INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **53**:753–775
Published online 23 August 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1304

Comparison of derivative free Newton-based and evolutionary methods for shape optimization of flow problems

Zerrin Harth*,†, Hongtao Sun‡ and Michael Schäfer§

Department of Numerical Methods in Mechanical Engineering, Darmstadt University of Technology, Darmstadt, Germany

SUMMARY

The object of this study is to investigate two derivative free optimization techniques, i.e. Newton-based method and an evolutionary method for shape optimization of flow geometry problems. The approaches are compared quantitatively with respect to efficiency and quality by using the minimization of the pressure drop of a pipe conjunction which can be considered as a representative test case for a practical three-dimensional flow configuration. The comparison is performed by using CONDOR representing derivative free Newton-based techniques and SIMPLIFIED NSGA-II as the representative of evolutionary methods (EM).

For the shape variation the computational grid employed by the flow solver is deformed. To do this, the displacement fields are scaled by design variables and added to the initial grid configuration. The displacement vectors are calculated once before the optimization procedure by means of a free form deformation (FFD) technique.

The simulation tool employed is a parallel multi-grid flow solver, which uses a fully conservative finite-volume method for the solution of the incompressible Navier–Stokes equations on a non-staggered, cell-centred grid arrangement. For the coupling of pressure and velocity a pressure-correction approach of SIMPLE type is used. The possibility of parallel computing and a multi-grid technique allow for a high numerical efficiency. Copyright © 2006 John Wiley & Sons, Ltd.

Received 22 February 2006; Revised 7 June 2006; Accepted 12 June 2006

KEY WORDS: computational fluid dynamics; numerical shape optimization; derivative-free optimization; Newton based optimization; evolutionary methods; free form deformation

*Correspondence to: Zerrin Harth, Department of Numerical Methods in Mechanical Engineering, Darmstadt University of Technology, Petersenstr. 30, 64287 Darmstadt, Germany.
†E-mail: zerrin@fnb.tu-darmstadt.de, http://www.fnb.tu-darmstadt.de
‡E-mail: sun@fnb.tu-darmstdt.de
§E-mail: schaefer@fnb.tu-darmstadt.de

1. INTRODUCTION

Over the last quarter century, optimization has become an essential tool in computer aided engineering and has found applications in design, control, operations, and planning. In engineering practice, one is interested in, e.g. reducing the drag force of airplanes or vehicles, minimizing the dissipation in channels or hydraulic valves, etc. Such an investigation combines various distinct modules for the solution of the problem: geometric modelling, mesh generation, non-linear analysis of the fluid flow, sensitivity analysis, mathematical programming, and shape optimization. The first area of research on shape optimization was the field of structural analysis applications [1–3]. In recent years also in fluid mechanics a variety of applications have been considered [4–9].

Since there are a multitude of different optimization techniques the question arises, which suits best for a specific problem. In the past decade, the availability of parallel computers and faster computing hardware and the need to incorporate complex simulation models within optimization studies, have led a number of optimization researchers to reconsider classical direct search approaches. Using complex fluid flow simulation programs it is usually not possible to get easy access to the derivative information of the requested approximations. Therefore, in such situations it is advantageous to use an optimization technique that does not directly depend on the derivative information.

In the present paper we consider two such techniques which appear to be promising in this context: CONDOR [10] representing a derivative free Newton-based optimization method (DFNBM) and SIMPLIFIED NSGA-II [11] representing an evolutionary method (EM).

One of the first researches about optimization without any derivative information has began with Rosenbrock [12] in 1960s. The Rosenbrock method is a 0th-order search algorithm which means that it does not require any derivatives but only a number of evaluations of the objective function. (A rigorous proof of convergence of the direct search method of Rosenbrock is given in Reference [13].) The work of Rosenbrock has spawned considerable research on the analysis and code development for derivative free Newton-based optimization methods like Powell's UOBYQA [14], DFO [15], CONDOR [10] and lately NEWUOA [16]. It is very well-known that old Newton-based methods are faster, needs less number of function evaluations, but are usually giving solutions of poor quality.

Conn *et al.* [15] construct a multivariable DFO algorithm that uses a surrogate model for the objective function within a trust region method. In their work points are sampled to obtain a well-defined quadratic interpolation model and descent conditions from trust region methods enforce convergence properties. The CONDOR package employed is a variant of these kinds of methods.

Evolutionary methods, first proposed in Reference [17], are based on the analogy of improving a population of solutions through modifying their gene pool. Two forms of genetic modification, crossover or mutation, are applied where the elements of the optimization vector are represented by binary strings. Crossover deals with random swapping of vector elements (among parents with highest objective function values or other rankings of the population) or any linear combinations of two parents. Mutation deals with the addition of random values to elements of the vector. Genetic algorithms (GAs), as one computational model from the family of evolutionary methods, are widely used in process engineering and a number of codes are available.

The geometry variation is done by a specially adapted free form deformation (FFD) technique which already have been applied successfully for fluid flow applications [4, 6, 18, 19]. The numerical flow simulations are performed with the finite-volume flow solver FASTEST which, due to a multi-grid technique and the possibility for parallel computing, allows for a high numerical efficiency [20, 21].

For the comparison of the two optimization approaches, an integrated procedure is applied that modularly couples the optimization tool, the free form deformation technique, and the flow solver. As representative test case for a complex three-dimensional flow configuration, we investigate the optimization of the connection of two pipes with respect to the minimization of the pressure drop.

2. OPTIMIZATION

In general, a non-linear optimization problem with non-linear constraints can be stated as

$$\begin{aligned} &\min f(x_i) \\ &\text{w.r.t. } l_i \leqslant x_i \leqslant u_i, \quad i=1,\ldots,I \quad \text{box constraints} \\ &g_m(x_i) \geqslant 0, \quad m=1,\ldots,M \quad \text{inequality constraints} \\ &h_n(x_i)=0, \quad n=1,\ldots,N \quad \text{equality constraints} \end{aligned} \tag{1}$$

where f is the function to be optimized, x_i are the I design variables, and g_m and h_n are the M inequality and N equality constraints, respectively. The problems to be optimized in engineering fluid flow applications can be pressure drop, volume, lift, drag, heat exchange, etc. or any combination of these. The algorithms developed for the solution of such optimization problems as stated in (1) vary greatly. In the case of fluid dynamics based optimization problems when the fluid is modelled by the Navier–Stokes equations, it is usually nearly impossible to compute the exact derivatives of the finite dimensional approximations of these equations with respect to shape and mesh motion. Therefore, it is desirable to employ methods which do not use the derivative information of the objective function directly.

The optimization techniques which do not attempt to directly compute approximations to the derivative information are called derivative free optimization methods. It is difficult to state where the idea of derivative free optimization was first introduced, but in the survey of Wright [22] it is remarked that the direct search methods are first suggested in the 1950s. The first research was done by Spendley *et al.* [23] and afterwards various other methods were developed to handle similar classes of problems, e.g. References [10, 14–16]. Besides his many important suggestions, Powell [14, 24, 25] has also suggested the use of variational criteria, e.g. good information from an approximation can be inherited by its successors at subsequent iterations.

2.1. Derivative free Newton-based methods

Derivative free Newton-based methods (DFNBM) are based on trust region algorithms which are a natural evolution of line search algorithms. In trust region methods an approximate solution is constrained to lie within a region where an approximation model is sufficiently accurate. The trust region framework is usually used in the context where at least the gradient and sometimes the Hessian of the objective function can be evaluated or estimated accurately. If the current trust radius is binding, minimizing the approximation model function subject to this constraint may modify the direction as well as the length of the Newton step. The accuracy of the model is assessed by comparing the actual decrease in the objective function with the one predicted by the model, and the trust radius is increased or decreased accordingly.

Generally the trust region method can be stated as

1. Given a current iterate, build a good local approximation model (e.g. based on a second-order Taylor series approximation)
2. Choose a neighbourhood around the current iterate where the model is 'trusted' to be accurate.
3. Minimize the model in this neighbourhood.
4. Determine if the step is successful by evaluating the true function at the new point and comparing the true reduction in value of the objective with the reduction predicted by the model.
5. If the step is successful, accept the new point as the next iterate and proceed (possibly, increasing the size of the trust region if the success is really significant). If the step is not successful, reject the new point and reduce the size of the trust region.
6. Repeat until convergence.

An important improvement of Powell to derivative free optimization is the method that he has proposed for constrained optimization where the objective function and constraints are approximated by linear multivariate interpolation. He described an algorithm using a multivariate quadratic interpolation model of the objective function for unconstraint optimization problems [14, 24, 25].

For our comparative studies we use the derivative free Newton-based optimization algorithm CONDOR, developed by Frank Vanden Berghen [10]. It is based on the trust region framework together with a local quadratic approximation model $\theta(x)$ of the objective function

$$f(x) \approx \theta(x) = \sum_{k=1}^{K} \alpha_k \phi_k(x), \quad x \in \Re^n \tag{2}$$

where $\phi_k(x)$ constitutes a basis of the space of quadratic polynomials and the coefficients α_k are determined such that $\theta(x^k) = f(x^k)$ at K sampling points x^k. For a unique definition of a quadratical model in n variables one needs at least $\frac{1}{2}(n+1)(n+2) = K$ points and their function values.

In the trust region based process, instead of the objective function (1), the interpolation model (2) is minimized by applying a standard optimization method within a trust region, i.e. find

$$\begin{gathered} \min \theta(x_i) \\ \text{w.r.t. } \|x_i - x_i^k\|_\infty \leqslant \nabla^k \\ l_i \leqslant x_i \leqslant u_i, \quad i = 1, \dots, I \quad \text{box constraints} \\ g_m(x_i) \geqslant 0, \quad m = 1, \dots, M \quad \text{inequality constraints} \\ h_n(x_i) = 0, \quad n = 1, \dots, N \quad \text{equality constraints} \end{gathered} \tag{3}$$

where x_i^k is the midpoint of the trust region for the kth iteration step, ∇^k is the trust region radius for the kth iteration.

A distinguished feature of Powell's UOBYQA method, later on inherited by CONDOR, is its use of two trust region radii. In these methods the optimization process is calculated

$$\text{w.r.t.} \quad \rho^k < \|x_i - x_i^k\|_2 < \nabla^k \tag{4}$$

where the additional variable ρ^k represents the average distance between the sample points at iteration k.

CONDOR combines the advantages of both trust region and line-search worlds via using a trust region due to its robustness and speed when confronted to highly non-linear objective function and using line-search techniques because of their superiority when confronted to non-linear constraints. It uses the Moré and Sorensen algorithm [26] which is numerically very stable in step calculation and leads to very fast convergence. When no non-linear constraints are active, then the step calculation is performed by using an Augmented Lagrangian method. CONDOR is specially designed for small dimensional problems and computationally intensive load objective functions. It constructs a fully quadratic model of the objective function based on Lagrange interpolation and the curvature information is obtained from the quadratical model. The main benefit of this approach is that the local gradient of the objective function with respect to the design variables does not have to be provided by the flow solver.

2.2. Genetic algorithms

Evolutionary methods cover a group of search and optimization algorithms based upon the natural evolution. One of the main representatives is the Genetic Algorithm (GA),which was invented by John Holland in 1960s [17] and has been developed successfully to a useful search and optimization process for more than decades. Although conceptually simple, this algorithm is sufficiently complex to provide robust and powerful search mechanisms. As a direct method, GA does not require any gradient information. Therefore, it may need more function evaluations but is suitable to be applied to fluid flow problems. The next subsection details the working principle and some exemplary genetic operators which we will apply to our numerical experiments. Here only the real-coded genetic algorithms [27, 28] are concerned as they have superior ability for handling continuous search space optimization problems.

2.2.1. Working principle. The working principle of GAs is to spread a set of solutions in the potential design space in a randomized manner. In real-coded genetic algorithm, the solution will be represented directly by using a vector of real parameters, these parameters are also called individuals. Then each solution is assigned a fitness value related to the objective function in the optimization problem. Thereafter, the solutions are varied iteratively by the selection, recombination and mutation, which are formalized by mimicking the evolution phenomena in nature, towards an optimal state. Selection operator decides which solutions are maintained and used as parents to produce new solutions for the proceeding generation. New solutions are the combinations of existing good solutions with some occasional variations. They are created by recombination and mutation operators. The procedure is illustrated in Figure 1.

Selection: Inspired by the role of natural selection in evolution, the selection operator selects two parent solutions for generating new solutions, i.e. offsprings. In the selection, a solution with a high fitness value has more chance to be selected as one of the parents than a solution with a low fitness value. There are several selection schemes which are often employed, such as roulette wheel selection, tournament selection, and so on. The tournament selection scheme is getting increasingly popular because of its implicitness and controlled takeover property [29], where a number of solutions is chosen randomly from the population and the best solution from this group is selected as parent. This process is repeated as often as solutions must be chosen. These selected parents produce offsprings afterwards. The tournament size τ takes values ranging from 2 to N (number of solutions in the population).

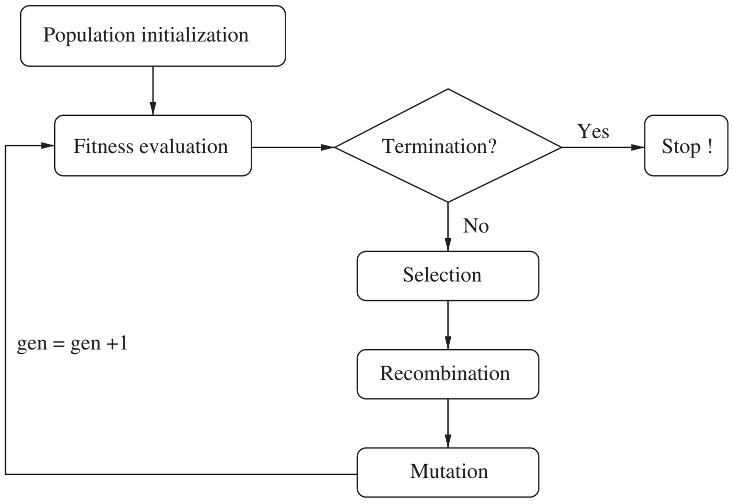

Figure 1. Flow chart of genetic algorithms.

Recombination: Recombination operator defines the way to create new solutions by combining and varying the selected parents operators. A detailed description of real-coded recombination operators can be found in References [30, 31]. The operator used here is called simulated binary crossover (SBX) [32]. It is one of the parent-centric operators, whose idea is to assign equal probability for each parent to create offsprings and more probability to create offsprings near the parents. In Reference [33], it has been argued that choosing parent-centric recombinations are more meaningful than mean-centric recombinations for a steady and reliable search. In SBX, two parents x_i^{p1} and x_i^{p2} are involved in the recombination procedure and two offsprings x_i^{o1} and x_i^{o2} are supposed to be created. A spread factor β_i is defined as the ratio of the absolute difference in offspring values and parents values

$$\beta_i = \left| \frac{x_i^{o2} - x_i^{o1}}{x_i^{p2} - x_i^{p1}} \right| \tag{5}$$

From the probability distribution function

$$P(\beta_i) = \begin{cases} 0.5(\eta_c + 1)\beta_i^{\eta_c} & \text{if } \beta_i \leqslant 1 \\ 0.5(\eta_c + 1)^{1/\beta_i^{\eta_c+2}} & \text{otherwise} \end{cases} \tag{6}$$

the ordinate β_{qi} can be calculated by equation

$$\beta_{qi} = \begin{cases} (2\mu_i)^{1/\eta_c+1} & \text{if } \mu_i \leqslant 0.5 \\ \left[\dfrac{1}{2(1-\mu_i)}\right]^{1/\eta_c+1} & \text{otherwise} \end{cases} \tag{7}$$

so that the area under the probability curve from 0 to β_{qi} is equal to a randomly chosen number μ_i.

The distribution index η_c is any non-negative real number and its value is proportional to the probability for creating offsprings near the parents. After obtaining β_{qi}, we can get the offsprings as follows:

$$\begin{aligned} x_i^{o1} &= 0.5[(1+\beta qi)x_i^{p1} + (1-\beta qi)x_i^{p2}] \\ x_i^{o2} &= 0.5[(1-\beta qi)x_i^{p1} + (1+\beta qi)x_i^{p2}] \end{aligned} \tag{8}$$

In SBX, since the distance between offsprings is proportional to the distance of parents (5), the optimization will get a large search space at the beginning as the parents are randomly generated and far away from each other and also self-adaptively narrow down the search space at the later iterations to achieve convergence to optima.

Mutation: In real-coded GA, the mutation operator adds perturbations to the individuals with a probability. It ensures sufficient population to be spread in the decision variable space, therefore GAs own the global search ability after an infinite computation time. The added perturbations are created randomly by a suitable distribution. And the mutation probability is inversely proportional to the number of design variables. It means that more design variables will lead to less chance for an individual to undergo mutation. Based on the type of distributions, there are a number of mutation operators such as random mutation, non-uniform mutation [34], normally distributed mutation, and polynomial mutation [35]. In this paper, we use the polynomial mutation operator, which employs a polynomial function as the probability distribution

$$x_i^{\text{new}} = x_i^{\text{old}} + (x_i^{\max} - x_i^{\min})\bar{\delta}_i \tag{9}$$

where $\bar{\delta}_i$ is calculated from polynomial probability distribution

$$\bar{\delta}_i = \begin{cases} (2r_i)^{1/(\eta_m+1)} - 1 & \text{if } r_i < 0.5 \\ 1 - [2(1-r_i)]^{1/(\eta_m+1)} & \text{if } r_i \geqslant 0.5 \end{cases} \tag{10}$$

in which a user-determined parameter η_m produces a perturbation of order $O(1/\eta_m)$ and r_i is a random number between 0 and 1.

2.2.2. *Simplified* NSGA-II.

In this paper, we use the simplified non-dominated sorting genetic algorithm (NSGA-II). NSGA-II is proposed by Deb *et al.* [11] and has demonstrated its superior performance on a number of multi-objective optimization problems [36–39]. Designed to be an elitist strategy, it preserves best solutions by the application of selection operators on the combined parent–offspring population R_g. Nevertheless, for solving single-objective optimization problem, instead of a set of optimal solutions, the procedures of non-dominated sort and crowded selection could be simplified by just choosing the best N solutions from R_g into the next generation. SIMPLIFIED NSGA-II inherits the elitist property of NSGA-II. Thereby, compared with normal GA, it has the ability of keeping the best solutions in the parents generation, without being affected by genetic operators such as crossover and mutation operators. The main loop of the SIMPLIFIED NSGA-II is as follows:

1. Generation counter gen = 0; Initialize parent population P_0 and offspring population Q_0 of size N, respectively.

2. Function evaluation, i.e. fitness assignment of P_g and Q_g.
3. $R_g = P_g \cup Q_g$.
4. Choose the best N solutions from R_g as P_{g+1}.
5. Termination: if stopping criterion is satisfied, then set $P_{\text{final}} = P_{g+1}$ and terminate.
6. Select parents for mating process.
7. Perform crossover and mutation operators to create new population Q_{g+1}.
8. $g = g + 1$.
9. Go to step 2.

3. GEOMETRY DEFORMATION

In a numerical design optimization process, the shape deformation is obtained by deforming the computational grid used by the flow solver. For a given mesh of M grid points, the shape of the model is defined by the $3M$ dimensional vector $\mathbf{G}$ which contains all coordinates $\mathbf{g}(j),\ j = 1, \ldots, M$ of the grid points $\mathbf{G} = (\mathbf{g}(1), \mathbf{g}(2), \ldots, \mathbf{g}(M))$. Changing the geometry means changing the grid point coordinates. Denoting the vector of initial grid points with $\mathbf{G}^{\text{ini}}$ and the vector of deformed grid points with $\mathbf{G}^{\text{def}}$, this process can be summarized by the equation

$$\mathbf{G}^{\text{def}} = \mathbf{G}^{\text{ini}} + \mathbf{T} \tag{11}$$

where $\mathbf{T}$ is the $3M$ dimensional deformation vector consisting of the deformation information for each node. Since in this classical approach for each grid point a deformation has to be defined, the number of design variables (DVs) is equal to the number of grid point coordinates. However, for the variation of a flow geometry it is important to employ methods which reduce the number of the DVs because of the high computational costs. For this purpose, a tool based on free form deformation (FFD) technique [40] is employed.

In this technique the shape and its deformation is defined through a few control points compared to the number of nodes needed for the discretization of the shape for a simulation. An example of a 3-dimensional geometry deformation with 4 control points is given in Figure 2. In this technique the deformation is defined as

$$\mathbf{G}^{\text{def}} = \mathbf{G}^{\text{ini}} + \sum_{i=1}^{N} x_i \mathbf{T}_i \tag{12}$$

where $\mathbf{T}_i$ are N shape basis vectors (SBVs) defining the deformation direction and x_i are the coefficients of the shape basis vectors acting as N design variables (DVs) for the optimization process.

The shape basis vectors are defined and computed once before the optimization process starts by applying a free form deformation technique. The generation of shape basis vectors is achieved by the deformation of a unit cube in a logical coordinate system where the points within the unit cube are denoted by

$$\xi^0 = \begin{bmatrix} s \\ t \\ u \end{bmatrix}, \quad s, t, u \in [0, 1] \tag{13}$$

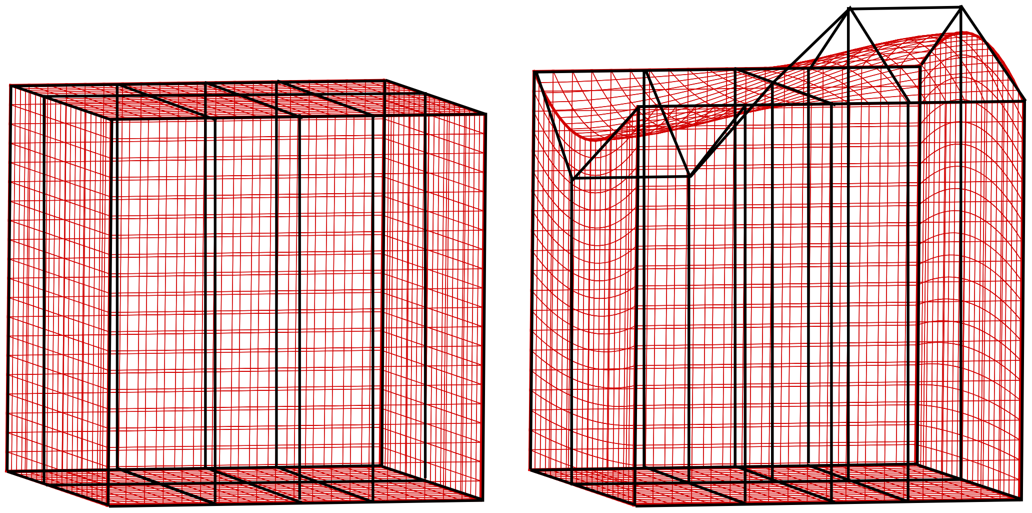

Figure 2. An example of 3-dimensional geometry deformation with FFD technique obtained by moving 4 control points. (left: original shape; right: deformed shape).

By dividing the unit cube equidistantly in all directions, the so-called control points are defined by

$$\boldsymbol{\pi}^0 = \begin{bmatrix} d/\delta \\ e/\varepsilon \\ f/\zeta \end{bmatrix}, \quad d = \{0, \ldots, \delta\}, \quad e = \{0, \ldots, \varepsilon\}, \quad f = \{0, \ldots, \zeta\} \tag{14}$$

where δ, ε and ζ represent the total number of control points in each direction. Moving the control points from their initial position leads to a deformation of the geometry inside the unit cube and the resulting displacement field can be used as the shape basis vector corresponding to that displacement. Usually individual or pairs of control points inside one face of the unit cube are moved to create a shape basis vector, i.e. $\boldsymbol{\pi}^0_{\text{def}} \rightarrow \boldsymbol{\pi}^i_{\text{def}}$. The new points are obtained by

$$\boldsymbol{\xi}^i = \sum_{d=0}^{\delta} \sum_{e=0}^{\varepsilon} \sum_{f=0}^{\zeta} a_d^{\delta}(s) a_e^{\varepsilon}(t) a_f^{\zeta}(u) \boldsymbol{\pi}^i_{\text{def}} \tag{15}$$

where the product of three Bernstein polynomials $a_d^{\delta}(s)$, $a_{\varepsilon}^{m}(t)$ and $a_f^{\zeta}(u)$ defines the deformation. The Bernstein polynomial $a_l^m(n)$ is defined as

$$a_l^m(n) = \binom{m}{l} (1-n)^{m-l} n^l = \frac{m!}{l!(m-l)!} (1-n)^{m-l} n^l \tag{16}$$

with $n \in [0, 1]$. The number of control points in the corresponding directions defines the orders of the Bernstein polynomials.

The application of this basic procedure to a computational grid corresponding to a complex geometry configuration is done by mapping the corresponding part of the grid from the physical domain into the unit cube in the logical domain by

$$\boldsymbol{\xi} = \boldsymbol{\xi}(\mathbf{x}) = \begin{bmatrix} \xi_1(x_1, x_2, x_3) \\ \xi_2(x_1, x_2, x_3) \\ \xi_3(x_1, x_2, x_3) \end{bmatrix} \tag{17}$$

This transformation is accomplished by defining an arbitrary parametric space (shape box) which has the shape of an ordinary hexahedral element that contains the part of the geometry to be modified.

The inverse of transformation (17) is given by

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a_1^1 & a_1^2 & \dots & a_1^8 \\ a_2^1 & a_2^2 & \dots & a_2^8 \\ a_3^1 & a_3^2 & \dots & a_3^8 \end{bmatrix} \begin{bmatrix} 1 \\ \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_1\xi_2 \\ \xi_1\xi_3 \\ \xi_2\xi_3 \\ \xi_1\xi_2\xi_3 \end{bmatrix} \tag{18}$$

where the coefficients of the transformation matrix $a_0^1 \dots a_8^1$; $a_0^2 \dots a_8^2$; $a_0^3 \dots a_8^3$ can be computed by using the location of the corners of the shape box and the ones of the unit cube. The (non-linear) system (18) is solved by the Newton method. Any grid point outside the shape box, i.e. $\boldsymbol{\xi} \notin [0, 1]^3$, is not considered in the following transformations and its displacement components are set to zero.

The control points in the logical space $\boldsymbol{\pi}_{\text{def}}^0$ are also transformed to the physical space by the mapping rule (18), i.e. $\boldsymbol{\pi}_{\text{def}}^0 \rightarrow \mathbf{p}_{\text{def}}^0$. Displacing the control points into a new position $\mathbf{p}_{\text{def}}^i$, the deformed grid is determined by

$$\mathbf{x}^i = \begin{cases} \sum\limits_{d=0}^{\delta} \sum\limits_{e=0}^{\varepsilon} \sum\limits_{f=0}^{\zeta} a_d^{\delta}(s) a_e^{\varepsilon}(t) a_f^{\zeta}(u) \mathbf{p}_{\text{def}}^i, & \boldsymbol{\xi} \in [0, 1]^3 \\ \mathbf{x}^0 & \text{otherwise} \end{cases} \tag{19}$$

The shape basis vectors $\mathbf{T}_i$ are the difference between the displaced positions $\mathbf{x}^i$ and the initial positions $\mathbf{x}^0$. The overall procedure is illustrated in Figure 3.

The considered approach avoids performing expensive grid generation at each iteration step since the shape deformation within the optimization process is obtained simply by applying (12). Another major advantage is the possibility of reducing the number of the design variables. Since the choice of the deformation vectors are defined once at the beginning of the optimization process,

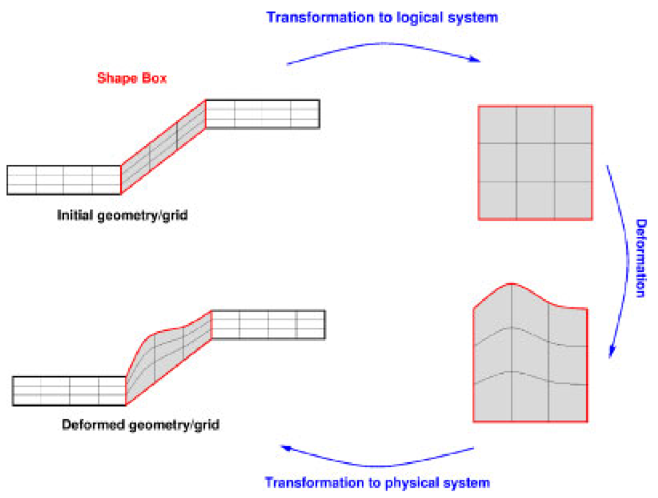
Transformation to logical system
Shape Box
Initial geometry/grid
Deformation
Deformed geometry/grid
Transformation to physical system

Figure 3. Geometry deformation with FFD technique.

the achievable shape is restricted. Hence an optimization result depends on the choice of the control points and on their displacements.

4. FLUID DYNAMICS

We consider an incompressible Newtonian fluid flow. The numerical analysis of the flow is based on the discretization of the Navier–Stokes equations with necessary initial and boundary conditions. Detailed derivation of these equations can be found in e.g. References [41, 42].

The mass conservation equation is given by

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{20}$$

The momentum conservation equations are

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho f_i \quad \text{for } i, j = 1, 2, 3 \tag{21}$$

with the pressure p, the density ρ, the body forces f_i, the velocity components u_i and the time t. For incompressible Newtonian fluids the viscous stress tensor is

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{22}$$

with the dynamic viscosity μ. Using the necessary initial and boundary conditions, the system of equations (20) and (21) can be solved for the unknowns u_i and p [21, 43].

The flow solver FASTEST [20, 21] is employed to solve the Navier–Stokes equations. The solver is based on a fully conservative finite volume method on a non-staggered, cell-centred grid arrangement, with a pressure-correction approach of SIMPLE type. For particular information about the employed computational techniques we refer to References [21, 43]. A multi-grid technique and the possibility for parallel computing allow for a high numerical efficiency for many application cases [21, 44, 45].

5. NUMERICAL RESULTS

To compare the two optimization approaches we consider the shape optimization of a 3-dimensional pipe conjunction with respect to the pressure drop Δp between inlet and outlet. A sketch of the investigated geometry is shown in Figure 4.

The object of the optimization is the shape of the geometry part B2, where the shown simple connection serves as starting solution in the optimization procedure. The flow problem domain is discretized using 32 768 control volumes. The flow medium is water with constant density and viscosity ($\rho = 1000\,\mathrm{kg/m^3}$, $\mu = 10^{-3}$ Pa). The characteristic Reynolds number is 200 based on the diameter H and the block inlet velocity.

To investigate how the final shape and efficiency is influenced by the degree of the optimization problem, different numbers of control points are employed to generate three different sets of displacement fields, i.e. shape basis vectors. For each set of shape basis vectors, the shape box around B2 is discretized equidistantly by (3,3,3), (4,3,3) and (5,3,3) (x, y, z-directions) as shown in Figure 5. The number of control points with these discretizations on the shape box surface is 24, 32 and 40, respectively. Since the control points on the corners are not allowed to be moved to assure the conjunction to B1 and B3, we have chosen 4, 8 and 12 control points for our three test cases to be moved that are directly intersecting with the surface of the pipe. The control points chosen on the shape box to be moved are illustrated in Figure 5.

The shape basis vectors corresponding to the chosen control points on B2 can be seen in Figure 6 together with the deformation directions. The deformations are performed perpendicular to the surface they act on with an initial amount of $H/20$, i.e. if the design variable is equal to 1 the actual distortion is $H/20$. The deformation in each case is bounded in y-direction between 0 and $20H$, which means that the total amount of distortion in this direction is at most $H/20 * 20H = H^2$. In the xz-direction deformations different box constraints are considered as given in related problem definitions (23)–(25).

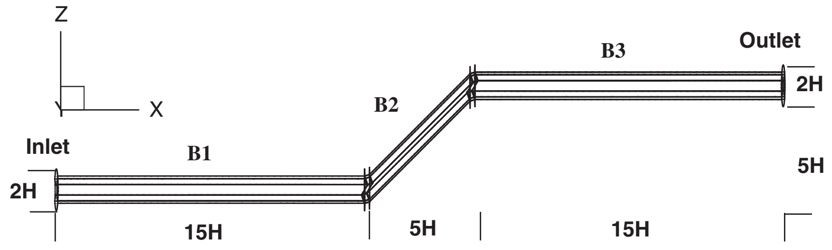

Figure 4. Sketch of the initial geometry configuration.

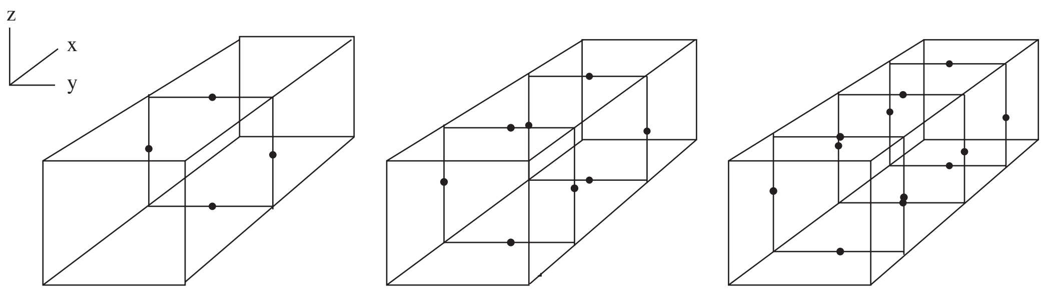

Figure 5. Chosen control points on shape boxes with (3,3,3), (4,3,3) and (5,3,3) discretizations.

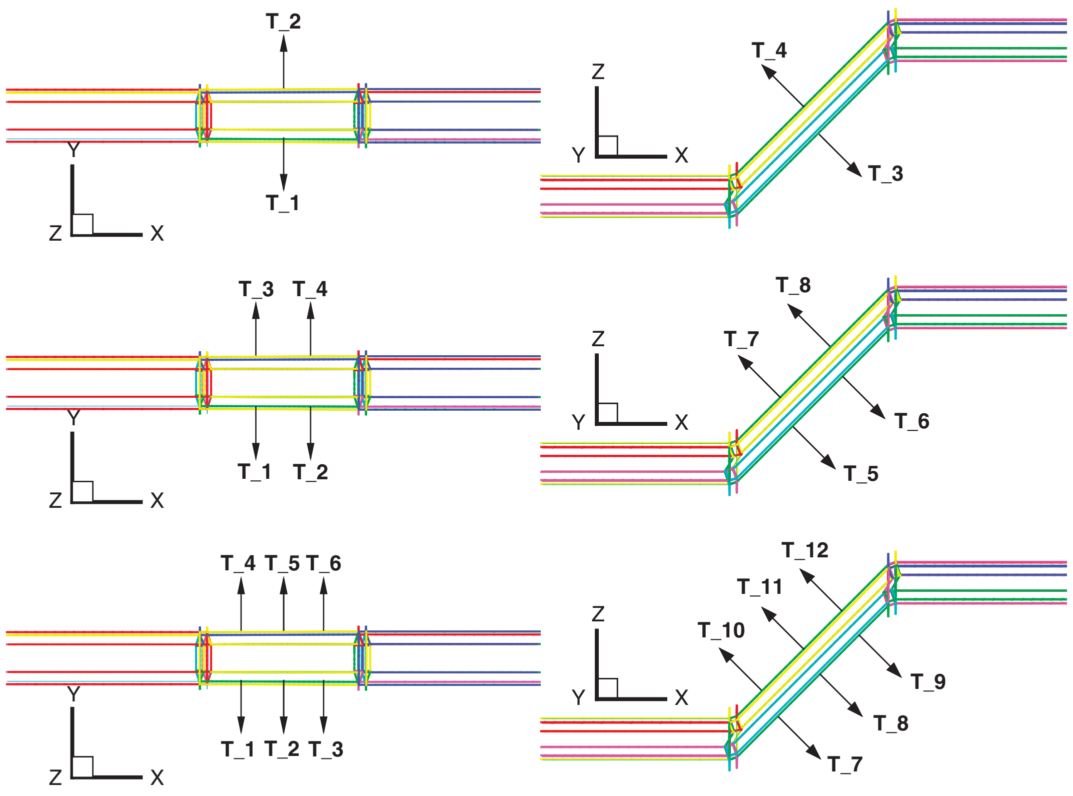

Figure 6. Shape basis vectors for the given problem for 4, 8 and 12 DVs.

In summary we consider the following three optimization problems:

$$\text{4 design variables:} \quad \min \Delta p(x_i)$$

$$0 \leqslant x_1, x_2 \leqslant 20H \qquad (23)$$

$$0 \leqslant x_3, x_4 \leqslant 15H$$

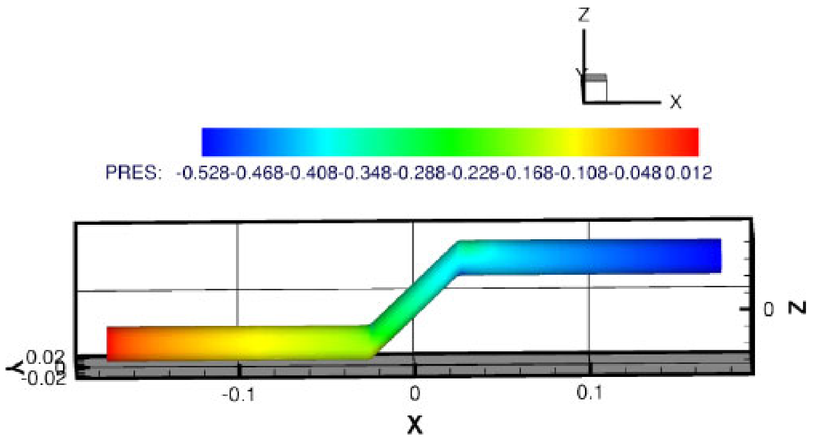

Figure 7. Pressure distribution of the initial configuration.

$$\text{8 design variables:}\quad \begin{aligned}&\min \Delta p(x_i)\\ &0\leqslant x_1,\ldots,x_4\leqslant 20H\\ &0\leqslant x_5, x_8\leqslant 8H\\ &0\leqslant x_6, x_7\leqslant 15H\end{aligned} \tag{24}$$

$$\text{12 design variables:}\quad \begin{aligned}&\min \Delta p(x_i)\\ &0\leqslant x_1,\ldots,x_6\leqslant 20H\\ &0\leqslant x_7, x_{12}\leqslant 8H\\ &0\leqslant x_8, x_{11}\leqslant 9H\\ &0\leqslant x_9, x_{10}\leqslant 15H\end{aligned} \tag{25}$$

For all three test cases both optimization tools, CONDOR and SIMPLIFIED NSGA-II start the optimization process with the same initial geometry configuration (Figure 4). For CONDOR the initial trust region radius is taken as the interval of the box constraints, given in Equations (23)–(25) where the optimization problems are summarized, and the initial sampling distance as H. If the sampling distance falls below 0.0001 the optimization process is stopped automatically.

SIMPLIFIED NSGA-II is employed with a predefined population number of 20. The initial population is generated randomly in the box constraints of the optimization problem. Tournament selection, SBX crossover for real parameters and real polynomial mutation are employed as the genetic operator in this method. We chose a relatively high recombination probability, i.e. $p_c = 0.9$. And the mutation probability is chosen by $p_m = 1/n_{\text{real}}$, where n_{real} is the number of the real design variables. Stopping criterion is defined such that the optimizer stops automatically if the optimal pressure drop does not improve after 20 generations.

In Figure 7 the pressure distribution of the initial configuration and in Figure 8 the shapes and pressure distributions at the surface of the achieved optimization results with both methods

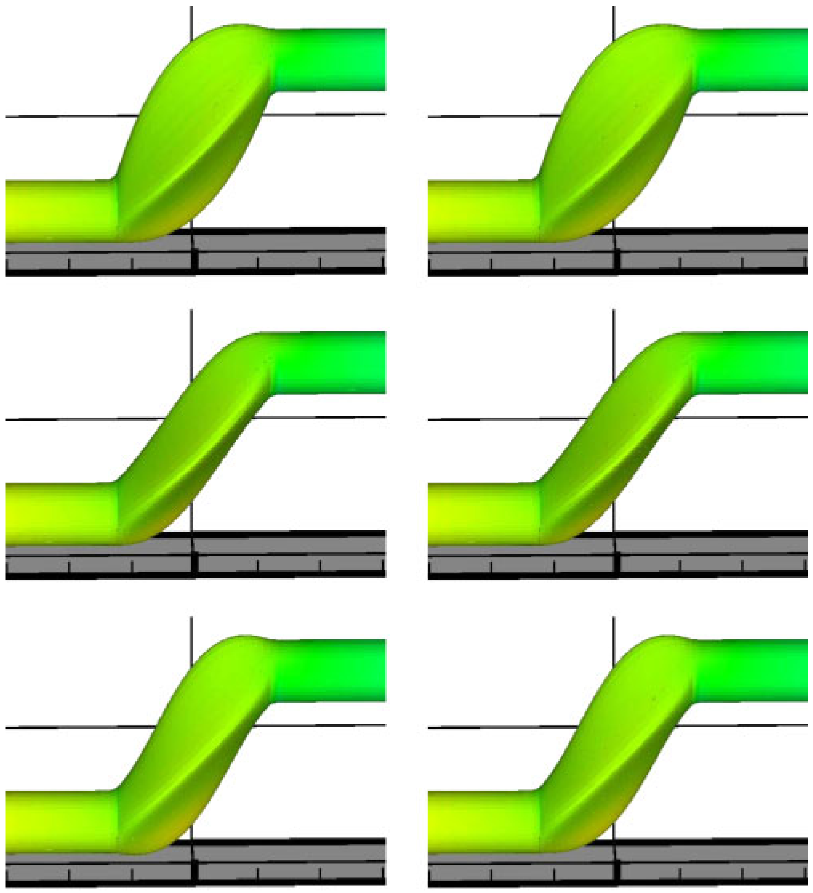
Figure 8. Pressure distribution of the optimum configurations with 4 (top), 8 (middle) and 12 (bottom) DVs calculated by CONDOR (left) and SIMPLIFIED NSGA-II (right).

are given. The corresponding deformations and pressure distributions in y-direction are given in Figure 9.

Figures 10 and 11 show pressure distributions of the found optima achieved by CONDOR with 8 and 12 design variables and achieved by SIMPLIFIED NSGA-II with 12 design variables together with streamlines. In these figures, the recirculation zones are observable at the bottom of the enlarged part of the pipe.

The reason of the achieved pressure drop is mainly the recirculation zones by enlargement of the pipe, e.g. Figures 10 and 11. Apparently, these separations help considerably in reducing the pressure drop. An explanation of this phenomenon is that one can distinguish different kinds of energy losses in such pipe junctions. On the one hand, energy is dissipated in recirculation zones, whose amount is proportional to its size and its vorticity. On the other hand, energy is employed to overcome the effect of the wall, where the no-slip condition holds. That energy loss is proportional to the normal gradient of the tangential velocity component at the wall. Compared to an attached

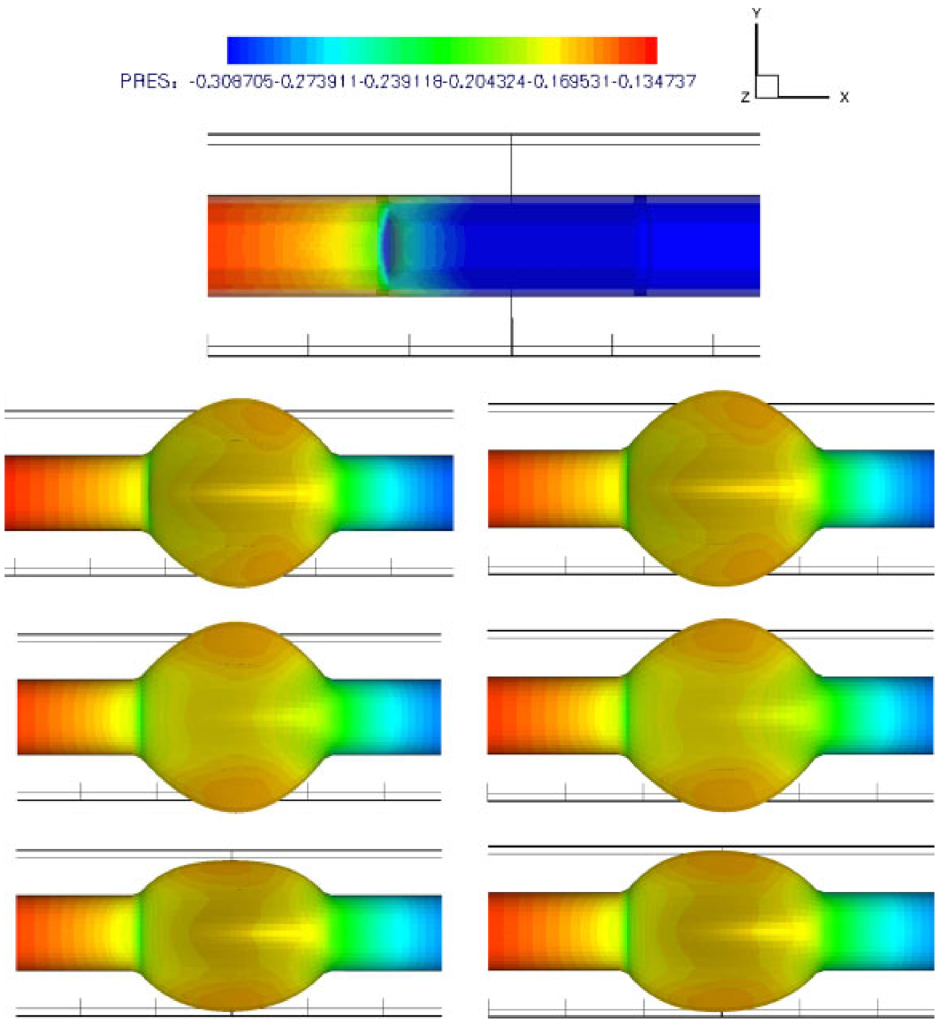

Figure 9. Pressure distributions of the initial configuration (very top) and optimum configurations with 4 (top), 8 (middle) and 12 (top) DVs and deformations to the y-direction calculated by CONDOR (left) and SIMPLIFIED NSGA-II (right).

flow this gradient is reduced by the separation. Obviously, the found shape realizes a compensation between both kinds of energy losses.

The solution vectors for the optimization problems obtained with both methods are summarized in Table I.

It is cognizable that the results both obtained by CONDOR and SIMPLIFIED NSGA-II are almost symmetric in the xz-direction deformations. For example, in the case of 12 DVs (Row 3) $x_9 = x_{10} = 0$ and $x_8 = x_{11} = 9$ and $x_7 \approx 6, x_{12} \approx 8$. The solution vectors in each case are similar to each other at many control points (Table I), but there are a few points where slightly differing

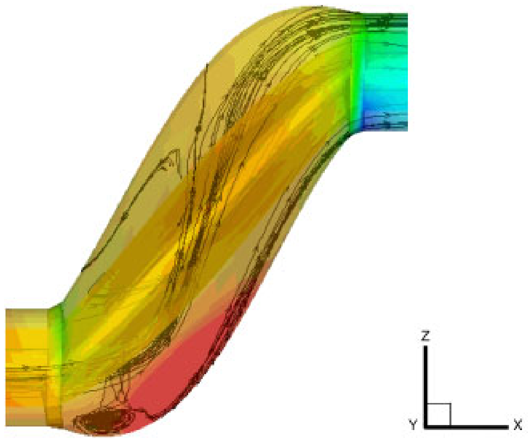

Figure 10. Recirculation of the flow noticed in the optimum with 12 DVs calculated by SIMPLIFIED NSGA-II.

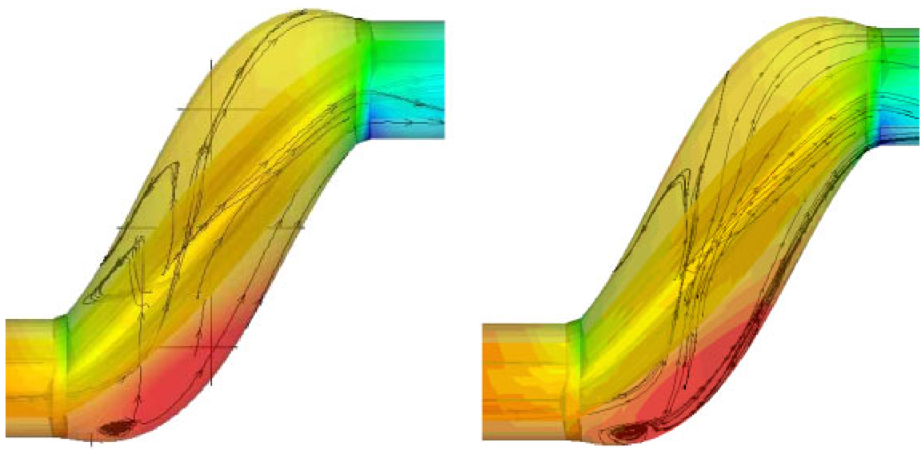

Figure 11. Recirculation of the flow noticed in the optimum with 8 and 12 DVs respectively calculated by CONDOR.

values for the two optimization techniques arise. This difference may be a result of the randomness of SIMPLIFIED NSGA-II within the box constraints.

The shapes achieved by the optimum configurations of different numbers of design variables are compared in Figures 12 and 13 illustrating the midsections at $x = 0$ and $y = 0$. Both optimization techniques yield similar results in the sense that in the case of 4 design variables the amount of the deformation is larger than in the other cases.

In Table II the total number of function evaluations, the evaluation number where the optimum is achieved and the efficiency in each case for both optimization tools are given.

Table I. Solution vectors achieved by both optimization techniques.

Case	Solution vector	With CONDOR	With SIMPLIFIED NSGA-II
4 DVs	$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix}$	$\begin{Bmatrix} 12.45 \\ 12.47 \\ 11.97 \\ 15 \end{Bmatrix} H$	$\begin{Bmatrix} 12.54 \\ 12.54 \\ 11.98 \\ 15 \end{Bmatrix} H$
8 DVs	$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{Bmatrix}$	$\begin{Bmatrix} 7.72 \\ 8.97 \\ 7.74 \\ 8.96 \\ 8 \\ 7.5 \\ 0 \\ 8 \end{Bmatrix} H$	$\begin{Bmatrix} 7.56 \\ 8.72 \\ 7.51 \\ 8.79 \\ 8 \\ 5.98 \\ 0 \\ 8 \end{Bmatrix} H$
12 DVs	$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{Bmatrix}$	$\begin{Bmatrix} 6.32 \\ 3 \\ 5.33 \\ 6.64 \\ 2.34 \\ 6.58 \\ 6.31 \\ 9 \\ 0 \\ 0 \\ 9 \\ 8 \end{Bmatrix} H$	$\begin{Bmatrix} 6.53 \\ 3.38 \\ 5.91 \\ 6.46 \\ 3.73 \\ 5.82 \\ 5.60 \\ 9 \\ 0 \\ 0 \\ 9 \\ 8 \end{Bmatrix} H$

The efficiency is given in terms of pressure drop reduction compared to the pressure drop of the initial configuration.

The optimization tool CONDOR uses the first 15, 45 and 91 (for the design variables 4, 8 and 12, respectively) iterations for the development of the quadratic approximation model. In the remaining 75, 146 and 231 iterations the actual optimization process takes place. With CONDOR the number of function evaluations nearly increases linearly with the number of design variables, while with SIMPLIFIED NSGA-II a more disproportionate behaviour is observed.

In Figure 14 the proportion of number of function evaluations of NSGA-II to CONDOR for all three cases are given. In all cases NSGA-II needs about 8 to 13 times more number of function evaluations (i.e. more computation time) than CONDOR.

Figures 15 and 16 show the history of the pressure drop reduction over the number of iterations for the 3 test cases when using CONDOR and SIMPLIFIED NSGA-II. Using SIMPLIFIED NSGA-II, the best solution is achieved at the generation number 41, 80 and 119, respectively. The corresponding function iterations are 820, 1600 and 2380 (Table II).

The achieved efficiencies for the two optimization are quite similar (see Table II). It can be deduced that for the given problem properties the efficiency is mainly bounded by the constraints and it only is slightly depending on the number of design variables for both optimization tools.

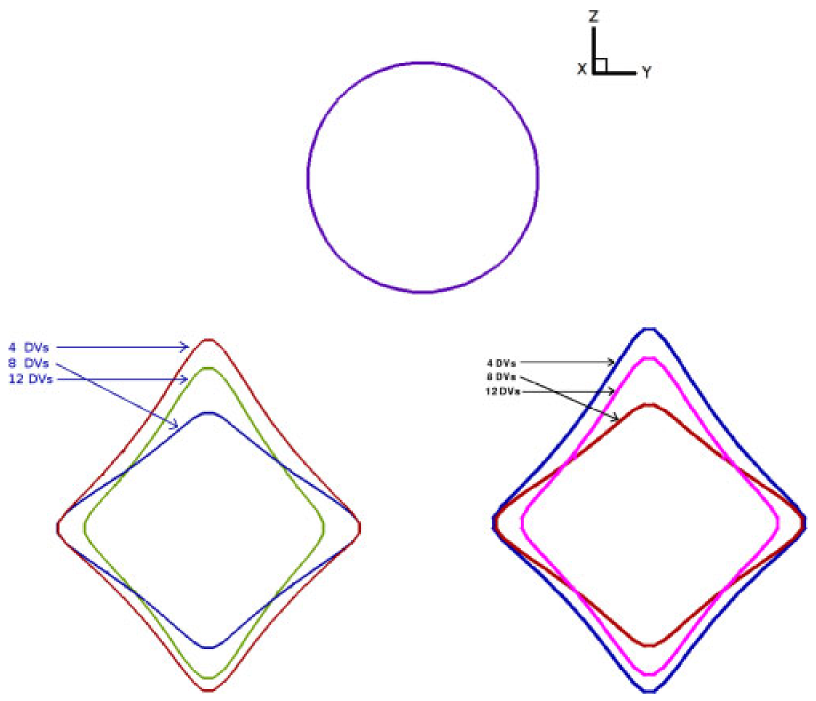

Figure 12. Midsection of the original geometry at $x=0$ (top) and the differences in the optimized geometries calculated by CONDOR (left) and SIMPLIFIED NSGA-II (right) at $x=0$.

6. CONCLUSION

We have presented a comparison between two approaches for the optimization of fluid flow geometries, i.e. derivative free Newton-based tool CONDOR and the evolutionary algorithm SIMPLIFIED NSGA-II. We have compared the methods regarding the quality of the optima and the computational efficiency.

From the presented results it can be concluded that the two optimizations techniques yield nearly the same optimal values together with nearly the same sets of solutions corresponding to similar shapes. By using different number of design variables we have investigated the influence of the degree of the optimization problem. Increasing the number of design variables for both methods yields only slightly better optima, however, with different solution sets and optimum shapes.

Concerning the computational costs of the optimization process, in all cases CONDOR requires much less number of function evaluations than SIMPLIFIED NSGA-II to reach the optimum. The number of design variables does not significantly influence the ratio of required number function evaluations. It has been seen that the Newton-based optimizer compares favourably to evolutionary methods regarding the quality of the solution and consumes much less computation time.

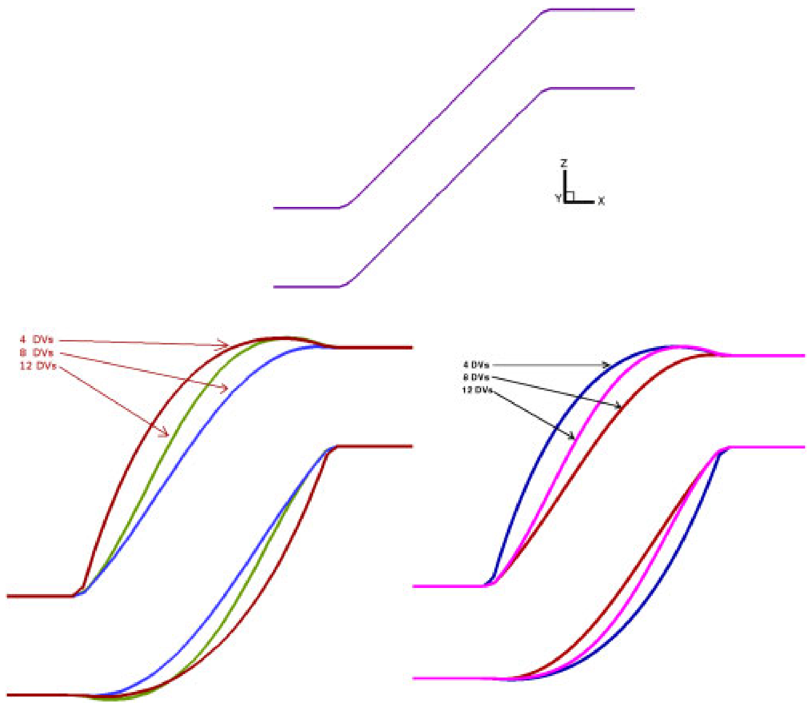

Figure 13. Midsection of the original geometry at $y=0$ (top) and the differences in the optimized geometries calculated by CONDOR (left) and SIMPLIFIED NSGA-II (right) at $y=0$.

Table II. Comparison of optimization results.

	SIMPLIFIED NSGA-II			CONDOR		
	Opt. at:	# Total FEs	Efficiency (%)	Opt. at:	# Total FEs	Efficiency (%)
4 DVs	820	1200	≈ 23.29	85	90	≈ 23.29
8 DVs	1600	1980	≈ 24.09	178	191	≈ 23.19
12 DVs	2380	2760	≈ 25.00	273	288	≈ 25.07

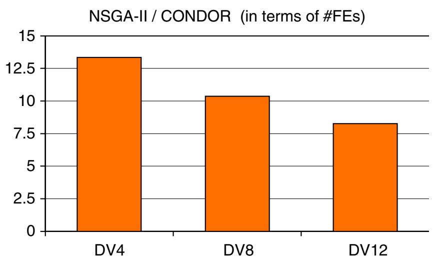

Figure 14. The proportion of number of total function evaluations of SIMPLIFIED NSGA-II to CONDOR. The figure shows how much faster CONDOR ends the optimization process.

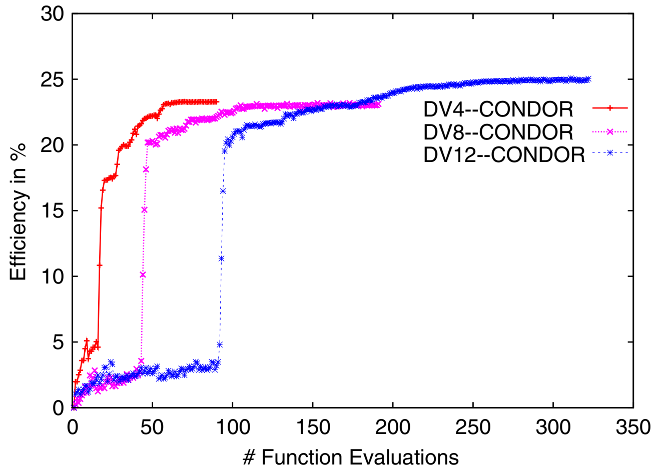

Figure 15. Efficiency history with CONDOR.

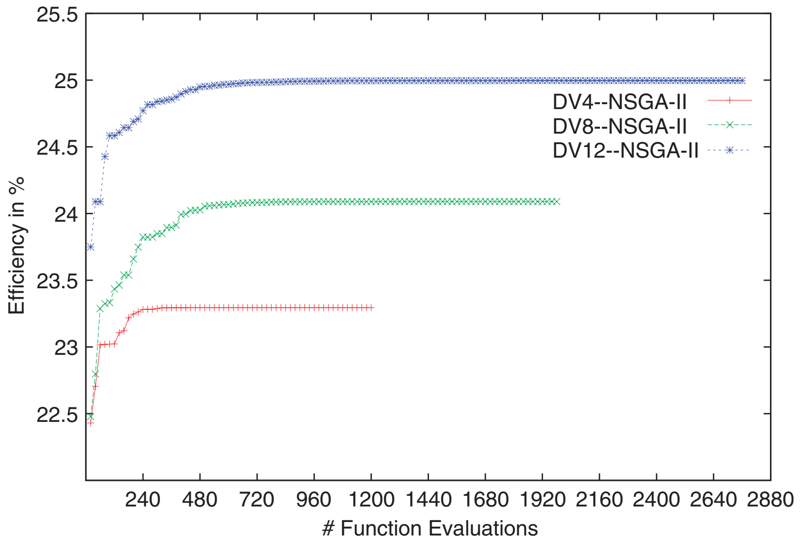

Figure 16. Efficiency history with SIMPLIFIED NSGA-II.

REFERENCES

1. Burczyński T, Adamczyk T. The boundary element formulation for multiparameter structural shape optimization. *Applied Mathematical Modelling* 1985; **9**(3):195–200.
2. Haftka R, Grandhi R. Structural shape optimization—a survey. *Computer Methods in Applied Mechanics and Engineering* 1986; **57**(1):91–106.
3. Herskovits J, Dias G, Soares CM. A full-stress technique for structural shape optimization. *Appl. Math. Comput. Sci.* 1996; **6**(2):303–319, shape optimization and scientific computations (Warsaw, 1994).

4. Harth Z, Schäfer M. Investigation of derivative-free optimization tools for optimizing flow geometries. *Evolutionary and Deterministic Methods for Design*, *Optimization and Control with Applications to Industrial and Societal Problems*, *EUROGEN 2005*, Munich, Germany, 2005.
5. Hirschen K, Schäfer M. Artificial neural networks for shape optimization in cfd. *ERCOFTAC Design Optimization*: *Methods and Applications*, *Conference Proceedings*, Athens, Greece, 2004.
6. Lehnhäuser T, Schäfer M. Efficient discretization of pressure-correction equations on non-orthogonal grids. *International Journal for Numerical Methods in Fluids* 2003; **42**:211–231.
7. Mohammadi B, Pironneau O. Shape optimization in fluid mechanics. *Annual Review of Fluid Mechanics* 2004; **36**:255–279.
8. Perry E. Three dimensional shape optimization of internal fluid flow systems using arbitrary shape deformation coupled with computational fluid dynamics. *Ph.D. Dissertation*, Brigham Young University, Provo, Utah, 1999.
9. Ronzheimer A. Post-parameterization of complex cad-based aircraft-shapes using freeform deformation. *8th International Conference on Numerical Grid Generation in Computational Field Simulations*, Honolulu.
10. Vanden Berghen F, Bersini H. Condor, a new parallel, constrained extension of Powell's UOBYQA algorithm *Technical Report TR/IRIDIA/2204-11*, 2004.
11. Deb K, Agrawal S, Pratap A, Meyarivan T. A fast elitist non-dominated sorting algorithm for multi-objective optimization: Nsga-II. *Proceedings of the Parallel Problem Solving from Nature VI Conference*, Paris, 2000; 849–858.
12. Rosenbrock HH. An automatic method for finding the greatest or least value of a function. *Computer Journal* 1960/1961; **3**:175–184.
13. Bazaraa Mokhtar S, Shetty CM. *Nonlinear Programming*, *Theory and Algorithms*. Wiley: New York, Chichester, Brisbane, 1979.
14. Powell MJD. UOBYQA: unconstrained optimization by quadratic approximation. *Mathematical Programming* 2002; **92**(3, Series B):555–582, iSMP 2000, Part 2 (Atlanta, GA).
15. Conn A, Scheinberg K, Toint PL. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming* 1997; **79**:397–414.
16. Powell MJD. The NEWUOA software for unconstrained optimization without derivatives. *Large-scale nonlinear optimization*, *Nonconvex Optim. Appl.* vol. 83. Springer: New York, 2006; 255–297.
17. Holland JH. *Adaptation in Natural and Artificial Systems*. University of Michigan Press: Ann Arbor, Mich. An introductory analysis with applications to biology, control, and artificial intelligence.
18. Harth Z, Schäfer M. Numerical shape optimization for flow configurations. *ERCOFTAC Design Optimization*: *Methods and Applications*, *Conference Proceedings*, Athens, Greece, 2004.
19. Harzheim L, Graf G, Liebers J. Shape200: a program to create basis vectors for shape optimization using solution 200 of msc/nastran. *Proceedings of the Tenth International Conference on Vehicle Structural Mechanics and CAE* 1997; **308**:219–228.
20. Durst F, Schäfer M. A parallel blockstructured multigrid method for the prediction of incompressible flows. *International Journal for Numerical Methods in Fluids* 1996; **22**:549–565.
21. *FASTEST User Manual*. Department of Numerical Methods in Mechanical Engineering, Darmstadt University of Technology, Darmstadt, Germany, 2004.
22. Wright MH. Direct search methods: once scorned, now respectable. *Numerical Analysis* (*Dundee 1995*), vol. 344, Pitman Res. Notes Math. Ser.: Longman, Harlow, 1996; 191–208.
23. Spendley W, Hext GR, Himsworth FR. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics* 1962; **4**:441–461.
24. Powell MJD. On the convergence of the DFP algorithm for unconstrained optimization when there are only two variables. *Mathematical Programming* 2000; **87**(2, Series B):281–301, studies in algorithmic optimization.
25. Powell MJD. On trust region methods for unconstrained minimization without derivatives. *Mathematical Programming* 2003; **97**(3, Series B):605–623, new trends in optimization and computational algorithms (NTOC 2001) (Kyoto).
26. Moré J, Sorensen D. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing* 1983; **4**(3):553–572.
27. Beyer HG, Deb K. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation Journal* 2001.
28. Deb K, Beyer HG. On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 2001.
29. Goldberg DE, Deb K. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms* (Bloomington, IN, 1990). Morgan Kaufmann: San Mateo, CA, 1991; 69–93.

30. Deb K, Beyer HG. Self-adaptive genetic algorithms with simulated binary crossover. *Technical Report*, Department of Computer Science/*XI*, University of Dortmund, Germany, 2001.
31. Verdegay JL, Herrera F, Lozano M. Tackling real-coded genetic algorithms: operators and tools for behavioral analysis. *Artificial Intelligence Review* 1998.
32. Agrawal RB, Deb K. Simulated binary crossover for continuous search space. *Complex System* 1995.
33. Deb K. A population-based algorithm-generator for real-parameter optimization. *Technical Report*, *KanGAL Report*, 2003.
34. Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer: Berlin, 1992.
35. Goyal M, Deb K. A combined genetic adaptive search (geneas) for engineering design. *Computer Science and Informatics* 1996.
36. Deb K, Mohan M, Mishra S. A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions. *Technical Report*, *KanGAL Report*, 2003.
37. Zitzler E, Thiele L. Multiobjective evolutionary alogrithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation* 1999; **3**(4):257–271.
38. Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation* 2000; **8**(2):173–195.
39. Zitzler E, Deb K, Thiele L. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation* 2002; **10**(3):263–282.
40. Sederberg T, Parry S. Free form deformation of solid geometric models. *Proceedings of SIGGRAPH'86*, August 1986; 151–159.
41. Brodkey RS. *The Phenomena of Fluid Motions* (1st edn). Dover Publications: Newyork, 1995.
42. Spurk J. *Fluid Mechanics*. Springer: Berlin, 1997.
43. Schäfer M. *Computational Engineering—Introduction to Numerical Methods*. Springer: Berlin, 2006.
44. Basara B, Durst F, Schäfer M. A parallel multigrid method for the prediction of turbulent flows with Reynolds stress closure. In *Parallel Computational Fluid Dynamics*, vol. 95. Elsevier: Amsterdam, 1996; 347–354.
45. Schäfer M. Large-scale scientific flow computations, large-scale scientific computations of engineering and environmental problems. *Notes on Numerical Fluid Mechanics* 1998; **62**:98–110.