Numerical Optimization of the Voith-Schneider Propeller

Dirk Jürgens, Michael Palm, Sebastian Singer und Karsten Urban

Preprint Series: 2005-04

Fakultät für Mathematik und Wirtschaftswissenschaften
UNIVERSITÄT ULM

Numerical Optimization of the Voith-Schneider® Propeller

Dirk Jürgens* Michael Palm* Sebastian Singer† Karsten Urban†

June 20, 2005

Abstract

The Voith-Schneider® Propeller (VSP) is an efficient ship propeller that was invented by Ernst Leo Schneider (1894–1975) in 1927. It is a complex propeller having several parameters determining the behavior of the VSP. One of them is the Blade Steering Curve (BSC) which describes the angle of a propeller blade for a time instant. We investigate the choice of this BSC in order to optimize the efficiency of the VSP. We use numerical optimization and Computational Fluid Dynamics (CFD), in particular the Vortex-Lattice Method for the optimization and Finite Volume Method for validation. We obtain a significant improvement of 4.7%.

AMS MSC (2000): 78M25, 76B75, 76D05
Keywords: Numerical optimization, computational fluid dynamics.

1 Introduction

The importance of numerical simulations in various fields of application is by now widely accepted. They are in particular relevant in cases where experiments are too costly, too dangerous or even impossible when the process to be simulated is not even observable in experiments.

If such a process is in addition equipped with several parameters, it is a natural question to ask for a choice of parameters so that the process behaves in some sense *optimal*. It is immediately clear that such questions arise e.g. in engineering, science, medicine, economy and other areas.

In this paper, we describe the optimization of the Voith-Schneider® Propeller (VSP), a complex and efficient propeller powering and steering ships. Such a complex technical machine of course has many parameters that could be subject to an optimization. On the other hand, there is also a whole variety of targets in such a scenario that one could possibly be interested in optimizing, e.g. minimize fuel consumption, maximize efficiency, maximize speed, minimize noise, and so on. In this sense, the optimization problem considered in this paper is a prototype for a wide class of complex technical problems.

From this description it is already clear that not all possible questions can be investigated at once. Thus, this paper introduces into this field by using numerical optimization and Computational Fluid Dynamics (CFD) to optimize the efficiency of the VSP using one parameter only, namely the so-called *blade steering curve* (BSC).

*Voith Turbo Marine GmbH & Co. KG, Alexanderstr. 18, 89522 Heidenheim, Germany, {dirk.juergens,michael.palm}@voith.com
†University of Ulm, Dept. of Numerical Mathematics, Helmholtzstr. 18, 89069 Ulm, Germany, {sebastian.singer,karsten.urban}@uni-ulm.de

This paper is organized as follows. In Section 2 we describe the Voith-Schneider® Propeller, where we focus on those parts that enter the optimization problem. Section 3 is devoted to the description of the mathematical modelling, in particular to the formulation of the optimization problem and its discretization. The numerical simulation of the flow induced by the VSP is performed by CFD-methods. In Section 4, we describe the numerical methods used for the numerical optimization problem. The mathematical properties of the optimization problem described in Section 3 lead us to specific choices of optimization methods, local and global, which are detailed in Section 4.2. Our numerical results are described in Section 5 and we end this paper in Section 6 with conclusions and an outlook to further research activities.

2 The Voith-Schneider® Propeller

In 1927 this ship's propulsion system, the only one of its kind in the world, was developed by Voith based on the basic idea of the Austrian engineer Ernst Leo Schneider (1894–1975). In Figure 1 on the left-hand side two Voith-Schneider® Propellers integrated in a tug boat are shown, the right-hand side shows a sectional drawing of the VSP. The propeller allows thrust of any magnitude to be generated in any direction quickly, precisely and in a continuously variable manner. It combines propulsion and steering in a single unit.

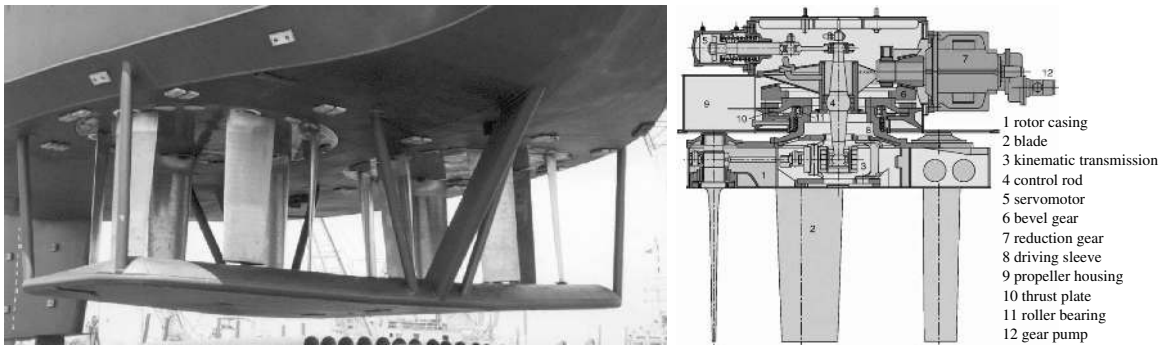

Figure 1: Voith-Schneider® Propeller

On the Voith-Schneider® Propeller, a rotor casing (1) which ends flush with the ship's bottom is fitted with a number of axially parallel blades (2) and rotates about a vertical axis. To generate thrust, each of the propeller blades performs an oscillating motion about its own axis (similar to the motion of the tail fin of a fish). This is superimposed by a uniform rotary motion. Blade excursion determines the amount of thrust, while the phase angle of between 0° and 360° determines its direction. As a result, the same amount of thrust can be generated in any direction, making this the ideal variable-pitch propeller. Both variables – the magnitude and the direction of thrust – are controlled by a control rod (4), which is pulled by two servomotors (5) and a mechanical kinematic transmission (3).

Due to its abilities, the Voith-Schneider® Propeller is so far used in water tractors, ferries, minehunters and special ships floating cranes or oceanographic research ships. For all those different purposes there are different propeller types, that differ in size, number of blades, shape of blades, blade steering and many other construction details. All those parameters have great influence on the characteristics of the propeller.

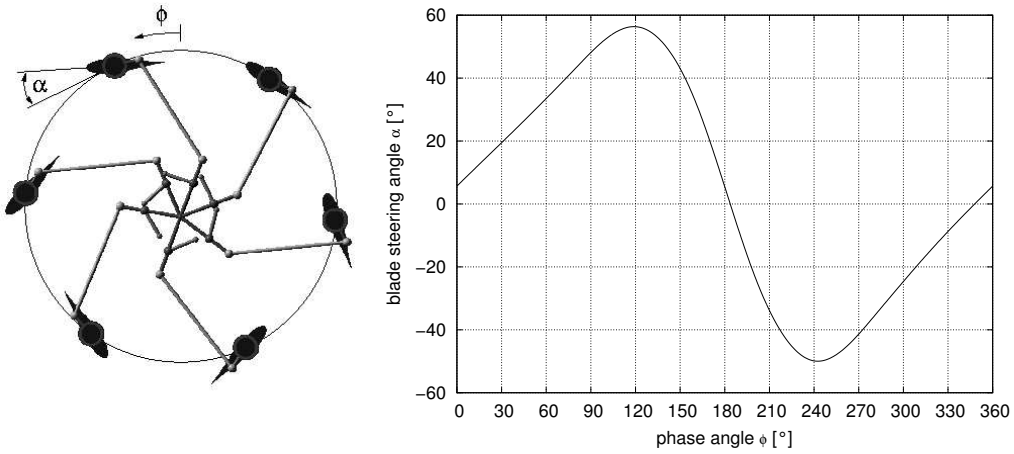

Figure 2: blade steering curve (BSC).

The steering of the blade, i.e., the oscillating motion mentioned above, can be described as steering angle (α in Figure 2) in dependency of the rotor (1) position (Φ in Figure 2). We call this the *blade steering curve (BSC)*. This curve is not only defining the magnitude and direction of thrust, it obviously also has enormeous influence on the flow around the propeller and therefore on its efficiency. It was therefore chosen as our variable parameter for optimization. Our problem under investigation is thus to find an optimal BSC in the sense that the efficiency of the VSP is maximal for the determined BSC as compared with all others. First we have to specify the mathematical modelling of the problem.

3 The Optimization Problem

In principle, the question of determining an optimal BSC could also be investigated by physical experiments. In fact, Voith has a powerful water tank which allows several experiments. For such an optimization, however, one would need a huge amount of experiments, which is both expensive and time-consuming. Moreover, there would be no guarantee that the determined curve is in some sense optimal. Furthermore, one encounters scaling problems when comparing results obtained in the model to data measured in reality. As an example, the effect of the Reynolds number does not scale appropriately, because the experiments are carried out accordingly to Froude's Law.

Therefore, we use numerical simulations as a base to determine the relevant parameters numerically. Thus, we need a mathematical model for the flow induced by the VSP and a mathematical formulation of the corresponding optimization problem.

3.1 The Mathematical Model

We start by introducing the mathematical model which does not yet involve any discretization for numerical purposes. This is done in a second step.

3.1.1 The Blade Steering Curve (BSC)

Let us first describe in mathematical terms an appropriate BSC. If we parameterize the rotation of a propeller by its angle, we obtain the parameter interval $[0, 2\pi]$. Thus, we are looking for functions

$$s : [0, 2\pi] \to [-\pi, \pi], \Phi \to s(\Phi) = \alpha$$

representing a BSC, with Φ and α as introduced in Section 2. Obviously, such a function needs to be *periodic*, i.e., $s(0) = s(2\pi)$. Finally, due to technical reasons, we do not allow rapid changes of the orientation of the blade and this corresponds to smoothness of the BSC, say $s \in C^2[0, 2\pi]$. Thus we call the space

$$C^2_{2\pi} := \{s : [0, 2\pi] \to [-\pi, \pi] : \ s \in C^2[0, 2\pi], \ s(0) = s(2\pi)\} \tag{3.1}$$

the *set of valid BSC*.

One should note, however, that not every $s \in C^2_{2\pi}$ gives rise to a reasonable BSC in the technical sense. In fact, the above model does not exclude that the performance of the ship may not show a desired behavior with a given BSC. We will come back to this point later.

If we want to optimize the efficiency of the VSP as a function of the BSC, our objective function obviously takes the form

$$E : C^2_{2\pi} \to \mathbb{R}, \qquad E : s \mapsto E(s) := \eta_0(s), \tag{3.2}$$

where η_0 denotes the *efficiency*. In order to define $E(s)$, we need to describe the flow that is induced by the rotation of the VSP.

3.1.2 The Navier-Stokes Equations

The Navier-Stokes equations are the core equations of continuum, gas and fluid dynamics. Hence, they also form an appropriate model here. Since the medium of the flow is water, we use the *incompressible* Navier-Stokes equations which read as follows

$$\begin{aligned} \frac{\partial}{\partial t}\vec{u} + \vec{u} \cdot \nabla\vec{u} - \nu\Delta\vec{u} + \frac{1}{\rho}\nabla p = \vec{f}, \\ \operatorname{div} \vec{u} = \nabla \cdot \vec{u} = 0, \end{aligned} \tag{3.3}$$

where $\vec{u}$ denotes the unknown velocity field, p the unknown pressure, $\vec{f}$ a given exterior force, ρ is the given (constant) density and ν is the kinematic viscosity. All these functions depend on time $t \in [0, T]$ and space $x \in \Omega \subset \mathbb{R}^3$, where the final time T corresponds to one full rotation of the propeller and Ω is the domain of the flow. In addition, appropriate initial and boundary conditions have to be imposed.

The influence of the BSC s to the flow is via the boundary conditions, the exterior force $\vec{f}$ is determined e.g. by the water speed and other exterior flow conditions. Assuming that we have a solution of (3.3), we obtain properties

$$\vec{u}(s) : [0, T] \times \Omega \to \mathbb{R}^3, \qquad p(s) : [0, T] \times \Omega \to \mathbb{R}$$

depending on the particular BSC $s \in C^2_{2\pi}$.

3.1.3 Efficiency

Now assuming that we have the quantities from the CFD calculations at hand, we are able to calculate the relevant technical quantity, i.e., the *efficiency*.

Figure 3 shows some of the relevant geometry. We start with the force initiated by the pressure

$$\vec{F}_p(t) := \int_{A(t)} p(t,\vec{x})\vec{n}(t,\vec{x})\, dA, \tag{3.4}$$

where A denotes the surface of the blades at time t and $\vec{n}(t,\vec{x})$ is the outward normal vector on the blade surface. Next the force created by the shear stress $\vec{\tau} := \mu\frac{\partial u}{\partial \vec{n}}$ is calculated

$$\vec{F}_r(t) := \int_{A(t)} \vec{\tau}(t,\vec{x})\, dA. \tag{3.5}$$

The part of the forces in driving direction $\vec{d}$ of the ship is the *thrust force*

$$T(t) := \left(\vec{F}_p(t) + \vec{F}_r(t)\right) \cdot \vec{d}. \tag{3.6}$$

The *torque* M consists of two parts: M_G is the torque related to the global propeller rotation

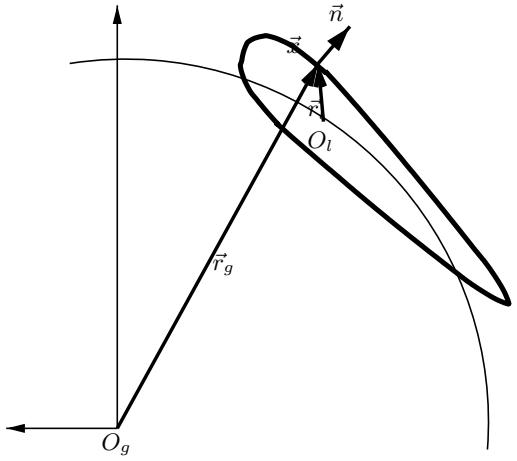

Figure 3: coordinates and geometry of the Voith-Schneider® Propeller.

$$M_G(t) := \left(\int_{A(t)} p(t,\vec{x})\vec{n}(t,\vec{x}) \times \vec{r}_g + \vec{F}_r(t) \times \vec{r}_g\, dA\right) \cdot \vec{z}_D, \tag{3.7}$$

where $\vec{r}_g$ denotes the *lever* of the global rotation, i.e., the vector from the center of the global rotation O_g to $\vec{x}$ and $\vec{z}_D$ denotes the global axis of rotation. M_B is the torque necessary to steer the blades accordingly to the BSC s:

$$M_B(t) := \left(\int_{A(t)} p(t,\vec{x})\vec{n}(t,\vec{x}) \times \vec{r}_l + \vec{F}_r(t) \times \vec{r}_l\, dA\right) \cdot \vec{z}_D, \tag{3.8}$$

where $\vec{r}_l$ denotes the *lever* of the blade rotation. Together we have

$$M(t) := M_G(t) + M_B(t)\frac{1}{2\pi n}\frac{\partial s(t)}{\partial t}. \tag{3.9}$$

By taking the integrals with respect to time t over one complete propeller turn we obtain

$$T := \int_0^{2\pi} T(t)\, dt \quad \text{and} \tag{3.10}$$

$$M := \int_0^{2\pi} M(t)\, dt. \tag{3.11}$$

The propulsion efficiency of the free running propeller is defined as the ratio of the propulsion power (Tv_a) and the delivered power ($M2\pi n$), where v_a is the inflow speed to the propeller:

$$\eta_0 := \frac{Tv_a}{M2\pi n}. \tag{3.12}$$

Next we calculate the dimensionless quantities for thrust and torque. Those are

$$ks := \frac{T}{\frac{\rho}{2}DL(D\pi n)^2} \quad \text{and} \tag{3.13}$$

$$kd := \frac{2M}{\frac{\rho}{2}D^2L(D\pi n)^2}, \tag{3.14}$$

where ρ is the density of water, D is the diameter of the propeller, L denotes the length of the blades and n is the number of revolutions of the propeller. Hence we have

$$\eta_0 = \frac{ks}{kd}\lambda. \tag{3.15}$$

Finally, the optimization problem under consideration is the following

$$\eta_0(s) \to \max_{s \in C^2_{2\pi}}! \tag{3.16}$$

This formulation corresponds to an *unrestricted optimization problem* with a *nonlinear* objective function. Obviously, we could also formulate the problem as a PDE constrained optimization problem. Methods, advantages and difficulties of this approach are detailed in [1]. However, as far as we could see, this does not offer a better choice for the numerical treatment for the considered problem. Especially without giving up the black box approach and rewriting the CFD software, which is almost impossible for comercial software.

3.2 Discretization

Now we introduce the discretization that we use in order to determine a numerical approximation to the objective function $\eta_0(s)$.

3.2.1 Parametrization of the BSC

The function space $C^2_{2\pi}$ is infinite-dimensional. Thus, we need a finite-dimensional approximation which is suitable for the numerical treatment. This means, we need a *parametrization* $p : \mathbb{R}^n \to C^2_{2\pi}$, $\mathbb{R}^n \ni X \mapsto s(X)$, i.e., a BSC s is represented by n degrees of freedom.

The next step is to introduce a particular approximation using n degrees of freedom. Due to the smoothness properties of a BSC, it is meaningful to do this e.g. by periodic B-splines. Let us denote by $\Xi_n := \{\xi_1 \leq \cdots \leq \xi_n\}$, $\xi_i \in [0, 2\pi]$ a set of knots and by

$$\mathcal{S}(n; \Xi_n)$$

the space of quadratic periodic splines corresponding to the knots Ξ_n. The particular choice of Ξ_n, i.e., the precise value for n and the choice and location of free or fixed knots depends of course on the properties of the underlying problem. This will be described in Section 5.1 below.

Hence we replace (3.16) by the *discrete* optimization problem

$$\eta_0(s_n) \to \max_{s_n \in \mathcal{S}(n;\Xi_n)} ! \tag{3.17}$$

This means that we are faced with an m-dimensional optimization problem, where

$$m := n - 1 \text{ and} \tag{3.18}$$

$$m := 2n - 3 \tag{3.19}$$

for fixed knot splines and free knot splines respectively. Of course, the efficiency of a numerical method crucially depends on the dimension m. Thus, it will be an important task to choose n in an appropriate way. This will be described later. The character of the original optimization problem (nonlinear unrestricted problem) is of course not changed.

3.2.2 Computational Fluid Dynamics

In principle, we could use any discretization for the instationary, incompressible Navier-Stokes equations. This is by now part of a well-established field called *Computational Fluid Dynamics (CFD)*. Note, however, that we have an instationary, nonlinear 3d problem at hand which cannot just be solved by 'numerical brute force'. Moreover, the numerical solution of the Navier-Stokes equations coincide with the evaluation of the objective function of the optimization problem for *one* particular BSC. Thinking of an iterative optimization method, one has to expect a large number of these evaluations and hence efficiency is a major target here.

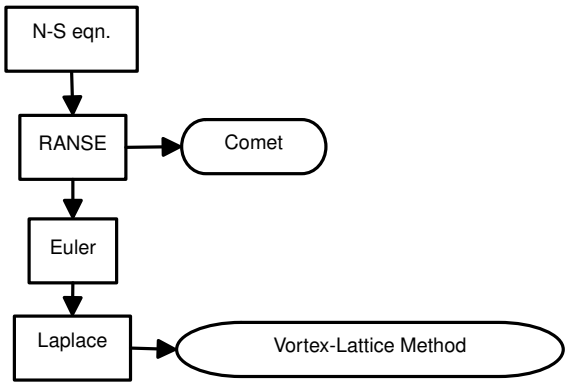

Figure 4: Hierarchy of methods.

A common way of solving Eqn.(3.3) also for complex geometries are Finite Element (FE) or Finite Volume (FV) methods. For the VSP, Voith has adapted a commercial FV-code (Comet®) and this is intensely used for research and development. However, the simulation of the full model still takes in the order of days, so that in the current state we do not yet use a FV scheme for the Navier-Stokes equations for the optimization. By introducing simplifications as shown in Figure 4 the fluid flow can be described by the Euler-equation or for *potential flow* by the Laplace-equation. For the optimization we use a fast Vortex-Lattice method, which is

described below. The FV methods and experiments are used later for validation. In Section 4.4 this strategy is explained in detail.

The Vortex-Lattice method. The theory of potential flows and the vortex-lattice method is outlined in detail in [9]. The adaption to the VSP and the code we use is explained in [8]. The theory of potential flow is based on the assumptions that the fluid is incompressible, inviscid and irrotational. We already limited our consideration to the incompressible version of the Navier-Stokes equations. Now we introduce the additional assumption, that for flows at high Reynolds numbers $\mathrm{Re} \sim \frac{1}{\nu}$ the effects of viscosity can be neglected outside thin boundary layers around the blades and their wakes. For the given problem the Reynolds number is in the order of 10^7. Given an irrotational inflow to the domain under consideration, the fluid remains irrotational. Such a flow $\vec{u}$ can be described as the gradient of a potential function $\Phi : \mathbb{R}^3 \to \mathbb{R}$. Using this, the continuity equation becomes the Laplace equation:

$$\Delta\Phi = \nabla \cdot \vec{u} = 0. \tag{3.20}$$

Since this is a linear equation the solution can be obtained by linear combination of solutions, i.e., the principle of superposition. Here we use vortices on the blade surface (bound vortices) and the wake (free vortices). Several boundary conditions have to be satisfied. Those are the time dependend kinematic boundary condition

$$\vec{u}(x) \cdot \vec{n}(x) = \vec{v}_a(x) \cdot \vec{n}(x), \quad x \in \partial B, \tag{3.21}$$

and

$$\lim_{r(x)\to\infty} (\nabla\Phi(x)) = 0, \tag{3.22}$$

where $r(x)$ denotes the distance of x to the blade and wake. The viscous effects near the blades and wakes are introduced by the empirical Kutta condition, which states that there is no flow around the trailing edge of the blades, i.e., the flow is parallel to the trailing edge. Since the wake vortices must be free of load, their position and movement is derternined by the local velocity. The strength of the wake vortices is determined by Thomson's theorem [9]. The Laplace equation does not contain time dependency. This is introduced through the boundary conditions. Once the velocity is calculated, the pressure p and the loads can be calculated by using the Bernoulli-equation and the Kutta-Joukovski theorem [9].

For numerical calculations the problem is discretized. Vortices are placed on a lattice on the blade surface. Figure 5 shows such a vortex scheme of one blade and its wake after two propeller turns. The equations described above have to be satisfied on a so-called collocation point in each element of the fixed vortex lattice. This gives linear equations to be solved

$$A\Gamma = b, \tag{3.23}$$

where A denotes a matrix according to the geometry of the blade, Γ is the strength of the bound vortices and b is the right-hand side given through the influence of the free vortices, the inflow v_a and the boundary conditions. Since the blade geometry does not change, the matrix A is inverted and solving the equations reduces to a matrix-vector-multiplication in subsequent timesteps. However, the expensive part is to compute the free vortex wake and its influence to the right-hand side b.

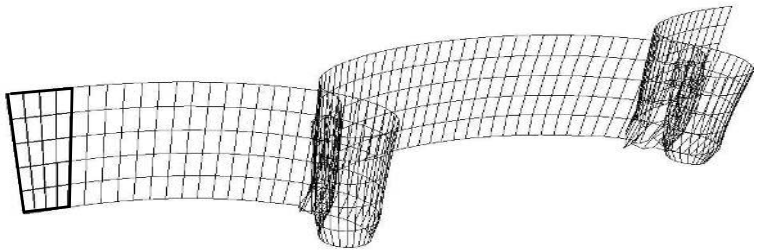

Figure 5: Vortex lines of the blade and wake.

4 Numerical Optimization

We have now described the discretization that enables us to perform an approximate evaluation of the objective function. Next we describe the choice of the optimization method which is based upon this evaluation. Of course, the decision which method is appropriate heavily depends on the properties of the objective function, which we describe first.

4.1 Properties of the Objective Function

In Section 3, we have described the mathematical modelling yielding the objective function under consideration. It is clear from the description there that an analytical investigation of the properties of the objective function is a delicate task. Moreover, the optimization algorithm uses the numerical approximation of the objective function. Hence, we focus on the mathematical properties of the numerical approximation only. Moreover, since we use the simulation scheme merely as a *black box* to compute an approximation of the objective function $\eta_0(s)$ we cannot distinguish between analytical properties of the target function and effects of the numerical simulation. Of course, we have to be aware of the fact, that deficiencies that might occur (e.g., a lack of smoothness of the objective function) might result from numerical effects and are not caused by analytical properties. We will come back to this point later.

4.1.1 Regularity

Several numerical schemes for optimization problems use information on the gradient of the objective function, either by numerically solving the Euler-Lagrange equations or within a method of steepest descent, gradient or conjugate gradient type. This of course requires that the objective function is at least locally continuously differentiable with respect to the parameters. In our case this would mean that η_0 is differentiable with respect to the BSC s in the continuous case and with respect to the parameters $X \in \mathbb{R}^n$ in the discrete case.

Even though one might expect that $\eta_0(s)$ is smooth, our numerical experiments show, that the function seems to be disturbed by noise[1]. To be precise, we chose a discretization of the BSC using 9 equidistant knots for the spline approximation. We fixed 8 of the corresponding values according to a known BSC which is used by Voith. The remaining parameter was varied. Figure 6 shows the resulting numerical approximation for the efficiency. It seems as if there is a global smooth function in the background which is polluted by 'numerical noise'.

[1]Note that η_0 is of course evaluated in a deterministic manner and noise is used for *non-stochastic* errors like truncation errors in this context as proposed in [10].

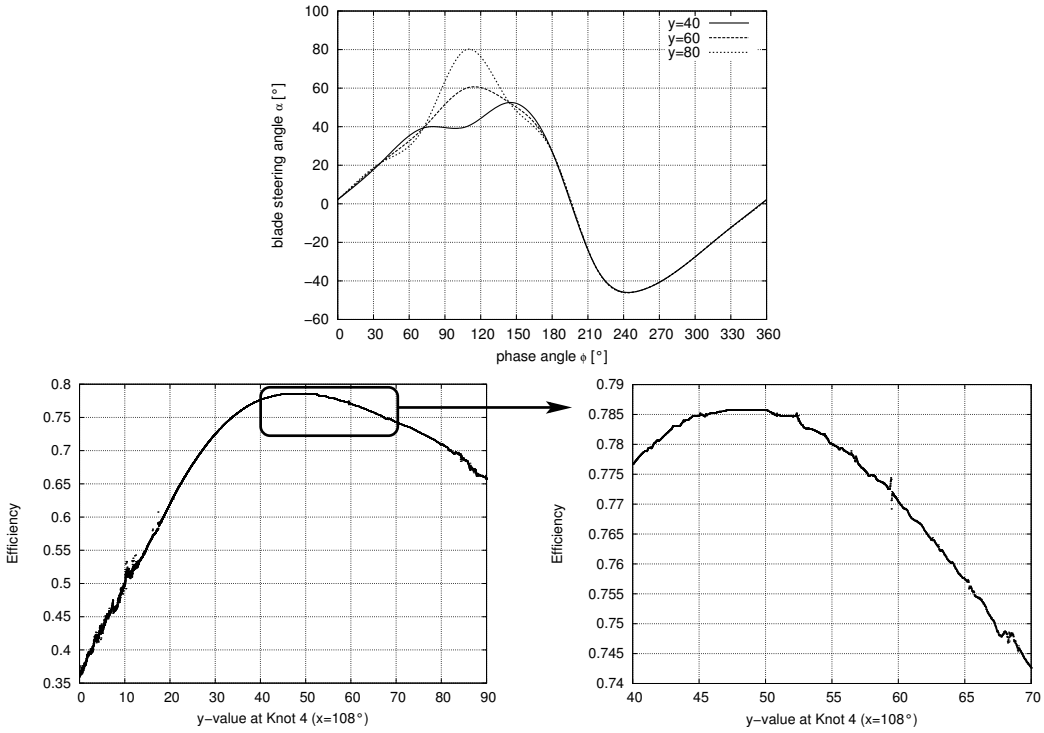

Figure 6: Efficiency in dependence of the value at the third spline knot. The third picture is a zoom-in of the second one.

To demonstrate the effect of such noise on numerically calculated gradients, we considered a BSC $s \in C^2_{2\pi}$ represented by splines as above but with 11 fixed knots and with variable parameters $X \in \mathbb{R}^{10}$. We calculated the difference quotients $\frac{\eta_0(s(X))-\eta_0(s(X)+he_i)}{h}$ for different stepsizes $h \in \mathbb{R}$, where $e_i \in \mathbb{R}^{10}$, $i = 1, \ldots, 10$ denotes the i-th unit vector. In Figure 7 each line corresponds to one $e_i \in \mathbb{R}^{10}$. As one can see, most of the lines are straight with only a small slope, i.e., the dependency on h is small. However, in some cases, e.g. $i = 3, 4, 8$, the difference quotients vary dramatically with respect to h and in some cases even change their sign. A gradient based on this difference quotients would change its direction for different values of h and therefore would be rather unreliable as descent direction for example. This corresponds to the well-known result, that numerical differentiation by difference quotients requires function evaluations with high precision, which we cannot guarantee here. Thus, numerical schemes using the gradient have been avoided.

4.1.2 Convexity and Existence of a Global Maximum

For existence and uniqueness of local and global maxima one usually requires the objective function to be convex. It is clear that convexity is a property which is impossible to check numerically. Referring again to Figure 6, it seems that there is a smooth and convex function in the background which is polluted by some kind of noise. However, a closer look shows that we cannot hope for a unique global maximum but have to deal with several local maxima. Thus it is not sufficient only to use a numerical scheme that is able to determine an approximation of a local maximum (which otherwise, by convexity, could be guaranteed to be also a global one).

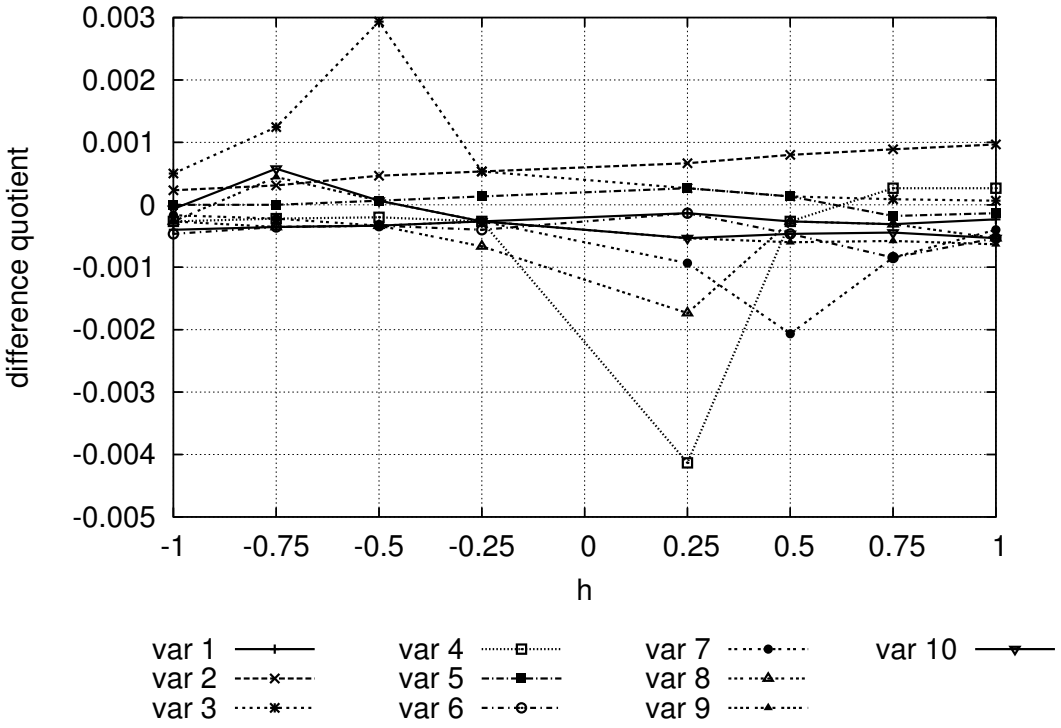

Figure 7: Difference quotients for a BSC with 11 fixed spline knots.

4.2 Local Optimization

As already said, since our objective function seems to be lacking smoothness, standard gradient-based methods cannot be used. One could of course try to approximate derivatives either by finite differences or by automatic differentiation. However, since we are facing perturbation by some kind of noise, this could result in heavy oscillations and would make numerical results unreliable.

For these reasons, we chose known direct search methods, since those methods only depend on values of the objective function itself. The major drawback of these methods is their sometimes slow convergence and their failure in finding the minimum in some examples. However, these methods have been shown to be useful in general and rather robust when confronted with noisy data. For a detailed review of this class of methods we refer to [10, 15].

The methods we used are the search method of *Hooke and Jeeves* [7] and the *simplex method* by Nelder and Mead [12]. Both methods will be briefly explained in the following.

4.2.1 The Method of Hooke and Jeeves

This method consists of two steps, namely the *pattern move* as an outer iteration and an *exploration step* as inner loop. Starting with a given point $X^{(0)} \in \mathbb{R}^n$ it creates a sequence of points $(X^{(i)})_i$ such that

$$f(X^{(0)}) > f(X^{(1)}) > f(X^{(2)}) > ...$$

Note that it may happen that the iteration stops with some $X^{(i)}$ without improving further.

The pattern move checks for all coordinate directions in order to search for a direction of possible descent. For a given coordinate direction, the exploration move checks if the objective functional can be minimized in that direction and looks for a maximal step size in that direction. Whenever the exploration move cannot find a better point X^*, the step size is halved and the process is iterated until the step size reaches a prescribed minimal value $h_{\min}$.

The algorithm can be described as follows:
Let $i = 0$ and $e_i \in \mathbb{R}^n$ be the i-th unit vector. Start with an initial point $X^{(0)} \in \mathbb{R}^n$ and an initial step size $h \in \mathbb{R}^n$.

Pattern move.

1. If $\max_{j\in\{1,\cdots,n\}} h_j < h_{\min}$ stop the algorithm, $f(X^{(i)})$ is the approximation of the minimum.
2. Set $X^* = \text{explore}(X)$.
3. If $f(X^*) \not< f(X^{(i)})$ replace h by $\frac{h}{2}$ and continue with step 1.
4. Set $X^{**} = \text{explore}(2X^* - X^{(i)})$.
5. If $f(X^{**}) < f(X^*)$ set $X^{(i+1)} = X^{**}$, otherwise set $X^{(i+1)} = X^*$.
6. Replace i by $i + 1$ and continue with step 1.

Exploration move.

1. Evaluate $f(X + h \cdot e_i)$.
2. If $f(X + h \cdot e_i) < f(X)$ replace X by $X + h \cdot e_i$ and continue with step 5.
3. Evaluate $f(X - h \cdot e_i)$.
4. If $f(X - h \cdot e_i) < f(X)$ replace X by $X - h \cdot e_i$.
5. If $i < n$ set $i = i + 1$ and continue with step 1.
6. Return X.

4.2.2 The Simplex Algorithm of Nelder and Mead

This algorithm starts with $n + 1$ points $(X_i)_{i=1,\ldots,n+1}$, where $X_i \in \mathbb{R}^n$. These points form a simplex and must not lie in a hyper plane. We define the best and the worst point as follows:

$$m := \underset{i=1,\ldots,n+1}{\text{Argmin}}\, f(X_i) \quad \text{and} \quad M := \underset{i=1,\ldots,n+1}{\text{Argmax}}\, f(X_i). \tag{4.24}$$

These are calculated in each iteration. Then the algorithm replaces iteratively one or all but one points in the simplex. The new points are chosen by geometric operations based on the given points in the simplex. To this end, we define

$$\begin{aligned}
&\text{Centroid}: && \overline{X} &&:= \frac{1}{n} \sum_{\substack{1\le i\le n+1 \\ i\neq max}} X_i \\
&\text{Reflection}: && X^R &&:= (1+\alpha)\overline{X} - \alpha X_M \\
&\text{Expansion}: && X^E &&:= \gamma X^R + (1-\gamma)\overline{X} \\
&\text{Contraction}: && X^C &&:= \begin{cases} \beta X_M \pm (1-\beta)\overline{X}, & \text{if } f(X_m) < f(X^R), \\ \beta X^R \pm (1-\beta)\overline{X}, & \text{otherwise.} \end{cases}
\end{aligned} \tag{4.25}$$

The decision rule which points are to be replaced is shown in Figure 8.

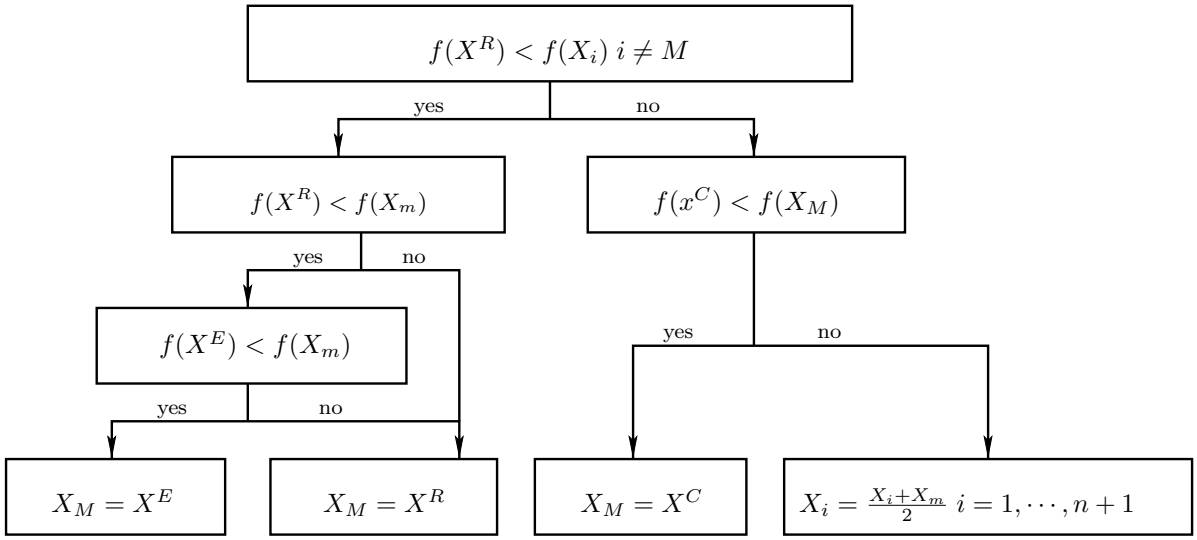

Figure 8: Decision rule for the simplex method.

4.3 Global Optimization

As we have seen in Section 3 above, we have to expect several local maxima. Since we are ultimately interested in *the best* BSC, the use of one of the above described direct search methods alone is not sufficient. We need a global optimization strategy in addition.

An overview of different concepts of global optimization is given in [13] and [4]. However, for the given problem with properties as determined in Section 4.1, like unknown, possibly infinite, number of local maxima and numerical noise that results in peeks as can be seen in Figure 6, there is no method known that guarantees to find the global maximum in a finite amount of time. Still we expect a great improvement with global optimization methods compared to local methods introduced above.

Several classes of methods have been considered. *Interval arithmetic* methods depend on Lipschitz-continuity, which cannot be guaranteed in our case. Finding an adequate sample for Monte-Carlo based methods is difficult, since it is not known a priori whether a BSC s_n is feasible in the sense of some simple tests like positive thrust, $T(s_n) > 0$. In fact choosing random parameters Ξ_n for the BSC s_n led to infeasible results of the CFD calculations in the majority of cases.

Therefore, we chose an algorithm called *tunneling method* that was introduced by Levy and Montalvo in [11] and that is also mentioned in [13]. In the original form, the method in [11] also involves gradient based local optimization methods, e.g. conjugate gradient or Newton's method. Hence, we had to do some modifications by replacing the gradient-based methods by the above mentioned Hooke and Jeeves algorithm. To explain the procedure we consider the optimization problem

$$f(x) \rightarrow \min_{x \in \mathbb{R}^n} ! \tag{4.26}$$

The tunnelling method starts with a local search (using any local method) on $f(x)$ for any given starting point $x_0 \in \mathbb{R}^n$. The local search stops when a minimum x_1 is found. Then, the *tunnelling*

phase follows. Therefore, x_1 is inserted as a pole to a new objective function T_1 defined as follows

$$T_1(x) := \frac{f(x) - f_1^*}{((x - x_1)^T(x - x_1))^{\lambda_1(x)}}, \tag{4.27}$$

where f_1^* denotes the local minimum and $\lambda_1 : \mathbb{R}^n \to \mathbb{R}$ is a function that controls the size (i.e., the radius) of the pole at x_1. Then a local search is performed using T_1 instead of f and the start point is moved away from the pole by $\varepsilon \in \mathbb{R}^n$, $|\varepsilon| \ll 1$. This procedure is iterated. The corresponding function reads

$$T_k(x) := \frac{f(x) - f_k^*}{\prod_{j=1}^{k}((x - x_j)^T(x - x_j))^{\lambda_j(x)}}, \tag{4.28}$$

where f_k^* is now the best local minimum found so far. Obviously, we have $T_k(x) \geq 0$ if and only if $f(x) \geq f_k^*$.

The tunnelling phase ends, when some $x \in \mathbb{R}^n$ with $T(x) < 0$ is reached. This means that a better value than f_k^* is obtained. This point is then the start point for a new local search on Problem (4.26). If no such x can be found by the tunnel-search then either the radius of the pole is to small or a new local minimum not better than x^* has been found. Respectively the radius is increased or a new pole is inserted. The decision depends on the distance between the actual position x and the position of the last inserted pole. Both increasing the radius and inserting new poles is only done a limited number of times and then the whole algorithm is restarted with a new randomly chosen start point.

Figure 9 shows an example function

$$f(x_1, x_2) = -(-\cos(\frac{x_1}{5})\cos(\frac{x_2}{5}) + 10^{-6}(x_1^4 + x_2^4) + 10^{-4}(x_1^2 + x_2^2) - \frac{3}{10}\cos(\frac{x_1 x_2}{10}), \tag{4.29}$$

that has several local maximal points and which we have additionaly disturbed by high-frequent noise. Since our algorithm does minimization, we used $-f$. The global maximum of this function is the value 1.3 at $(0, 0)$. As starting point we used $(-30, 20)$ and obtained the (local) maximal

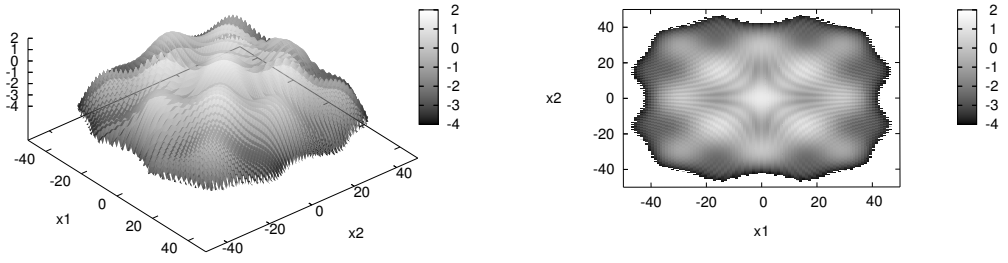

Figure 9: Test-function for global optimization as 3d and 2d plot. Note the high-frequent noise caused by the term $-\frac{3}{10}\cos(\frac{x_1 x_2}{10})$ and not due to pure graphical resolution.

points shown in Table 4.3. After 4 iterations, we habe reached the global maximum. Other test-functions were taken from [11] and our modified algorithm was successfull also for higher dimensional problems. Even the number of function evaluations was in the same order as with gradient based optimization algorithms in [11].

(x_1, x_2)	-f
$(-19.357, 19.357)$	-0.4919
$(-17.673, 17.673)$	-0.8908
$(-15.837, 15.837)$	-1.1230
$(5.464 \cdot 10^{-8}, 5.606 \cdot 10^{-8})$	-1.3000

Table 1: Results of Tunneling-Optimization on the example function (Eqn. (4.29)).

4.4 Strategy

As explained in Section 3.2.2, we depend on a fast low fidelity code for the CFD calculations. This means we encounter a loss of accurary, which we have to compensate. This is done by using the high-fidelity code and physical experiments for validation of the results of new BSCs which we found as result during the optimization process, as shown in Figure 10. If the improvement is

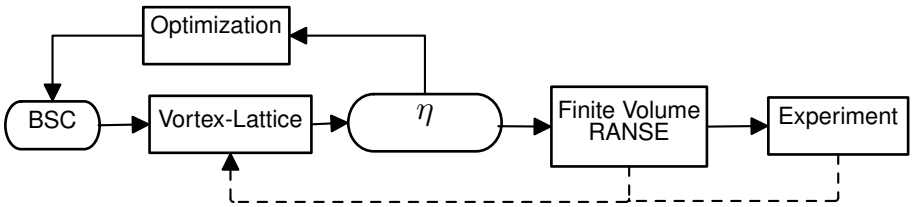

Figure 10: Strategy

validated, everything is fine. However, it is possible that an optimized BSC in the high-fidelity calculations performs worse than the BSC we started with. In this case we have to improve the low-fidelity code before starting the optimization again. Up to now this improvement is done by tuning parameter settings in the vortex-lattice code and by introducing some empirical correction terms by hand. In the future we intend to use approximation models to automatically improve the low-fidelity results.

5 Results

5.1 Parametrization of the BSC

The parametrization of the BSC by splines was already described in Section 3. Now we deal with the problem of determining the number of knots n necessary to guarantee a specific accuracy and whether to use free or fixed knot splines. On one hand the space $\mathcal{S}(n; \Xi_n)$ of generated BSC depends on n in a crucial way. Therefore, n has to be sufficiently large to ensure that the best BSC $s^* \in C^2_{2\pi}$ also lies in $\mathcal{S}(n; \Xi_n)$, or can at least be approximated up to a certain accuracy by some $s_n \in \mathcal{S}(n; \Xi_n)$. On the other hand, n corresponds to the dimension m, as defined by Eqn. 3.18, of the optimization problem that has to be solved. Thus, it also influences the number of necessary function evaluations during the optimization.

As a first choice for n, we investigate a spline approximation of an actually used BSC to a prescribed tolerance. Before we show results of such approximation experiments, we have to state more precisely what we choose as 'sufficient' accuracy for approximation. As a distance

measure of two curves $f, g \in C^2_{2\pi}$ we use the discrete norm

$$d(f,g) := \|f-g\|_{h,2}, \tag{5.30}$$

$$\|z\|^2_{h,2} := \sum_{i=0}^{\frac{2\pi-1}{h}} |z(ih)|^2. \tag{5.31}$$

Table 2 shows the results of approximations of a given BSC with fixed and free knot splines, where n denotes the number of knots and m the number of parameters. From a technical point of view of the underlying problem we take values below 0.01 as sufficient. Therefore we started our optimization with 10 fixed knot splines and 6 free knot splines, which amounts to 9 parameters in each case. We will later compare the results of the optimization with different number of parameter.

n	m_{fixed}	$d(f,g)$ fixed	m_{free}	$d(f,g)$ free
4	3	0.42863	5	0.30636
5	4	0.53962	7	0.07381
6	5	0.02322	9	0.00772
7	6	0.11344	11	0.00666
8	7	0.02923	13	0.00527
9	8	0.01148		
10	9	0.00629		
11	10	0.00199		

Table 2: Approximation of a given BSC by splines.

5.2 Local Optimization

A large number of optimization runs (where we call one complete application of the optimization method a 'run') has been undertaken with the described methods for different constellations of the VSP or different outside flow conditions. In almost all cases an improvement compared to the initial BSC was achieved. For the sake of simplicity we considered the simplified case with only one blade. This is mainly due to computation times. We show some of those examples, all with the advance coefficient chosen as $\lambda = 0.6336$, which corresponds to a normal driving situation. Table 3 shows the initial BSC (1) and results of two optimization runs with different number of knots. In both cases an improvement of 4.6% and 4.8% was achieved. Fig. 11 shows the corresponding three BSC. Table 4 contains the corresponding verifying calculations with the

BSC	no. of knots	free/fixed	ks	kd	η_0	improvement of η_0
1	-	-	0.1262	0.1050	0.7610	-
b1	10	fixed	0.0809	0.0644	0.7957	4.6%
b2	21	fixed	0.0808	0.0642	0.7976	4.8%

Table 3: Results of local optimization.

comercial CFD program Comet, which we cannot yet use for optimization directly due to CPU times as explained above. Of course the absolute values differ, but trends are validated.

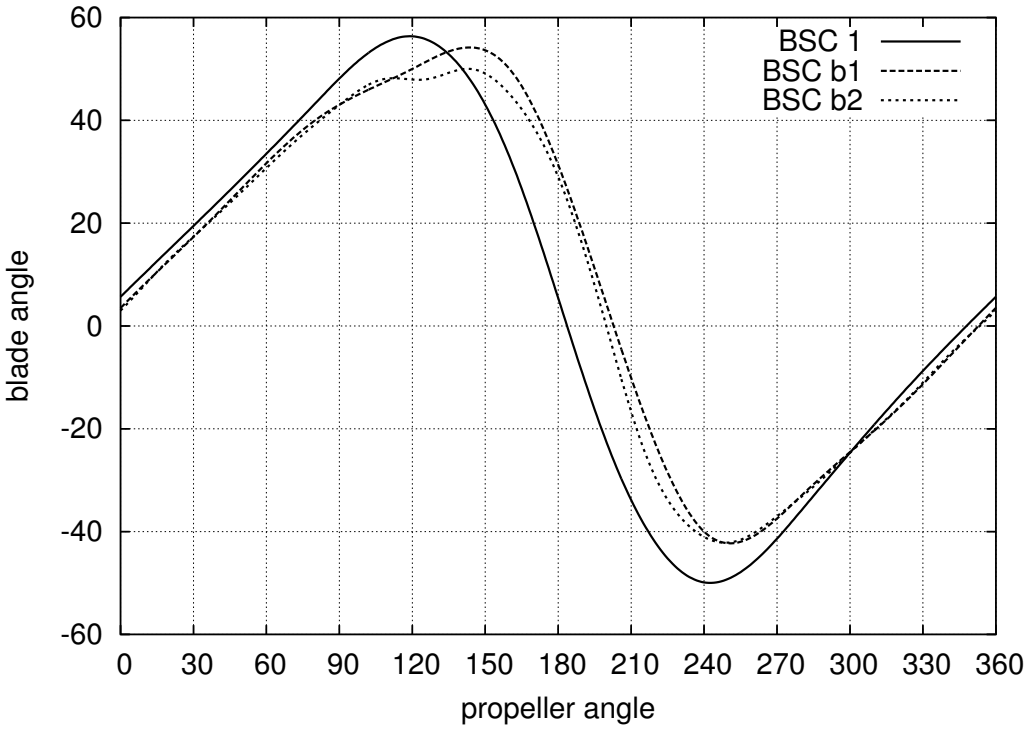

Figure 11: Original and optimized BSC.

BSC	no. of knots	free/fixed	ks	kd	η_0	improvement of η_0
1	-	-	0.1250	0.1060	0.747	-
b1	10	fixed	0.0789	0.0650	0.769	2.9%
b2	21	fixed	0.0790	0.0648	0.772	3.3%

Table 4: Verifying calculations of local optimization results using Comet.

Another observation from Table 3 is that different results were achieved for different number of knots that were used to discretize the BSC. We did another study to check the influence of the number of knots and parameters. Table 5 shows the results. The best result ($\eta_0 = 0.797$) was reached in the cases of splines with 9 fixed knots and 9 free knots. One can expect, that higher number of variables lead to better results. In the case of fixed (equidistant) knots it is still possible that $s \in \mathcal{S}(n;\Xi_n)$ and $s \notin \mathcal{S}(n';\Xi'_n)$ for some $n < n'$. However, for the free knot case we have $\mathcal{S}(n;\Xi_n) \subset \mathcal{S}(n';\Xi'_n)$ for $n < n'$. Clearly the results in Table 5 do not verify these expectations, but since we were expecting local maxima this is not to surprising. Using different start BSC also led to different results, which again is typical for local maxima.

The first try to overcome this problem is to approximate the best BSC with different number of knots and to use these approximations as new start BSC. The results are shown in the last column of Table 5. In the cases of $7, 8$ and 10 fixed knot splines the approximations failed and the new start BSC performed worse than $\eta_0 = 0.797$. In all other cases but one, the previous best curve was reached after the second optimization run. In the case of 19 fixed knots an even better BSC was found. For a more systematic way to deal with the problem of local maximizers

no. of knots n	no. of variables m	free/fixed	η_0	η_0 restarted
7	6	fixed	0.789	0.791
8	7	fixed	0.786	0.786
9	8	fixed	0.797	0.797
10	9	fixed	0.794	0.793
11	10	fixed	0.790	0.797
13	12	fixed	0.789	0.797
16	15	fixed	0.794	0.797
19	18	fixed	0.784	0.799
6	9	free	0.789	0.797
7	11	free	0.795	0.797
8	13	free	0.791	0.797
9	15	free	0.797	0.797
10	17	free	0.795	0.797
11	19	free	0.795	0.797

Table 5: Local optimization using different number of variable parameters.

we consider global optimization techniques. Results are presented in the following section.

5.3 Global Optimization

We picked two cases from the study above, one with free knot splines and the other one with fixed knot splines, each with 11 knots. The obtained results are shown in Tab. 6. In both cases

no of knots n	no of variables m	fixed / free	Local Maximum	η_0
11	10	fixed	1	0.792
11	10		2	0.794
11	10		3	0.797
11	19	free	1	0.794
11	19		2	0.798
11	19		3	0.799

Table 6: Results of tunneling optimization.

three local maximal BSCs were found. In the case of 11 fixed knots the result of $\eta_0 = 0.797$ is the same as result from local optimization with restart. In the other case of 11 free knots the final result is better than the result we obtained with local optimization of 11 free knots. The tunneling result equals the best results of all cases in Tab. 5, namely $\eta_0 = 0.799$ with 19 fixed knots.

For the problem under consideration the tunneling algorithm is able to overcome several local maximum points. However, for each local maximum a tunneling phase and a local search has to be performed. During the tunneling phase, typically several local searches with the objective function $T(x)$ are necessary to adjust the parameters of the poles, e.g. the radius. Thus, the amount of necessary function evaluations increases dramatically compared to a simple local search.

6 Conclusions and Outlook

We have shown that the use of numerical techniques gives rise to a significant improvement in the efficiency of the Voith-Schneider® Propeller. However, it should be noted that the numerical results have been achieved by a simplified model (Vortex Lattice) due to the complexity of the objective function. Even though in our case the results could also be verified within the more realistic framework of a Navier-Stokes solver, the CPU time gives some limitation. For our particular case, this could be overcome by using high performance computers. This will be a subject of future research. Note however, that this does not circumvent the general problem.

In fact, the BSC is only one of several parameters that determine the behavior of the VSP. It is an obvious question to include also other parameters into the optimization. This would of course increase the dimension of the optimization problem which in turns would not permit to use the Navier-Stokes solver for the inner loop. Hence, we need to find a good compromise between accuracy and efficiency of the flow solver. At this point, several ideas come to mind. Among those are model reduction techniques and so called multifidelity optimization. Another promising concept for nonsmooth optimization are bundle methods. Those will be a subject of further research in the near future.

Our success also shows that this environment can be used to optimize any technical system that is characterized by certain input parameters and for which a (numerical) simulation is available which gives an approximation of the target functional. We will provide a general framework for this task.

In summary, this paper shows that numerical methods from optimization and CFD do have an enormous potential to increase the quality even of high-standard products.

References

[1] L.T. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders Large-Scale PDE-Constrained Optimization, Springer-Verlag, 2003

[2] R.P. Brent, R.P., Algorithms for Minimization without Derivatives, Prentice-Hall, Englewood-Cliffs, N.J. (1973).

[3] R. Dautray and J.L. Lions, Mathematical Analysis and Numerical Methods for Science and Technology, Springer Verlag, 1993.

[4] L.C.W. Dixon and G.P. Szego (eds.), Towards global optimisation, Amsterdam: North-Holland, 1978.

[5] Q. Duan, S. Sorooshian, and V. Gupta, *Effective and efficient global optimization for conceptual rainfall-runoff models*, Water Resources Research **28**, no. 4 (1992), 1015-1031.

[6] V. Girault and P.-A. Raviart, Finite Element Methods for Navier-Stokes-Equations, Springer Verlag, 2nd edition, 1986.

[7] R. Hooke and T.A. Jeeves, *"Direct Search" Solution of Numerical and Statistical Problems*, J. ACM **8**, no. 2 (1961), 212–229.

[8] D. Jürgens, *Theoretische und experimentelle Untersuchungen instationärer Tragflügelumströmungen und Entwicklung eines Berechnungsverfahrens für Vertikalachsmotoren*, PhD. thesis, Rostock 1994.

[9] J. Katz, A. Plotkin, Low-Speed Aerodynamics, Cambridge University Press 2001

[10] T.G. Kolda, R.M. Lewis and V. Torczon, *Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods*, SIAM Review **45**, No. 3 (2003), 385–482.

[11] A.V. Levy and A. Montalvo, *The tunneling algorithm for the global minimization of functions*, SIAM J. Sci. Statist. Comput. **6** (1985), no. 1, 15–29.

[12] A. Nelder and R. Mead, *A simplex method for function minimization*, The Computer Journal **7** (1965), 308–313.

[13] G.L. Nemhauser, A.H.G. Rinnoy Kan, M.J. Todd, Optimization, Handbooks in Operations Research and Managment Science Vol. 1, North-Holland, 1989.

[14] J. Pinter, *Globally convergent methods for n-dimensional multiextremal optimization*, Optimization **17**, no. 2 (1986), 187–202.

[15] A. Quarteroni, R. Sacco, and F. Saleri, Numerical Mathematics, Texts in Applied Mathematics 37, Springer Verlag, 2000.