbf{P}_0), \quad \mathbf{C}'(1) = \frac{p}{1-u_{m-p-1}}\frac{w_{n-1}}{w_n}(\mathbf{P}_n - \mathbf{P}_{n-1}).$$

A.2 Representation by NURBs

We show how the centre curve in Section 4.3.1 can be represented by a NURB curve of degree 2 ([25]). The centre curve is continuous and consists of five segments, each one either a circular arc or a straight line.

Given $mp_x, R1, R2, \alpha$ we compute $\mathbf{Q}_i, i = 0, \cdots, 7$ which are the end points of the five segments that constitute the centre curve apart from the "midpoint" $\mathbf{Q}_3 = (mp_x, 1.5, 0)$ and the left end point $\mathbf{Q}_6 = (7.4, 3, 0)$ of the last straight line segment of length $0.6\,m$, see figure A.1. Then we calculate $\mathbf{R}_i, i = 1, \cdots, 7$ where $\mathbf{R}_i$ are the midpoint of the line segment $\mathbf{Q}_{i-1}\mathbf{Q}_i$ for $i = 1, 3, 4, 6, 7$ and the point of intersection between the lines $\overleftrightarrow{\mathbf{Q}_{i-2}\mathbf{Q}_{i-1}}$ and $\overleftrightarrow{\mathbf{Q}_i\mathbf{Q}_{i+1}}$ for $i = 2, 5$.

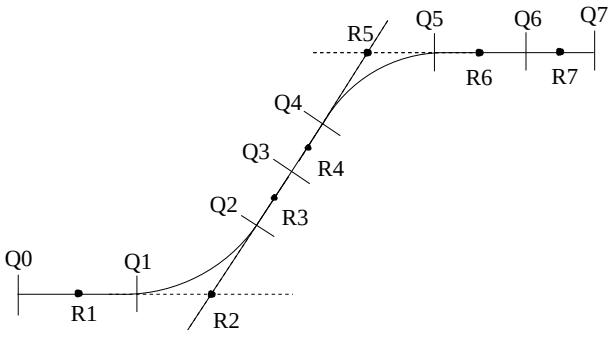


Figure A.1: The control polygon for the NURBs representation of the centre curve.

As a control polygon for the centre curve we now take

$$\{\mathbf{Q}_0, \mathbf{R}_1, \mathbf{Q}_1, \mathbf{R}_2, \mathbf{Q}_2, \cdots, \mathbf{Q}_6, \mathbf{R}_7, \mathbf{Q}_7\}$$

with weights

$$w_i = \begin{cases} cos(\alpha/2), & i = 3, 9, \\ 1, & i = 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14. \end{cases}$$

To get an approximation of the arc length parametrization, which facilitates all work with the curve, such as meshing, we let $\bar{u}_0 = 0, \bar{u}_{n=1}$ and

$$\bar{u}_i = \bar{u}_{i-1} + \frac{|\mathbf{Q}_i - \mathbf{Q}_{i-1}|}{d}, \qquad i = 1, \cdots, 6,$$

where $d = \sum_1^7 |\mathbf{Q}_i - \mathbf{Q}_{i-1}|$. We choose the knot vector

$$U = \{0, 0, 0, \bar{u}_1, \bar{u}_1, \bar{u}_2, \bar{u}_2, \cdots, \bar{u}_6, \bar{u}_6, 1, 1, 1\}.$$

Appendix B

Illustrations of numerical results

This appendix contains some pictures and tables of the numerical results discussed in Section 4.8.2.

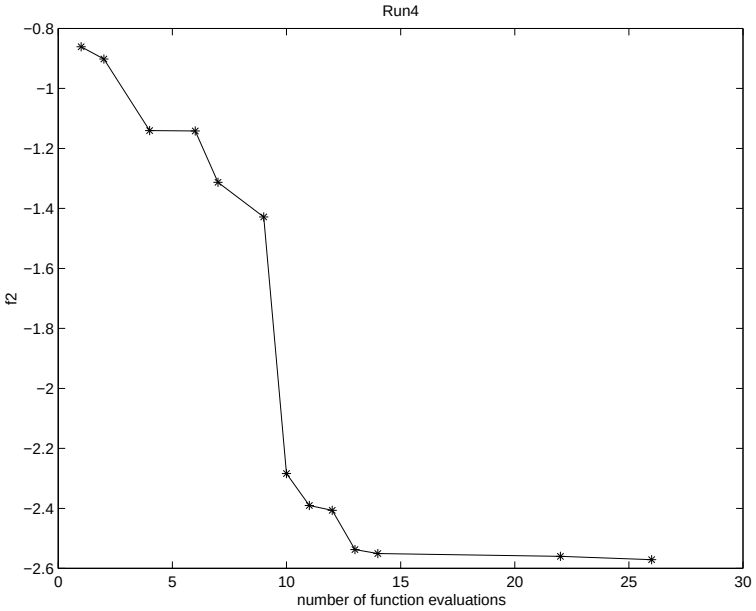


Figure B.1: Optimization results from run 4.

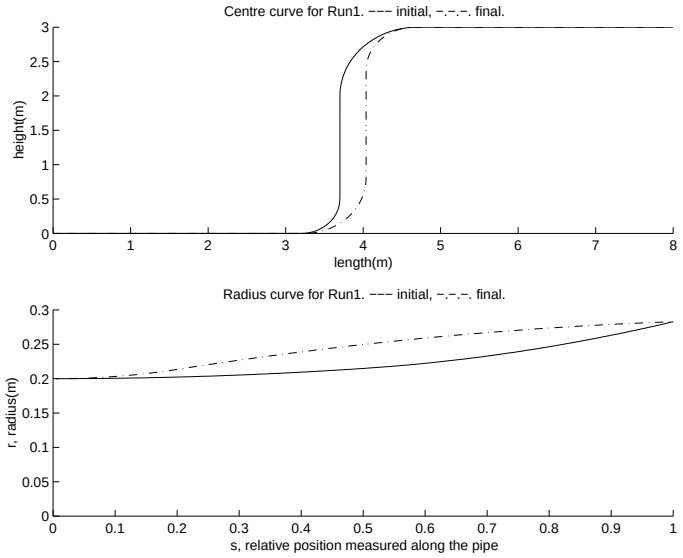


Figure B.2: Initial and final radius and centre curves from run 1.

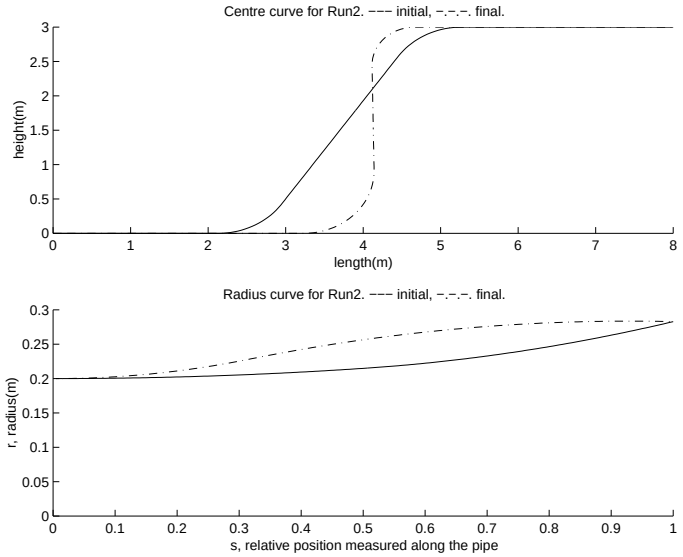


Figure B.3: Initial and final radius and centre curves from run 2.

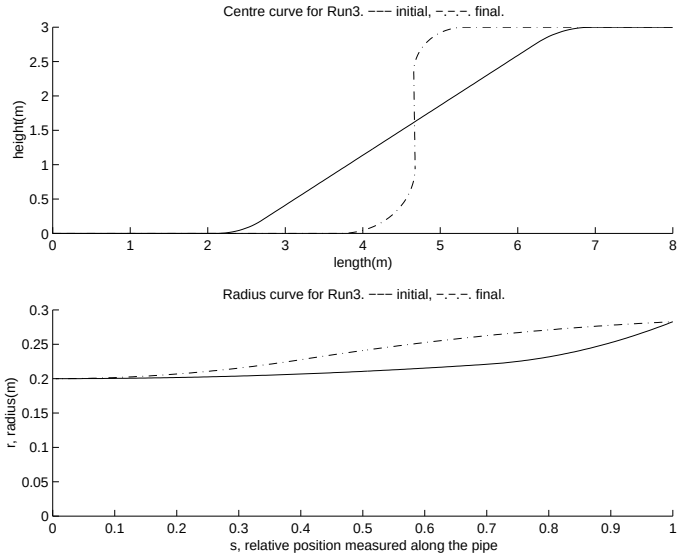


Figure B.4: Initial and final radius and centre curves from run 3.

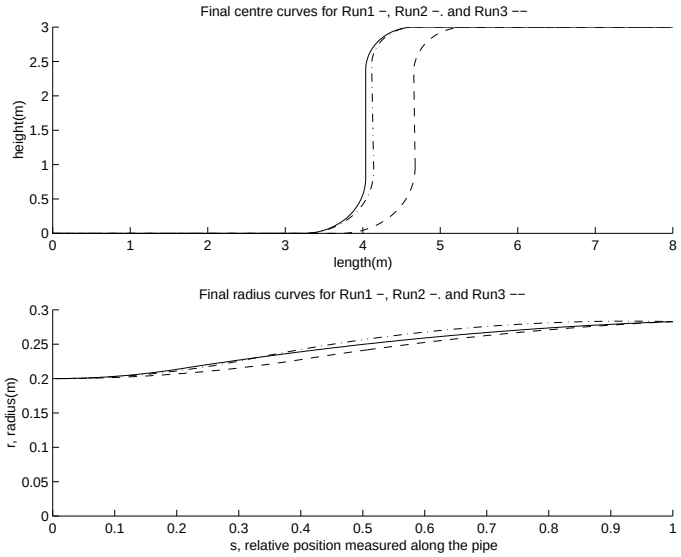


Figure B.5: The final radius and centre curves from runs 1, 2 and 3.

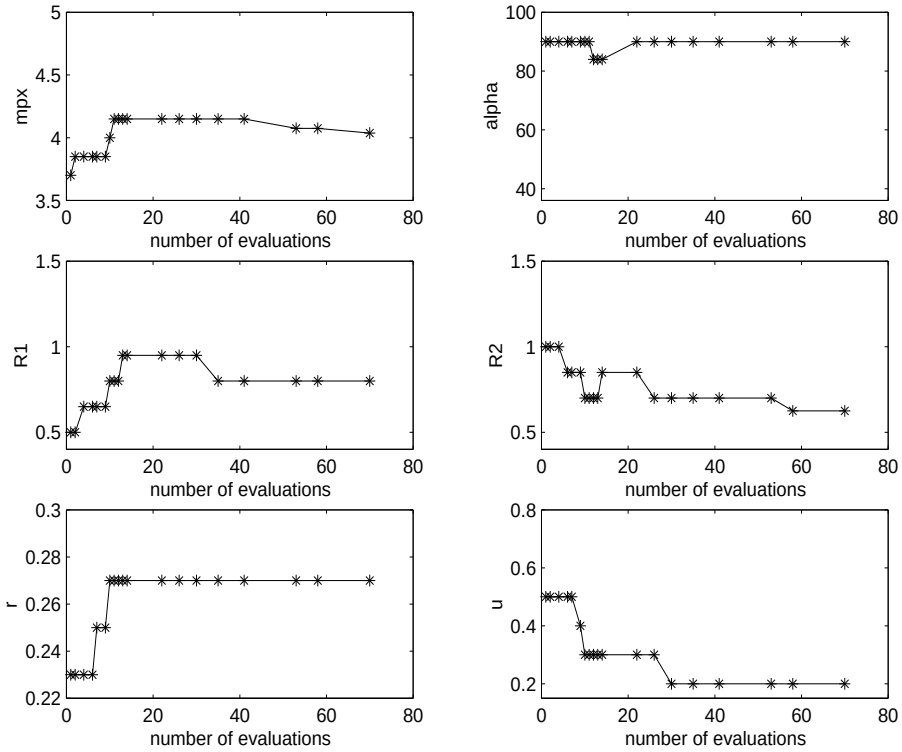


Figure B.6: Changes in each variable during run 1.

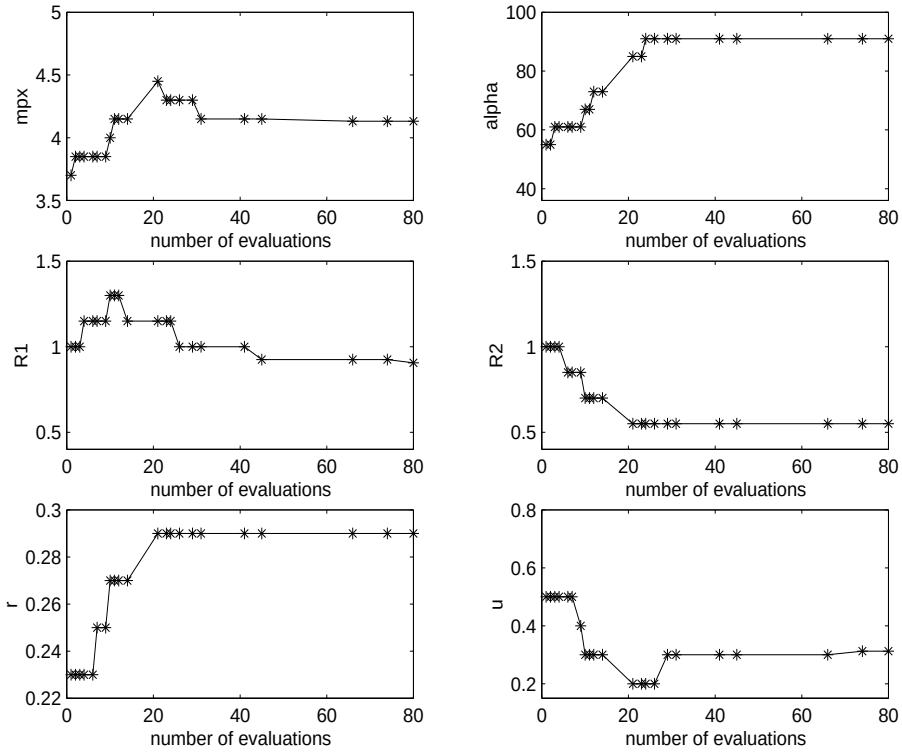


Figure B.7: Changes in each variable during run 2.

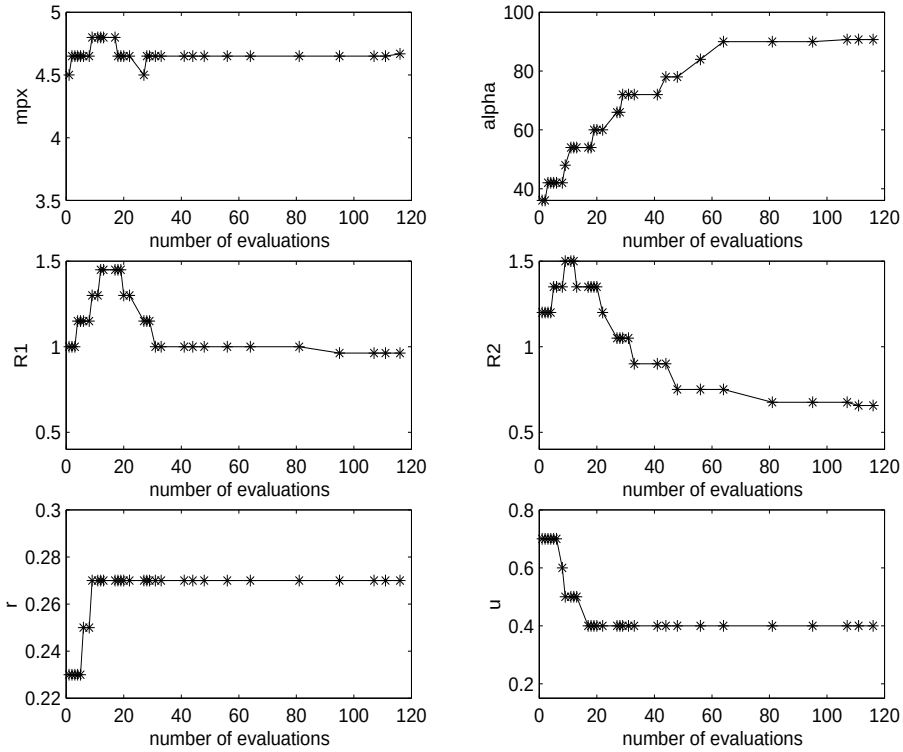


Figure B.8: Changes in each variable during run 3.

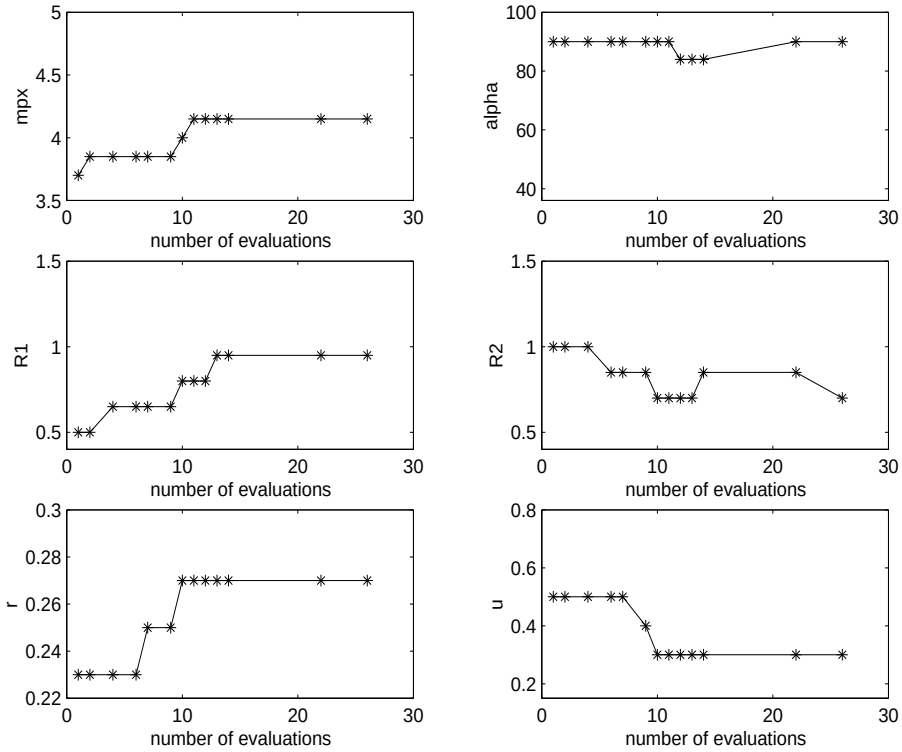


Figure B.9: Changes in each variable during run 4.

Optimization of the function $\mathbf{f_1}$

Run 1: 70 function evaluations

	Initial value	*26 eval.*	*Final value*
mp_x	3.7	4.15	4.0375
α	90	90	90
$R1$	0.5	0.95	0.8
$R2$	1	0.7	0.625
r	0.23	0.27	0.27
u	0.5	0.3	0.2
Function f_1	-0.6809	-2.54	-2.5605
Step length Δ	1		0.25

Run 2: 80 function evaluations

	Initial value	*Final value*
mp_x	3.7	4.1313
α	55	91
$R1$	1.0	0.9063
$R2$	1.0	0.625
r	0.23	0.29
u	0.5	0.3125
Function f_1	-1.5151	-2.5578
Step length Δ	1	0.125

Run 3: 120 function evaluations

	Initial value	*Final value*
mp_x	4.5	4.669
α	36	90.750
$R1$	1.0	0.963
$R2$	1.2	0.656
r	0.23	0.27
u	0.7	0.4
Function f_1	-1.5421	-2.5522
Step length Δ	1.0	0.125

Table B.1: Numerical results from runs 1,2 and 3.

Optimization of the function f_2

Run 4: 30 function evaluations		
	Initial value	*Final value*
mp_x	3.7	4.15
α	90	90
$R1$	0.5	0.95
$R2$	1.0	0.7
r	0.23	0.27
u	0.5	0.3
Function f_1	-0.8609	-2.5711
Step length Δ	1.0	1.0

Run 5: 30 function evaluations		
	Initial value	*Final value*
mp_x	4.1313	3.9813
α	91	91
$R1$	0.9063	0.7938
$R2$	0.55	0.5125
r	0.29	0.29
u	0.3125	0.3125
Function f_2	-2.5893	-2.6179
Function f_1	-2.5571	-2.2483
Step length Δ	1.0	1.0

Table B.2: Numerical results from runs 4 and 5.

Bibliography

[1] D.J. Acheson, *Elementary Fluid Dynamics*, Oxford University Press, Oxford, 1990.

[2] J.-F. M. Barthelemy, R. T. Haftka, Approximation Concepts for Optimum Structural Design - A Review, *Structural Optimization*, 5:129-144, 1993.

[3] 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*, Vol. 17, No. 1, p. 1-13, 1999.

[4] 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, p. 49-58, 1998.

[5] A.R. Conn, Ph.L. Toint, An Algorithm Using Quadratic Interpolation for Unconstrained Derivative Free Optimization, *Nonlinear Optimization and Applications*, Plenum Publishing, New York, p. 27-47, 1996.

[6] A.R. Conn, K. Scheinberg, Ph.L. Toint, A Derivative Free Optimization Algorithm in Practice, Proceedings of the AIAA St Louis Conference, 1998.

[7] A.J. Chorin, J.E. Marsden, *A Mathematical Introduction to Fluid Mechanics*, 3rd ed., Springer-Verlag, New York, 1993.

[8] O.L. Davies, *The Design and Analysis of Industrial Experiments*, Hafner Publishing Company, New York, 1954.

[9] 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.

[10] J.E. Dennis, V. Torczon, Managing Approximation Models in Optimization, *Multidisciplinary Design Optimization: State of the Art*, SIAM, Philadelphia, 1996.

[11] *Fluent5 User's Guide Volume 1-4*, Fluent Incorporated, 1998.

[12] G.R Hext, F.R. Himsworth, W. Spendley, Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation, *Technometrics*, 4, p. 441-461, 1962.

[13] J.O. Hinze, *Turbulence*, 2nd ed., McGraw-Hill, 1975.

[14] W. Hock, K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, 1981.

[15] R. Hooke, T.A. Jeeves, Direct Search Solution of Numerical and Statistical Problems, *Journal of the Association for the Computing Machinery*, 8, p. 212-229, 1961.

[16] W.M. Lai, D. Rubin, E. Krempl, *Introduction to Continuum Mechanics*, 3d ed., Butterworth-Heinemann Ltd, Oxford, 1993.

[17] B.E. Launder, D.B. Spalding, The Numerical Computation of Turbulent Flows, *Computer Methods in Applied Mechanics and Engineering*, vol. 3, p. 269-289, 1974.

[18] 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.

[19] 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.

[20] R.M. Lewis, V. Torczon, Pattern Search Methods for Linearly Constrained Minimization, *SIAM Journal of Optimization*, Vol. 10, p. 917-941, 2000.

[21] W. Malalasekera, H.K. Versteeg, *An Introduction to Computational Fluid Dynamics. The Finite Volume Method*, Addison Wesley Longman Limited, 1995.

[22] 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.

[23] K.I.M. McKinnon, Convergence of the Nelder-Mead Simplex Method to a Nonstationary Point, *SIAM Journal on Optimization*, 9, p. 148-158, 1998.

[24] J.A. Nelder, R.Mead, A Simplex Method for Function Minimization, *Computer Journal*, 7, p. 308-313, 1965.

[25] L. Piegl, W. Tiller, *The NURBs Book*, 2nd ed., Springer-Verlag Berlin Heidelberg New York, 1997.

[26] 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, p. 51-67, 1994.

[27] M.J.D. Powell, Direct Search Algorithms for Optimization Calculations, *Acta Numerica*, 7, p. 287-336, 1998.

[28] 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.

[29] Y.S. Sherif, B.A. Boice, An Efficient Algorithm for Solving the Unconstrained Nonlinear Multivariable Minimization Problems, *Advances in Engineering Software*, 17, p. 29-37, 1993.

[30] V. Torczon, On the Convergence of Pattern Search Algorithms, *SIAM Journal on Optimization*, 7, p. 1-25, 1997.

[31] V. Torczon, Managing Approximation Models in Optimization, *Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* , Bellevue, Washington, September 4-6, 1996.

[32] D.J. Tritton, *Physical Fluid Dynamics*, 2nd ed., Oxford University Press, Oxford, 1988.

[33] M.W. Trosset, V. Torczon, Numerical Optimization Using Computer Experiments, Technical Report 97-38, ICASE, NASA Langley Research Center, Hampton, VA, 1997.

[34] 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.