JYVÄSKYLÄ STUDIES IN COMPUTING
116

Jukka Toivanen

Shape Optimization Utilizing Consistent Sensitivities

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 116

Jukka Toivanen

Shape Optimization Utilizing Consistent Sensitivities

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen auditoriossa 2
elokuun 26. päivänä 2010 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in the building Agora, auditorium 2, on August 26, 2010 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2010

Shape Optimization Utilizing Consistent Sensitivities

JYVÄSKYLÄ STUDIES IN COMPUTING 116

Jukka Toivanen

Shape Optimization Utilizing Consistent Sensitivities

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2010

Editor
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo, Sini Rainivaara
Publishing Unit, University Library of Jyväskylä

URN:ISBN:978-951-39-3990-8
ISBN 978-951-39-3990-8 (PDF)

ISBN 978-951-39-3968-7 (nid.)
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2010

ABSTRACT

Toivanen, Jukka
Shape Optimization Utilizing Consistent Sensitivities
Jyväskylä: University of Jyväskylä, 2010, 54 p.(+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 116)
ISBN 978-951-39-3990-8 (PDF), 978-951-39-3968-7 (nid.)
Finnish summary
Diss.

This thesis deals with gradient based methods to solve shape optimization problems governed by partial differential equations (PDE). Automatic differentiation (AD) techniques provide a straightforward way to augment new and existing PDE solvers with derivative computation routines. We present a novel implementation of the so called sparse forward mode AD, which provides an automatic way to exploit sparsity in derivative computations. Using this technique it is possible to compute large sparse Jacobians of vector functions so that only minimal changes to the original code are required. Moreover, this technique can be used in the context of the discrete adjoint approach to efficiently compute large shape gradients. The implementation has only slightly larger computational overhead than traditional dense mode implementations.

The AD technique is used to implement shape sensitivity analysis capabilities into an existing electromagnetic solver based on the methods of moments, and the solver is used to solve various shape optimization problems related to antenna design. Sensitivity analysis is also implemented in the context of the finite element method, and this implementation is used for example to solve a fibre orientation control problem in a simplified paper machine headbox. Shape optimization governed by the Bernoulli free boundary problem is also considered. To this end, a so called pseudo solid approach is used to develop a solver that enables efficient solution of the free boundary problem, as well as the shape sensitivity analysis.

Keywords: shape optimization, sensitivity analysis, automatic differentiation

Author

Jukka Toivanen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Supervisor

Professor Raino A. E. Mäkinen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Reviewers

Professor Olivier Pironneau
Laboratoire Jacques-Louis Lions
Université Paris VI
France

Professor Ole Sigmund
Department of Mechanical Engineering
Danmarks Tekniske Universitet
Denmark

Opponent

Docent, Dr. Ing. Eduard Rohan
Department of Mechanics & New Technologies Research Centre
University of West Bohemia
Czech Republic

ACKNOWLEDGEMENTS

First of all, I want to thank my supervisor professor Raino A. E. Mäkinen for his guidance during my thesis work. I would also like to express my gratitude to the people at Numerola Ltd, especially Kai Hiltunen, Pasi Tarvainen, and Antti Niemistö, for introducing me to the world of numerical simulations, and encouraging me to start doctoral studies.

This work has been financially supported by Tekniikan Edistämissäätiö, Ellen and Artturi Nyyssönen Foundation, COMAS graduate school, and Finnish Cultural Foundation, which is hereby acknowledged. While writing my thesis, I have also worked in a project funded by the Nokia Corporation, and indirectly by TEKES.

For fruitful and inspiring collaboration I thank the co-authors of the articles included in this thesis, namely Jaroslav Haslinger, Jussi Rahola, Pasi Ylä-Oijala, and Seppo Järvenpää. Many thanks also to the other people from Numerola, Nokia, and Helsinki University of Technology, with whom I have worked during the recent years.

My colleagues from the Department of Mathematical Information Technology, and especially the simulation and optimization coffee group ”kahvisimot”, deserve a special thanks for making these years so enjoyable. The same goes for the ”Hietavala dynasty”, and all my friends, especially ”the ghetto posse”. Our common free time activities really helped me to relax and ”reboot” my mind every once and a while.

Finally, for all your love and support, I wish to express my deepest gratitude to my mother Mirja and my wife Tanja. You mean the world to me.

GLOSSARY

$\boldsymbol{q}$	$= (q_1, \ldots, q_N)^T$ degrees of freedom
$\boldsymbol{r}$	residual vector
N_n	number of nodes in the mesh
$\boldsymbol{\alpha}$	$= (\alpha_1, \ldots, \alpha_M)^T$ design variables
Ω	domain in $\mathbb{R}^m$
$\partial\Omega$	boundary of Ω
$\boldsymbol{x}$	$= (x^1, \ldots, x^m)^T$, a point in $\mathbb{R}^m$
u	function from $\mathbb{R}^m \rightarrow \mathbb{R}^n$
x_i^j	j^{th} coordinate of i^{th} mesh node
$\boldsymbol{X}$	vector of mesh nodal coordinates $(x_1^1, x_2^1, \ldots, x_{N_n}^m)^T$
J	$= J(u, \Omega)$ objective functional
$\boldsymbol{G}$	mesh deformation operator

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF INCLUDED ARTICLES

PI Jukka I. Toivanen and Raino A. E. Mäkinen. Implementation of Sparse Forward Mode Automatic Differentiation with Application to Electromagnetic Shape Optimization. *To appear in Optimization Methods & Software*.

PII Jukka I. Toivanen. A Non-Linear Mesh Deformation Operator Applied to Shape Optimization. *Proceedings of ERCOFTAC 2006 Design Optimization: Methods & Applications, Las Palmas, Spain*, 2006.

PIII Jan Stebel, Raino A. E. Mäkinen, and Jukka I. Toivanen. Optimal Shape Design in a Fibre Orientation Model. *Applications of Mathematics, 52, pp. 391–405*, 2007.

PIV Jukka I. Toivanen, Jaroslav Haslinger, and Raino A. E. Mäkinen. Shape optimization of systems governed by Bernoulli free boundary problems. *Computer Methods in Applied Mechanics and Engineering, 197, pp. 3803–3815*, 2008.

PV Jukka I. Toivanen, Raino A. E. Mäkinen, Seppo Järvenpää, Pasi Ylä-Oijala, and Jussi Rahola. Electromagnetic Sensitivity Analysis and Shape Optimization Using Method of Moments and Automatic Differentiation. *IEEE Transactions on Antennas & Propagation, 57, pp. 168–175*, 2009.

PVI Jukka I. Toivanen, Raino A. E. Mäkinen, Jussi Rahola, Seppo Järvenpää, and Pasi Ylä-Oijala. Gradient-based shape optimisation of ultra-wideband antennas parameterised using splines. *To appear in IET Microwaves, Antennas and Propagation*.

1 INTRODUCTION

Shape plays a crucial role in the design of many objects. Classical examples are cars [1] and aeroplanes [2], where the shape affects the aerodynamic drag, and thus the fuel efficiency. Traditional way to design such objects is to evaluate existing configurations based on measurements, and use intuition and experience to find ways for improvement. Often this kind of approach results in a design that is acceptable, but not the best possible.

Advances in computational power and mathematical modeling have provided an alternative approach to this design process. At least to some extent, measurements can be replaced with computer simulations when evaluating candidate designs. This offers several benefits: computer simulation is much cheaper and often faster than building prototypes and setting up experiments, which in turn enables quick experimentation with many different shapes. Moreover, computers can be harnessed to explore design candidates in a systematic fashion and seek the best possible one, i.e. to perform shape optimization.

Numerical shape optimization has been utilized for at least three decades [3, 4, 5], and the mathematical foundations date back even further. Still, despite the enormous potential, shape optimization capabilities are not a standard part of commercial simulators. This thesis aims to promote generic and simple techniques that enable the shape sensitivity analysis for a wide range of new and existing simulation codes.

In Section 2 of this thesis we introduce a simple model shape optimization problem, and take a look at some problems caused by the remeshing of the computation domain. To avoid the need for remeshing, a mesh deformation approach is often used. Article [PII] presents one possible technique for that purpose.

Sensitivity analysis is a key ingredient in efficient optimization methods. It can also serve other purposes. For example in [6] shape sensitivity analysis is used to predict the variation in the performance of microwave devices due to geometrical uncertainties in the manufacturing process. It also plays a crucial role in many solution strategies for free boundary problems [7]. In Section 3 we review possible approaches for the shape sensitivity analysis. For the reasons mentioned in Section 3.6 we will rely on discrete level sensitivities computed

with the aid of automatic differentiation [8] (AD, sometimes called algorithmic or computational differentiation).

The article [PI] presents our implementation of one particular AD technique, called the sparse forward mode automatic differentiation. By numerical examples, the technique is shown to be reasonably efficient in terms of the computational overhead. Perhaps even more significant is the ease of use offered by the technique. Once the AD operations have been implemented, the approach provides a transparent way to exploit the sparsity in derivative computations.

In Section 4 we describe our implementation of the finite element method, which is used to perform many of the computations presented in this thesis. Moreover, we take a look at the application of the automatic differentiation in the solution process of non-linear problems and in the shape sensitivity analysis.

In Section 5 we present another model problem, for which we perform the shape sensitivity analysis. The effect of incomplete convergence of the state problem solution on the results of the sensitivity analysis are discussed, as well as computational complexity of performing the shape sensitivity analysis with respect to a large number of design variables.

The articles [PIII, PIV, PV, PVI] include applications of shape optimization. The article [PIII] deals with the fibre orientation control in a simplified paper machine headbox. In the article [PIV] we consider optimization of systems where the state problem is a so called Bernoulli type free boundary problem, which arises for example in the modelling of ideal fluid flow or electro-chemical machining. The articles [PV, PVI] deal with antenna shape optimization, where the state problem is governed by the time-harmonic Maxwell equations. The article [PV] also highlights the fact that makes the automatic differentiation approach tempting: implementing sensitivity analysis capabilities into an existing simulator was only a matter of days in a case where manual implementation of sensitivity computations had been previously deemed infeasible due to complexity of the numerical methods utilized in the solver [9].

1.1 Setting of the problem

The physical phenomena that give rise to the shape optimization problems are often modelled using partial differential equations. Such a partial differential equation (PDE), or a set of equations, is often called the *state problem*, since it models the state of the studied system. Let us denote our state problem by

$$B(u, \Omega, \boldsymbol{\beta}) = 0, \tag{1}$$

where $u = u(\Omega) : \mathbb{R}^m \to \mathbb{R}^n$ is the solution to the partial differential equation(s) in the domain Ω, and $\boldsymbol{\beta}$ denotes the input data, such as the boundary conditions. The function u can be scalar valued (e.g. temperature in the domain Ω), a vector quantity (e.g. velocity field of fluid) or a combination of scalar and vector fields. Dimension m of Ω is typically 2 or 3. Physical phenomena also quite often depend

on time, but time dependences are handled using special techniques, and will not be considered within this thesis.

Using this notation, the shape optimization problem can be written as a constrained optimization problem:

$$\min_{\Omega \in U^{ad}} J(u(\Omega), \Omega, \boldsymbol{\beta}) \tag{2}$$

subject to the state constraint (1). Here J is so called *objective functional* and U^{ad} is the set of admissible domains. In what follows we shall suppose that the state problem has an unique solution for every $\Omega \in U^{ad}$. The optimization problem may have other constraints, which can also depend on the solution u, and multi-criteria optimization problems [10] even have several objective functionals. The techniques presented in this thesis can be extended to these cases without too much difficulty.

The choice of the set of admissible domains is important in order to assure that the optimization problem is mathematically well posed [11]. There might also be practical reasons to pose some limitations on the shapes that are allowed. Without proper limitations the domains resulting from the optimization can be arbitrarily complex, exhibiting for example more and more oscillations on the boundary as the optimization proceeds. If the optimization considers a model of a real world device, such a result may be impossible or impractical to manufacture, and therefore of no real value.

In *sizing optimization* only some dimensions related to the geometry are optimized, and the set of admissible geometries is therefore quite limited. When non-trivial changes in the shape are allowed, one often speaks of *shape optimization*. *Topology optimization* permits also changes in the topology, such as creating new holes inside the geometry.

Topology optimization is considered a discipline of its own, because standard shape optimization techniques are not capable of handling changes in the topology. For example, if the boundary of the domain intersects itself, some location in space can be occupied by the material "twice" or have a "negative" volume (see Figure 1). In these cases the numerical solution of the state problem will in general be incorrect.

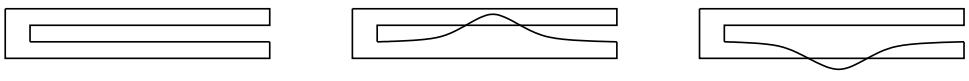

FIGURE 1 Initial geometry and two illegal deformed shapes.

One way to realize the topology optimization problem is to consider a binary design variable in each element of the computation mesh such that the variable defines whether or not material is present in that element. However, such a combinatorial optimization problem is very hard to solve, and therefore heuristics such as genetic algorithms are sometimes used [12]. Another remedy is to consider a relaxed problem. A topology optimization technique known as

the solid isotropic material with penalization (SIMP) method [13], developed for structural mechanics problems, relates the rigidity of each element to a design variable that can obtain any value from the interval $[0, 1]$. A sort of penalization technique is then used to drive the design towards a state where all design variables obtain a value close to 0 or 1.

Topological changes can be made possible also using so called level set parametrization [14]. This approach is based on implicit representation of the geometry through a scalar function $\phi(x)$ which is defined in a larger domain. The domain Ω is then defined to be that part of space where this scalar function obtains certain values, for example $\phi \geq 0$. The boundaries of Ω are thus defined by the level curves of the scalar function, and no explicit restrictions are placed on the topology of Ω.

In the rest of this thesis we shall not consider methods that allow topological changes, but we shall focus on optimizing shapes of domains with a fixed topology.

1.2 Discretization of the state problem

In general, the partial differential equations that constitute the state problem can not be solved analytically. Instead, numerical methods and computers must be utilized. In order to be solved with a computer, the problem must be discretized, i.e. converted into a form that is represented by a finite set of real variables. That is, the continuous function u is approximated with a function u_h, which is defined by a set of variables $\boldsymbol{q} = (q_1, \ldots, q_N)^T$ called the degrees of freedom. The infinite dimensional problem is thus converted into a finite dimensional one.

Common discretization methods include the finite element method (FEM) [15, 16], the finite volume method (FVM) [17], and the boundary element method (BEM) [18]. The methods are based on slightly different principles, and differ in the way they approximate the unknown function and seek a solution of the discrete problem.

To overcome the problem of mesh distortion during shape changes, and to avoid remeshing, so called meshfree or meshless methods [19] have been developed. In such methods, a division of the computation domain into elements is not required in order to define the basis functions. These methods are less sensitive to large shape changes than the finite element method, making them tempting for shape optimization purposes. Drawbacks of these methods include difficulties in imposing essential boundary conditions, and relatively high computational cost of the analysis [20].

In this thesis we shall focus on FEM and BEM discretizations. The basics of the finite element method will be introduced through a simple model problem in the next chapter, and the method is utilized in the articles [PII, PIII, PIV]. The articles [PI, PV, PVI] consider a boundary element type discretization.

1.3 Discretization of the geometry

Discretization of the optimization problem also includes representing the geometry as a function of a finite set of design variables $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)^T$. There are many ways to define the design variables and to reflect the geometrical changes in the computation. In the traditional finite element method, the computation domain is defined using a mesh, which is fitted to the boundaries of the geometry e.g. by placing the nodes on the boundary curve. Changes in the geometry are therefore reflected as changes in the mesh. These kind of techniques shall be utilized also in this thesis.

An interesting methodology that has emerged during recent years is the isogeometric analysis [21]. In this approach, the same spline functions that are used to define the geometry in many CAD systems are used also to approximate the unknown functions during the analysis. Therefore, unlike in the traditional finite element meshing, it is no longer necessary to construct e.g. a piecewise linear approximation of the spline based geometry, which is an obvious advantage. Instead, the geometry is exactly represented even in the coarsest discretization, and is preserved during refinements, which can be performed without interaction with the CAD system.

This approach has been successfully applied also in the context of shape optimization. In [22] two-dimensional structural shape optimization is considered, analytical sensitivities for the NURBS (Non-Uniform Rational B-Splines) discretizations are derived, and gradient based optimization methods are applied. In [23] continuous level sensitivity analysis is considered. Also in that context the isogeometric approach offers additional benefits. Namely, the NURBS basis functions naturally provide the design velocity fields needed in the sensitivity analysis process. Moreover, the normal vector and the curvature of the boundary can be calculated exactly.

The idea behind so called fictitious domain methods is to avoid doing most of the mesh related computations each time there are changes in the geometry. Instead one uses a regular mesh covering a larger domain that includes all admissible computation domains. Changes in the actual design domain are then reflected either by modifying the mesh only locally [24], or by keeping the mesh completely constant and forcing the appropriate boundary conditions using penalization [25] or Lagrange multipliers [26].

The choice of suitable design variables is obviously problem dependent. It is common to parameterize some parts of the shape using for example spline curves, and take the design variables to be the control points of the splines. In so called CAD free setting [27] the boundary nodes of the mesh are directly used as design variables. This way there is no need to know anything about the CAD model which produced the mesh used in the computation. However, the number of design variables can become very large. Furthermore, some form of smoothing may have to be applied to the shape variations in order to keep the geometry feasible.

1.4 Optimization methods

Once the problem is discretized, some optimization method can be utilized to solve the discrete optimization problem

$$\min_{\boldsymbol{\alpha} \in \boldsymbol{\alpha}^{ad}} J_h(\boldsymbol{u}_h(\boldsymbol{\alpha}), \boldsymbol{\alpha}, \boldsymbol{\beta}_h) \tag{3}$$

subject to

$$u_h = F(\Omega_h(\boldsymbol{\alpha}), \boldsymbol{\beta}_h). \tag{4}$$

Here u_h is the discrete approximation of the solution to the PDE, produced with the numerical method denoted by F, Ω_h is an approximation of the domain used in the solution process, $\boldsymbol{\beta}_h$ is discrete version of the input data, J_h is the numerically evaluated objective function, and $\boldsymbol{\alpha}^{ad}$ is the set of admissible design variable values.

There are many ways to solve the optimization problem (3). Usually the combined function $J_h : \alpha \mapsto J_h(\alpha)$ is non-linear, and the optimization process must be iterative. Optimization methods in general can roughly be divided into two groups: deterministic and stochastic methods.

Stochastic methods include for example simulated annealing [28] and genetic algorithms [29]. These methods include some amount of randomness in the optimization process. Reason for this is to avoid falling into the nearest local optimum in cases when the objective function has multiple local optima. A drawback is that these kind of methods often require a large amount of function evaluations, which is very costly when each function evaluation requires solving a partial differential equation.

Another class of methods are deterministic optimization methods, which do not include any randomness. Instead, in each iteration these methods do a local search and try to find a design that improves the value of the objective functional. Gradient based methods utilize the derivative of the objective functional to construct a search direction that guarantees an improvement in the objective. For some methods of this type, global convergence can be proven (see e.g. [30]). This means that the method will converge to a critical point from any initial guess. However, without additional assumptions there is no guarantee that the obtained solution is a global optimum. Some methods approximate the Hessian matrix of the objective functional, in which case one often speaks of quasi-Newton optimization methods. For some of these methods a superlinear convergence rate can be established.

Gradients can be exploited also in hybrid methods, which attempt to combine the best qualities of stochastic and deterministic methods. For example in [31] genetic algorithms are used to explore the search space, and a conjugate gradient algorithm is used to perform a local search, which guarantees that the final solution is at least a local optimum.

In this thesis we focus on gradient based methods. Most of this thesis is therefore devoted to techniques that allow the computation of the exact gradient of the discrete objective functional, i.e. so called consistent gradient.

2 MODEL PROBLEM I

Let us consider a classical shape optimization problem related to maximizing the torsional stiffness of a prismatic bar. Let the cross section of the bar $\Omega(\alpha)$ belong to the set of admissible shapes given by

$$U^{ad} = \left\{ (s,t) \in \mathbb{R}^2 \mid \frac{s^2}{\alpha^2} + \alpha^2 t^2 \leq 1 \right\}. \tag{5}$$

That is, $\Omega(\alpha)$ is an ellipse with semiaxis α and $1/\alpha$. Our state problem is as follows: we seek a scalar function $u : \Omega \to \mathbb{R}$ such that

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega, \end{cases} \tag{6}$$

where f is a constant force function $f \equiv 2$ in Ω.

We wish to find the design parameter α that maximizes the torsional rigidity of the bar, being equivalent to minimizing the functional

$$J = -\int_\Omega uf. \tag{7}$$

Because of the simplicity of the admissible shapes, analytical solution to the state problem can be found. It is given by

$$u(s,t) = \frac{\alpha^2}{\alpha^4+1} - \frac{1}{\alpha^4+1}s^2 - \frac{\alpha^4}{\alpha^4+1}t^2. \tag{8}$$

Moreover, the analytical expression for the objective functional J is

$$J(\alpha) = -\pi\frac{\alpha^2}{\alpha^4+1}. \tag{9}$$

2.1 Discretization

Finite element discretization of the model problem is based on the so called weak or variational formulation. We multiply the equation (6) with a test function ϕ,

integrate over the domain Ω, and perform partial integration, to obtain the problem:

$$\text{Find } u \in H_0^1(\Omega): \quad \int_\Omega \nabla u \cdot \nabla \phi - \underbrace{\int_{\partial\Omega} \nabla u \cdot \boldsymbol{n}\phi}_{=0} = \int_\Omega f\phi \quad \forall \phi \in H_0^1(\Omega). \tag{10}$$

Here the Sobolev space $H_0^1(\Omega)$ includes functions that vanish on $\partial\Omega$, are square integrable, and have square integrable weak first derivatives.

Discretization of the weak formulation then involves replacing u in (10) with an approximation u_h, which is defined as a linear combination of simple basis functions $\varphi_1, \ldots, \varphi_N \in V_N$:

$$u_h = \sum_{j=1}^{N} q_j \varphi_j, \tag{11}$$

where $V_N \subset H_0^1(\Omega)$ is a N-dimensional subspace. In the so called Galerkin's method one uses the same functions φ_i as the testing functions ϕ in (10). We now obtain

$$\int_\Omega \sum_{j=1}^{N} q_j \nabla \varphi_j \cdot \nabla \varphi_i = \int_\Omega f\varphi_i, \quad i = 1, \ldots, N, \tag{12}$$

which can be written as a system of linear equations of size $N \times N$:

$$\boldsymbol{A}\boldsymbol{q} = \boldsymbol{b} \text{ with } A_{ij} = \int_\Omega \nabla \varphi_j \cdot \nabla \varphi_i \text{ and } b_i = \int_\Omega f\varphi_i. \tag{13}$$

From this system the coefficients $\boldsymbol{q}$ can be solved to obtain the approximate solution u_h.

In the finite element method the basis functions are chosen so that they have a small support. To this end, one utilizes a mesh of the domain, like the one shown in Figure 2. The mesh is a collection of simple geometrical entities, such as triangles or quadrilaterals in 2D and tetrahedrons and hexahedrons in 3D, and defines a discrete approximation of the domain Ω. We denote this discretization by Ω_h, where h is a parameter characterizing the granularity of the discretization. Since curved boundaries can not be exactly represented with straight edges, the discrete domain is sometimes only an approximation of the true domain.

The standard linear basis functions are as follows. The basis functions are associated to nodes, which in this case coincide with the vertices of the mesh $\boldsymbol{x}_i = (x_i^1, \ldots, x_i^m)^T$, $i = 1, \ldots, N_n$. The basis functions are constructed so that they are continuous, linear in each element (triangle in the mesh), and in the nodes it holds that

$$\varphi_i(\boldsymbol{x}_j) = \begin{cases} 1, \text{ if } i = j \\ 0, \text{ if } i \neq j. \end{cases} \tag{14}$$

The support of one particular basis function is hilighted in Figure 2. It is clear from the expression given in (13) that the *system matrix* element A_{ij} is non-zero only if the supports of φ_i and φ_j overlap. From this choice of the basis and testing functions it follows that the resulting linear system of equations will be sparse, which is a characteristic feature of the finite element method.

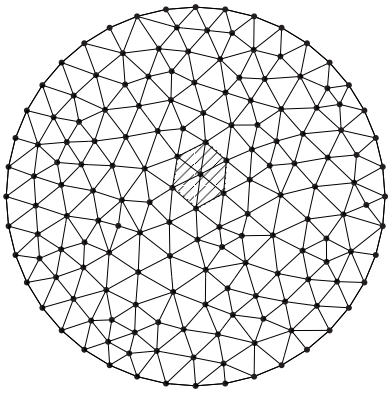

FIGURE 2 Mesh of the initial geometry.

2.2 Effects of remeshing

The type of mesh used in this example is called *unstructured*. This kind of meshes can be generated automatically by several methods using a *mesh generator* software [32, 33], given as input only the domain boundary. Thus we can automatize the evaluation of the functional (7): for a given design α we utilize the mesh generator to create a mesh approximating the corresponding domain $\Omega(\alpha)$, solve the state problem (6) approximately using the finite element method, and evaluate the value of the objective (7) using numerical integration. Some values obtained by this process and the corresponding values of the exact expression (9) are shown in Figure 3.

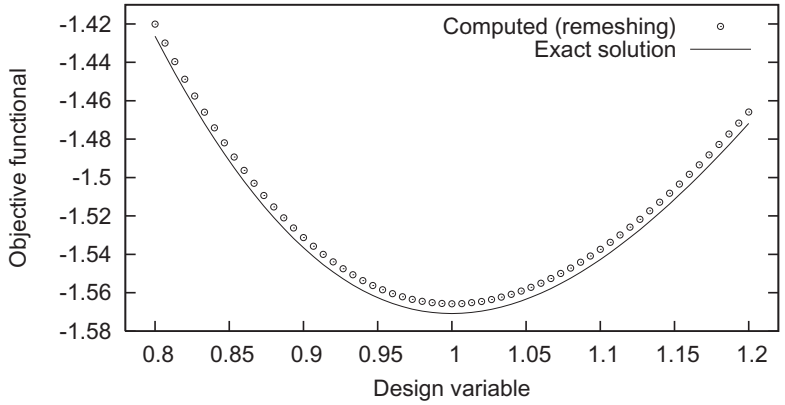

FIGURE 3 Solution of the model problem by completely regenerating the mesh for each design.

Based on the Figure 3, there is a small systematic error in the solution. This

can be explained as follows. It is known that the solution $u \in H_0^1(\Omega)$ of the problem (6) is the minimizer of the energy functional

$$u = \arg \min_{u \in H_0^1(\Omega)} \frac{1}{2} \int_\Omega \nabla u \cdot \nabla u - \int_\Omega f u = \arg \min_{u \in H_0^1(\Omega)} -\frac{1}{2} \int_\Omega f u. \tag{15}$$

The discrete solution, on the other hand, is the minimizer over the subspace V_N, and

$$\min_{v \in V_N} -\frac{1}{2} \int_\Omega f v \geq \min_{w \in H_0^1(\Omega)} -\frac{1}{2} \int_\Omega f w. \tag{16}$$

However, the location of the minimum of the objective functional seems to be quite close to the exact minimum. Moreover, the numerical solutions seem to draw a nice, continous looking curve as a function of the design. One could therefore be tempted to try for example some gradient based optimization method using the finite difference approximation to the derivative of the functional:

$$\frac{dJ_h}{d\alpha}(\alpha^0) \approx \frac{J_h(\alpha^0 + \epsilon) - J_h(\alpha^0)}{\epsilon}. \tag{17}$$

However, here one runs into trouble. Figure 4 shows the objective functional values obtained similarly than before, but over a smaller interval. It turns out that the approximation of J obtained using this process is not a continuous function of α, but instead it includes a significant amount of noise. Naturally, use of floating point arithmetic will always produce some amount noise, but the noise seen here is several orders of magnitude larger than that. The reason of this behaviour is as follows.

The mesh used in finite element computations consists of the nodal connectivity or topology $\boldsymbol{T}$, and nodal positions $\boldsymbol{X}$: $\Omega_h = \Omega_h(\boldsymbol{X}, \boldsymbol{T})$. Solving the same problem using two meshes, which represent exactly the same domain and have the same nodal coordinates but a different nodal connectivity, will usually result in slightly different solutions. Therefore also slightly different objective functional values are obtained. Similarly, using two meshes with the same topology but different internal nodal positions will also give two different solutions. All of this is basically caused by the discretization error.

Most unstructured mesh generators work by iteratively adding (and sometimes deleting) points to selected locations until some quality criteria are fulfilled [32]. Therefore, even a very small change in the boundary shape can cause a complete change in the mesh topology, and even a change in the number of nodes in the produced mesh. Such a change then causes a discontinuity in the objective functional.

The magnitude of the noise depends on the mesh density, and will tend to zero as the mesh is refined. However, in practical applications extremely dense meshes can not usually be used, since the computation times would become too long, especially as the computation has to be repeated several times during the optimization. Moreover, extremely dense meshes are often not required to obtain sufficient accuracy from the simulation point of view.

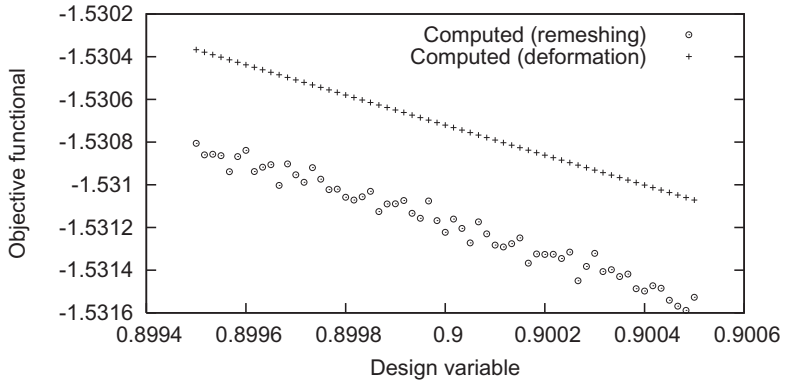

FIGURE 4 Solution of the model problem obtained by remeshing for each analysis, and by deforming a reference mesh.

The noisy behaviour of the objective functional has the following consequences. First of all, looking at Figure 4, it is evident that one should use a quite large value for the parameter ϵ in order to get a sensible approximation for the derivative using the formula (17). If the functional were highly non-linear, large values of ϵ would introduce a significant error in the approximation of the derivative.

Another problem is that gradient based optimization methods are usually designed to be used on continuous functions. Therefore, when applied to a function with a significant amount of noise, such optimization methods may stop at a point which is not very close to a local optimum.

Finally, remeshing the whole model for each objective functional evaluation can be quite time consuming. For these reasons, the following *mesh deformation* based approach is used in this thesis.

2.3 Mesh deformation approach

The idea of this approach is that the mesh topology is fixed, i.e. $\boldsymbol{T}(\boldsymbol{\alpha}) = \boldsymbol{T}(\boldsymbol{\alpha}_0)$, but the mesh nodal positions are allowed to move according to changes in the shape, i.e. $\boldsymbol{X}(\boldsymbol{\alpha}) \neq \boldsymbol{X}(\boldsymbol{\alpha}_0)$ in general. To this end we utilize a mesh deformation operator denoted formally by

$$\boldsymbol{G}(\boldsymbol{X}(\boldsymbol{\alpha}), \boldsymbol{\alpha}) = 0. \tag{18}$$

Usually the mesh deformation operator is constructed using a reference mesh, which is generated in the beginning of the optimization and corresponds to the initial design $\boldsymbol{\alpha}_0$. Many different approaches to deform the mesh have been developed, each having different relative merits in terms of the ease of implementation, quality of the resulting mesh, the amount of deformation that can be tolerated, and the computational cost.

Some methods perform the movement of the mesh nodes in iterative fashion, relocating each node based on the locations of its neighbours. A classical method of this type is based on a technique called Laplacian smoothing [34], and it involves defining the movement of each node as the average of the movement of its neighbours. This approach has resulted in many variants, which try to improve the robustness of the method and the quality of the resulting mesh. See [35] and [36] for examples.

Another possibility is to model the mesh movement using partial differential equations. This is not completely different from the previous approach: under certain assumptions the basic method of Laplacian smoothing is nothing more than iterative solution method for the Laplace equation. Other popular PDEs are the linear elasticity model [37] and the biharmonic equation [38].

Radial basis function interpolation has been recently introduced in the field of fluid structure interaction problems [39] and aerodynamic shape optimization [40]. This deformation technique is independent on the mesh type and topology, relatively straightforward to implement, and it is not computationally expensive.

In case of simple geometries the mesh deformation operator can sometimes be defined analytically. For example, let the reference mesh related to our model problem correspond to initial design α_0, and $\tilde{x}_i^j$ be the j^{th} coordinate of i^{th} node in that mesh. We can now define

$$\begin{cases} G_i = x_i^1 - (\alpha/\alpha_0)\,\tilde{x}_i^1 \\ G_{i+N_n} = x_i^2 - (\alpha_0/\alpha)\,\tilde{x}_i^2 \end{cases} \tag{19}$$

where $i = 1, \ldots, N_n$.

Besides avoiding the need of remeshing, use of the mesh deformation technique has the following benefit. Under the assumption of fixed mesh topology and continuous dependence of the nodal positions on the design variables, it can be shown (see e.g. [41]) that the mappings $\boldsymbol{\alpha} \mapsto \boldsymbol{A}(\alpha)$, $\boldsymbol{\alpha} \mapsto \boldsymbol{b}(\boldsymbol{\alpha})$ are continuous, and thus $\boldsymbol{\alpha} \mapsto \boldsymbol{q}(\boldsymbol{\alpha})$ is continuous. Moreover, assuming that J_h is continuous, the composite mapping $\boldsymbol{\alpha} \mapsto J_h(\boldsymbol{\alpha})$ is continuous.

In Figure 4 are shown the objective functional values obtained using an analytical mesh deformation approach and remeshing for each analysis. There is a small difference between the results obtained using the two approaches, which may result from the fact that the mesh was generated corresponding to the design $\alpha_0 = 1.2$, and relatively large deformation of the mesh was thus taking place. The meshes that are regenerated for each analysis include more regularly shaped elements, and may also include a different number of nodes and elements. However, the functional obtained using the mesh deformation approach is continuous, except for the very small scale noise caused by the use of floating point arithmetic.

In what follows, the discrete objective functional J_h is taken to mean the function that arises when the mesh topology is fixed to that of the reference mesh, and the mesh nodes are relocated according to some given mesh deformation operator $\boldsymbol{G}$. In practise, however, remeshing is still sometimes required, but we will consider these as cases of *restarting* the optimization with an essentially different objective functional (see e.g. article [PIV]).

3 SENSITIVITY ANALYSIS

The sensitivity analysis of optimization problems governed by partial differential equations can be performed in many ways. In the following the main features of different approaches will be briefly introduced. For a rigorous mathematical presentation see e.g. [42] in the continuous and [43] in the discrete case.

3.1 Continuous sensitivity equation method

In the continuous sensitivity equation method the differentiation is performed on the continuous level to obtain so called sensitivity equations, from which the sensitivities of the solution with respect to the design variables can be solved. The sensitivity expressions need to be derived separately for each problem, but there are some generic results that can be utilized (see e.g. [41], [42]).

The equations can be derived by considering a shape deformation mapping $\boldsymbol{x} \mapsto \boldsymbol{x} + t\mathcal{V}(\boldsymbol{x})$, $\Omega \mapsto \Omega_t$ and defining u_t to be the solution to the state problem which is posed in the deformed domain Ω_t:

$$u_t = u(t, \boldsymbol{x} + t\mathcal{V}(\boldsymbol{x})). \tag{20}$$

The resulting derivatives can be seen as directional derivatives characterizing the behaviour of the objective when Ω moves in the direction given by $\mathcal{V}$. Connection to more general shape changes can be established by noticing that if a point $\boldsymbol{x} \in \Omega$ moves according to changes in the design $\boldsymbol{\alpha}$, position of $\boldsymbol{x}$ is locally approximated by

$$\boldsymbol{x}(\boldsymbol{\alpha}_0 + te^i) \approx \boldsymbol{x}(\boldsymbol{\alpha}_0) + t \underbrace{\frac{\partial \boldsymbol{x}}{\partial \alpha_i}}_{:=\mathcal{V}}. \tag{21}$$

Assuming necessary smoothness, the chain rule can be applied to obtain

$$\dot{u}(0, \boldsymbol{x}) := \frac{d}{dt} u(t, \boldsymbol{x} + t\mathcal{V}(\boldsymbol{x}))\Big|_{t=0} = \frac{\partial}{\partial t} u(0, \boldsymbol{x}) + \nabla_{\boldsymbol{x}} u(0, \boldsymbol{x}) \cdot \mathcal{V}, \tag{22}$$

or, in abbreviated form,

$$\dot{u} = u' + \nabla_x u \cdot \mathcal{V}. \tag{23}$$

The term $\dot{u}$ is called the material derivative, and u' is called the pointwise shape derivative.

The shape derivative u' is obtained by differentiating the state problem with respect to t at $t = 0$. In case of our model problem the shape derivative is obtained as solution to a boundary value problem [41]

$$\begin{cases} -\Delta u' = 0 & \text{in } \Omega \\ u' = -(\nabla u \cdot \bar{n})\mathcal{V} \cdot \bar{n} & \text{on } \partial\Omega. \end{cases} \tag{24}$$

The velocity depends on the shape parametrization. In this case it is given by $\mathcal{V}(x^1, x^2) = (x^1/\alpha, -x^2/\alpha)$.

Alternatively, we may compute the material derivative of u. It is obtained by solving [41]

$$\begin{cases} -\Delta \dot{u} = \nabla \cdot (f\mathcal{V} + \mathcal{A}\nabla u) & \text{in } \Omega \\ \dot{u} = 0 & \text{on } \partial\Omega, \end{cases} \tag{25}$$

where [44]

$$\mathcal{A} = (\nabla \cdot \mathcal{V})\mathrm{id} - (D\mathcal{V})^T - D\mathcal{V}, \tag{26}$$

and $D\mathcal{V}$ denotes the Jacobian of $\mathcal{V}$.

The objective functional must also be differentiated with respect to t. For an integral type objective functional

$$J_t = \int_{\Omega_t} g(u_t)\,\mathrm{d}x_t, \tag{27}$$

the derivative $\dot{J}$ is given by [41]

$$\dot{J} = \frac{d}{dt} J_t \bigg|_{t=0+} = \int_\Omega \frac{\partial g}{\partial u}(u) u' \,\mathrm{d}x + \int_{\partial\Omega} g(u)(\mathcal{V} \cdot \bar{n})\,\mathrm{d}s. \tag{28}$$

Alternative expression utilizing the material derivative is given by

$$\dot{J} = \int_\Omega \frac{\partial g}{\partial u}(u)\dot{u}\,\mathrm{d}x + \int_\Omega g(u)\nabla \cdot \mathcal{V}\,\mathrm{d}x. \tag{29}$$

In case of the model objective functional (7) we have $g(u) = -fu$, and therefore $\partial g/\partial u = -f \equiv -2$. Moreover, $g(u) = 0$ on $\partial\Omega$, and $\nabla \cdot \mathcal{V} = 0$. Thus we obtain

$$\dot{J} = -2\int_\Omega u'\,\mathrm{d}x = -2\int_\Omega \dot{u}\,\mathrm{d}x. \tag{30}$$

We now have the ingredients to perform the shape sensitivity analysis. After solving the state problem, we solve the sensitivity equations for the shape derivative related to each α_i separately, and use the expression (30) to compute the gradient of the objective functional. In this approach, M different PDEs have to be solved, since the shape derivatives will in general be different for each α_i. A remedy to avoid this is to use the following technique.

3.2 Continuous adjoint method

Let λ satisfy the following adjoint equation:

$$\begin{cases} -\Delta\lambda = \frac{\partial g}{\partial u}(u) & \text{in } \Omega \\ \lambda = 0 & \text{on } \partial\Omega. \end{cases} \tag{31}$$

Now, the expression (28) can be written as

$$\dot{J} = -\int_\Omega \Delta\lambda u' \,\mathrm{d}x + \int_{\partial\Omega} g(u)(\mathcal{V}\cdot\bar{n})\,\mathrm{d}s. \tag{32}$$

Using integration by parts twice we obtain

$$\dot{J} = -\underbrace{\int_\Omega \lambda\Delta u' \,\mathrm{d}x}_{=0} + \underbrace{\int_{\partial\Omega} \lambda\nabla u'\cdot\bar{n}\,\mathrm{d}s}_{=0} - \int_{\partial\Omega} u'\nabla\lambda\cdot\bar{n}\,\mathrm{d}s + \int_{\partial\Omega} g(u)(\mathcal{V}\cdot\bar{n})\,\mathrm{d}s \tag{33}$$

where the first two terms are zero since from (24) we know that $\Delta u' = 0$ in Ω, and in (31) we specified that $\lambda = 0$ on $\partial\Omega$.

In particular, all terms involving the sensitivity u' inside the domain Ω are eliminated, and one does not have to solve the sensitivity equations any more. Plugging in the term u' on the boundary $\partial\Omega$ as given in (24), we obtain

$$\dot{J} = \int_{\partial\Omega} (\nabla u\cdot\bar{n})(\nabla\lambda\cdot\bar{n})(\mathcal{V}\cdot\bar{n})\,\mathrm{d}s + \int_{\partial\Omega} g(u)(\mathcal{V}\cdot\bar{n})\,\mathrm{d}s \tag{34}$$

Since in the case of our model problem $\partial g/\partial u = -f$, we notice that the adjoint problem (31) is exactly the same as the state problem (6), except that the right hand side has the opposite sign. Thus it follows that $\lambda = -u$, and no additional PDEs must be solved. This is a well known property of objective functionals that are defined as an integral of the solution times the force vector over the computation domain. Again, $g(u) = 0$ on $\partial\Omega$, and the sensitivity is given by

$$\dot{J} = -\int_{\partial\Omega} (\nabla u\cdot\bar{n})^2(\mathcal{V}\cdot\bar{n})\,\mathrm{d}s. \tag{35}$$

In the general case one actually has to solve the adjoint PDE, but the advantage is that there is only one adjoint function λ regardless on the number of design variables.

Similarly, starting from the expression (29) and making use of (25), we obtain

$$\dot{J} = \int_\Omega \lambda\nabla\cdot(f\mathcal{V} + \mathcal{A}\nabla u) + \int_\Omega g(u)\nabla\cdot\mathcal{V}\,\mathrm{d}x. \tag{36}$$

Noticing that $\nabla\cdot\mathcal{V} = 0$ in this case, integrating by parts once more, and plugging in $\lambda = -u$, we obtain

$$\dot{J} = \int_\Omega \mathcal{A}\nabla u\cdot\nabla u\,\mathrm{d}x. \tag{37}$$

3.3 Direct discrete sensitivity analysis

For brevity of notation, we write the discretization of the state problem in the so called residual form:

$$\boldsymbol{r}(\boldsymbol{q}, \boldsymbol{X}(\boldsymbol{\alpha})) = 0. \tag{38}$$

In case of our linear model problem, the residual is simply $\boldsymbol{r} = \boldsymbol{A}\boldsymbol{q} - \boldsymbol{b}$, and the condition $\boldsymbol{r} = 0$ means that $\boldsymbol{q}$ must be the solution to the linear system (13). In case of non-linear problems the system matrix and/or the right hand side depend on the solution vector $\boldsymbol{q}$, and driving the residual to zero often requires using some iterative method. However, the differentiation procedure presented in what follows applies also in this case.

The notation $\boldsymbol{X}(\boldsymbol{\alpha})$ is used to emphasize that in this case $\boldsymbol{\alpha}$ are geometrical design variables, and therefore affect the discretization only through the mesh nodal positions. The unknowns $\boldsymbol{q}$ also depend on $\boldsymbol{\alpha}$, but this dependence is implicit through the relation (38). To differentiate this dependence we apply implicit differentiation to the equation (38) and rearrange the terms to obtain

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \alpha_i} = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{X}} \frac{\partial \boldsymbol{X}}{\partial \alpha_i} \tag{39}$$

We have now obtained a linear system, from which the sensitivities $\partial \boldsymbol{q}/\partial \alpha_i$ can be solved and used to compute the derivatives of J_h:

$$\frac{dJ_h}{d\alpha_i} = \frac{\partial J_h}{\partial \boldsymbol{X}} \frac{\partial \boldsymbol{X}}{\partial \alpha_i} + \frac{\partial J_h}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \alpha_i}. \tag{40}$$

Here $\partial J_h/\partial \boldsymbol{X}$ denotes the direct dependence of J_h on the geometry, which exists for example if J is given in the form of an integral, and the domain of integration is affected by the design variables.

Since there are M design variables, this approach requires the solution of M linear systems. This is in analogy with the continuous sensitivity equation method. As in the continuous case, the adjoint approach can be used to overcome this problem.

3.4 Discrete adjoint approach

The discrete adjoint approach is based on the following algebraic manipulations. Since the residual $\boldsymbol{r}(\boldsymbol{q}, \boldsymbol{X}(\boldsymbol{\alpha}))$ is required to remain zero regardless of changes in the shape, J_h coincides with the Lagrangian

$$L(\boldsymbol{q}, \boldsymbol{X}(\boldsymbol{\alpha})) = J_h(\boldsymbol{q}, \boldsymbol{X}(\boldsymbol{\alpha})) + \lambda^T \boldsymbol{r}(\boldsymbol{q}, \boldsymbol{X}(\boldsymbol{\alpha})), \tag{41}$$

where λ is an arbitrary vector of Lagrange multipliers.

Differentiating this relation with respect to α_i and rearranging the terms we obtain

$$\frac{dL}{d\alpha_i} = \frac{\partial J_h}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \alpha_i} + \lambda^T \left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \alpha_i} \right) + \left(\frac{\partial J_h}{\partial \boldsymbol{q}} + \lambda^T \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \right) \frac{\partial \boldsymbol{q}}{\partial \alpha_i}. \tag{42}$$

Since the Lagrangian multiplier is arbitrary, we can choose it to satisfy the adjoint equation

$$\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \right)^T \lambda = -\frac{\partial J_h}{\partial \boldsymbol{q}}^T \tag{43}$$

This choice kills the last term in (42), and the sensitivity can be computed from

$$\frac{dL}{d\alpha_i} = \frac{\partial J_h}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \alpha_i} + \lambda^T \left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \alpha_i} \right). \tag{44}$$

3.5 "Doubly adjoint" discrete approach

In the direct discrete and discrete adjoint methods presented earlier, the mesh nodal sensitivities $\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\alpha}}$ are assumed to be known. They can of course be computed by applying implicit differentiation to the mesh deformation equation (18), and solving the resulting linear systems. However, in some cases the mesh deformation operator is itself a PDE, which makes this procedure quite inefficient. The following "doubly adjoint" approach can be used instead.

Following Nielsen and Park [45] we define a Lagrangian function

$$L(\boldsymbol{q}, \boldsymbol{X}(\boldsymbol{\alpha})) = J_h(\boldsymbol{q}, \boldsymbol{X}(\boldsymbol{\alpha})) + \lambda_s^T \boldsymbol{r}(\boldsymbol{q}, \boldsymbol{X}(\boldsymbol{\alpha})) + \lambda_m^T \boldsymbol{G}(\boldsymbol{\alpha}, \boldsymbol{X}(\boldsymbol{\alpha})) \tag{45}$$

Differentiation with respect to α_i yields

$$\frac{dL}{d\alpha_i} = \frac{\partial J_h}{\partial \boldsymbol{q}}\frac{\partial \boldsymbol{q}}{\partial \alpha_i} + \frac{\partial J_h}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \alpha_i} + \lambda_s^T \left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}}\frac{\partial \boldsymbol{q}}{\partial \alpha_i} + \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \alpha_i} \right) + \lambda_m^T \left(\frac{\partial \boldsymbol{G}}{\partial \alpha_i} + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}\frac{\partial \boldsymbol{X}}{\partial \alpha_i} \right) \tag{46}$$

Rearranging this we obtain

$$\frac{dL}{d\alpha_i} = \left(\frac{\partial J_h}{\partial \boldsymbol{q}} + \lambda_s^T \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \right) \frac{\partial \boldsymbol{q}}{\partial \alpha_i} + \left(\frac{\partial J_h}{\partial \boldsymbol{X}} + \lambda_s^T \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{X}} + \lambda_m^T \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}} \right) \frac{\partial \boldsymbol{X}}{\partial \alpha_i} + \lambda_m^T \frac{\partial \boldsymbol{G}}{\partial \alpha_i} \tag{47}$$

Now the derivative can be obtained from

$$\frac{dL}{d\alpha_i} = \lambda_m^T \frac{\partial \boldsymbol{G}}{\partial \alpha_i} \tag{48}$$

where λ_m satisfies

$$\left(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}} \right)^T \lambda_m = -\left(\frac{\partial J_h}{\partial \boldsymbol{X}}^T + \frac{\partial r}{\partial \boldsymbol{X}}^T \lambda_s \right) \tag{49}$$

and λ_s satisfies

$$\left(\frac{\partial r}{\partial \boldsymbol{q}} \right)^T \lambda_s = -\frac{\partial J_h}{\partial \boldsymbol{q}}^T. \tag{50}$$

In this formulation two adjoint systems need to be solved. Right hand side of (49) depends on λ_s, and therefore the system (50) must be solved first. Both adjoint vectors are independent on $\boldsymbol{\alpha}$, and the systems therefore have to be solved only once regardless of the number of design variables.

3.6 Concluding remarks

Applicability of the different sensitivity analysis methods has been the subject of much research and even debate. For survey papers on sensitivity analysis see [46, 47, 48, 49].

3.6.1 Accuracy and consistency

Following [48] we make a distinction between accuracy and consistency of gradients. By *accuracy* we mean the difference between the derivative obtained using the employed sensitivity analysis method and the exact derivative based on the exact solution of the governing state equations. Accurate derivatives are obviously desirable, but the accuracy is dependent on the discretization. That is, density of the mesh dictates the accuracy of the sensitivity analysis as well as the primal analysis, and accurate derivatives are in general obtained only as the element size tends to zero. *Consistency*, on the other hand, is defined as the difference between the computed derivatives and the exact derivative of the numerical model.

The continuous level sensitivity expressions in general depend on the solution of the state problem, and are therefore exact only assuming we know the exact solution. Since it is usually not known, the numerical approximation must be used instead. Depending on which formulation is used, the resulting derivatives may or may not be consistent with the derivatives of the numerical model. This approach is sometimes called computing the inexact gradient of the exact objective functional, or in other terminology, the differentiate-then-discretize approach.

In [50, 51] examples are given, where the optimization terminates to a point which is not very close to the optimum due to the inconsistency in the gradient obtained using the continuous level sensitivity analysis. This can happen because the optimizer uses the supplied gradient information to choose the search direction so that it should be a direction of descent, but due to the inconsistency it may fail to be the case. Thus the optimizer can get confused and stop the optimization.

On the other hand, authors of the review [49] do not consider the potential inconsistency a major problem, because the quality of the gradients is often sufficient to achieve convergence relatively close to the optimum. For example in [52] continuous and discrete level gradients were compared in the context of aerodynamic optimization. The gradients were found to be in good agreement, and similar convergence in the optimization was observed using both approaches.

The partial derivatives appearing in the discrete level formulas can be computed exactly (up to numerical precision), and consistent derivatives can thus be obtained. The discrete level derivatives are therefore sometimes called exact derivatives of the inexact objective functional. Term discretize-then-differentiate is also sometimes used. Obvious advantage of the consistency is that the optimization is able to converge to the exact local optimum of the discrete model.

Let us consider the sensitivity analysis of the Model problem I. We shall use the following abbreviations for the sensitivity analysis methods:

- HOFD means differentiation of objective using a response level finite difference formula

$$J_h'(\alpha) \approx \left(J_h(\alpha - 2\epsilon) - 8J_h(\alpha - \epsilon) + 8J_h(\alpha + \epsilon) - J_h(\alpha + 2\epsilon)\right) / (12\epsilon). \quad (51)$$

  The truncation error using this formula is of the order $\mathcal{O}(\epsilon^4)$. The step length $\epsilon = 10^{-4}$ was used, which provides a very low truncation error without significant round off errors. This method is used as a reference method to compute the "exact" derivative of the numerical model.

- CSE1 is the continuous sensitivity equation method based on the shape derivative u' obtained by solving (24).

- CSE2 is the continuous sensitivity equation method based on the material derivative $\dot{u}$ obtained by solving (25).

- CA1 is the continuous adjoint method based on the expression (35).

- CA2 is the continuous adjoint method based on the expression (37).

- AD is the discrete adjoint method (Section 3.4), where automatic differentiation is used to compute the terms $\partial J_h/\partial \boldsymbol{q}$, $\partial J_h/\partial \boldsymbol{\alpha}$, and $\partial \boldsymbol{r}/\partial \boldsymbol{\alpha}$ exactly. More details on the implementation of this method are given in Section 4.3.

The other discrete level methods were not included in the test, because they are known to produce the same results as we solve the linear systems with high precision and use accurate partial derivatives.

In Figure 5 is presented the results of the sensitivity analysis of the model problem performed with the different methods at $\alpha = 1.5$. Analytical value of the derivative at this point was $\partial J/\partial \alpha \approx 1.0417$. The Figure shows the difference between the analytical derivative and the numerical approximation as a function of characteristic element size h.

In this case the CSE2, CA2, and AD methods coincide: the maximal difference between the derivatives obtained using these methods was of the order 10^{-14}. Moreover, the derivatives produced using these methods are consistent: the maximal difference between them and the HOFD reference gradient was of the order 10^{-10}. Therefore these results are plotted using only a single curve in the Figure 5.

As we can expect, all the methods produce more and more accurate derivatives as the element size decreases. However, a different convergence rate is observed for some methods. Using a least squares fit of a linear model to the data presented in the Figure, the errors are found to depend on h as $\mathcal{O}(h^{1.00})$, $\mathcal{O}(h^{0.99})$, and $\mathcal{O}(h^{2.00})$ in case of the CSE1, CA1, and HOFD methods, respectively. This is due to a particular problem pointed out e.g. in [53]. Namely, gradients of the solution and/or the adjoint variable evaluated at the boundary are required in the sensitivity formulas, but such gradients are poorly captured using linear finite elements.

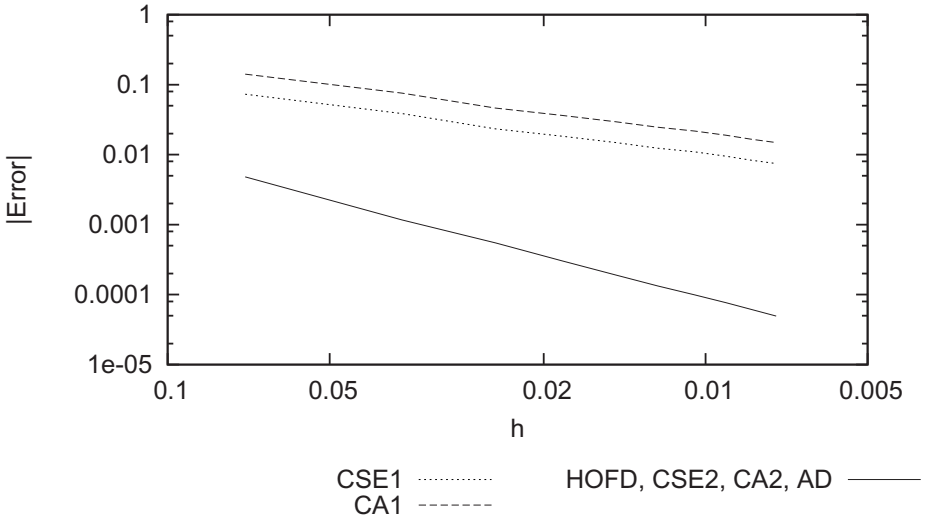

FIGURE 5 Absolute value of the error.

3.6.2 Implementation effort

While the continuous level methods are relatively well established for traditional fields of application, such as structural and fluid mechanics, new areas of application and multidisciplinary design cases remain a problem. For example, turbulence models are still often "frozen", i.e. neglected from differentiation, to simplify calculations [49]. Moreover, there are some limitations on what can be used as objective functions so that the adjoint problem is well posed [54, 49].

On the other hand, manually differentiating complex simulation codes to obtain discrete level derivatives is sometimes next to impossible [9]. Moreover, as pointed out in [27], even the setting of the optimization problem is often an iterative process: the objective functional and/or the constraints may have to be modified as the solution is found to be infeasible in some, unforeseen, sense. This makes manual differentiation of simulation codes even more impractical, as the sensitivities would have to be reprogrammed after each change to the simulation code or the optimization problem.

However, manual labour can be kept to minimum by using *automatic differentiation*. Automatic differentiation is the topic of the article [PI], and is not therefore discussed in detail here. Automatic differentiation computes the derivatives of a computer code exactly up to floating point precision, resulting in perfectly consistent derivatives.

3.6.3 Computational complexity

In general, if the number of objective functionals and solution dependent constraints is lower than the number of design variables, adjoint approaches (contin-

uous and discrete) are more efficient than the direct ones.

Difference in the computational cost between continuous and discrete approaches appears to be problem and implementation dependent [48]. In some cases the continuous adjoint method can be very efficient. For example, due to the self-adjoint nature of our model problem, the gradient could be obtained by simply computing one boundary integral per design variable. In the discrete case computation of the partial derivatives of the residual with respect to the design variables is required. The boundary layer concept is sometimes utilized to improve the efficiency of the discrete methods [55].

A common notion about automatic differentiation is that computational cost of differentiating a single objective functional with respect to M design variables is of the order of M times the original solution time when so called *forward mode* AD is used, whereas so called *reverse mode* is a way of obtaining the gradient so that computation time is independent on the number of design variables. The latter is true in theory, but the problem is that information about the whole computation has to be stored to a so called tape, size of which can easily become very large [56]. Thus the tape needs to written to a disk, which is very slow compared to memory access, and can lead to significant decrease in computational performance (see [PI]). To avoid this problem, significant changes to the solver code may be needed [57, 58], making this approach less automatic.

An often overlooked possibility is to apply forward mode AD in the context of the discrete level adjoint formulas. Let us look at the complexity of computing the required partial derivatives in the discrete adjoint formulas of Section 3.4 in a situation where the sensitivities $\partial \boldsymbol{X}/\partial \boldsymbol{\alpha}$ are easy to compute. This is the case for example if the geometry is defined using an analytical mapping, or if the mesh deformation operator is computationally inexpensive (see article [PVI] for an example of such a situation).

The term $\partial \boldsymbol{r}/\partial \boldsymbol{q}$ is often already available, since it is utilized while solving the state problem using Newton iteration. The terms $\partial J_h/\partial \boldsymbol{q}$ and $\partial J_h/\partial \boldsymbol{\alpha}$ are often quite cheap to compute, but care must be taken when AD is applied (see the section about partially separable functions in article [PI]). The term $\partial \boldsymbol{r}/\partial \boldsymbol{\alpha}$ is the one that is often the most complicated to compute, since it requires differentiating the whole residual with respect to all design variables, and the computational cost may therefore be of the order of M times the original assembly time. However, we can write $\partial \boldsymbol{r}/\partial \boldsymbol{\alpha} = (\partial \boldsymbol{r}/\partial \boldsymbol{X})(\partial \boldsymbol{X}/\partial \boldsymbol{\alpha})$. Here the Jacobian $\partial \boldsymbol{r}/\partial \boldsymbol{X}$ is extremely sparse, since each residual component is affected only by the shape of the elements that belong to the support of the associated testing function. Now we only have to compute this sparse Jacobian, and the gradient with respect to $\boldsymbol{\alpha}$ is obtained by a simple multiplication.

This can be implemented for example by applying the dense forward mode AD only to local contributions, i.e. differentiating the residual in each element with respect to the nodal coordinates of that particular element. However, this approach requires the mapping of indices to assemble these local contributions to the global Jacobian and/or gradient storages. This is done automatically for example in Numerrin 2.0 finite element software [59].

The approach taken in this thesis is even more general: exploitation of the sparsity is on the responsibility of the automatic differentiation implementation, as we use the *sparse forward mode* AD [60]. Our implementation of the method is described in the article [PI]. This approach offers maximum flexibility to experiment with things that are in some sense non-standard, since the automatic differentiation is not in any special way related to the finite element method. Indeed, the same implementation of automatic differentiation can be easily applied to existing simulation codes (see article [PV]). We return to the exploitation of the sparsity using this technique through a numerical example in Section 5.3.

If the mesh deformation operator is expensive, making it impractical to compute the derivatives $\partial \boldsymbol{X}/\partial \boldsymbol{\alpha}$, the technique discussed in the Section 3.5 may be used. In this method $\partial \boldsymbol{X}/\partial \boldsymbol{\alpha}$ is never explicitly computed. Instead one must compute $\partial \boldsymbol{G}/\partial \boldsymbol{X}$, which e.g. in the case of PDE based mesh deformation possesses similar sparsity properties than $\partial \boldsymbol{r}/\partial \boldsymbol{X}$, and solve one additional adjoint system. The dependence of $\boldsymbol{G}$ on $\boldsymbol{\alpha}$ is usually through boundary conditions, which means that the computation of $\partial \boldsymbol{G}/\partial \boldsymbol{\alpha}$ is not complicated. The computational cost of obtaining the gradient using this approach is essentially independent on the number of design variables.

3.6.4 Options for commercial solvers

Adding sensitivity analysis capabilities to existing commercial solvers would often be very tempting. There are a few ways for doing this, but certain properties are required from the solver. If the source code of the solver is available, applying automatic differentiation may be feasible also in case of solvers of commercial complexity. For example in [61] ADIFOR tool is used to differentiate FLUENT fluid dynamics solver. However, source codes of commercial solvers are usually not available.

Finite differences in the objective functional value level are in principle always possible. However, as seen in the case of our example problem, the objective functional can be noisy. This means that relatively large steps must be used, resulting in poor accuracy of the derivatives. Accuracy can be improved by using higher order finite difference formulas, but this also significantly increases the computational cost.

If there is a way to adapt the mesh to geometrical changes while keeping the mesh topology constant, and if the system matrix and right hand side vector are accessible, finite differences can be utilized to approximate the partial derivatives required in the discrete adjoint method. For examples of this approach in electromagnetic shape optimization see [62].

Finally, the continuous level methods may also be employed. To do this the solver must be able to solve the sensitivity or adjoint equations, which may or may not be similar to the original problem. For an example of this approach in the context of magnetostatics see [63]. Problems related to structural mechanics are considered in [64].

4 FINITE ELEMENT LIBRARY

In this Chapter we describe our finite element library, which is used to perform the shape optimizations described in the publications [PII, PIII, PIV]. The purpose of the library is to enable quick experimentation with different problem formulations and numerical techniques. The finite element routines are quite standard textbook implementations following mostly [65].

The library is written in C++ programming language, and utilizes external state of the art open source codes when possible:

- Gmsh [1] [66] is used to generate meshes.
- Boost [2] is used for sparse matrix storage.
- Interfaces to SuperLU [3] [67] to solve sparse linear systems.

Currently supported element types are a segment and a triangle, both of which are available as linear and quadratic versions. Adding more Lagrange elements in any dimension is straightforward. Isoparametric element mappings are used, and numerical integrations are performed in the reference elements. To this end, the library enables the evaluation of the basis functions and their gradients with respect to the global coordinates in the quadrature points.

4.1 System matrix assembly

Let us look at the solution of the model state problem using the library. We loop over the elements of the mesh and the quadrature points of each element. At each quadrature point, the gradients of the basis functions are evaluated using the syntax

[1] Available at http://www.geuz.org/gmsh/
[2] Available at http://www.boost.org/
[3] Available at http://crd.lbl.gov/∼xiaoye/SuperLU/

```
elem->evalBasisGrad(l, gbas);
elem->evalBasisGrad(k, gphi);
```

where l and k are local indices. The contribution of $\nabla\varphi_j \cdot \nabla\varphi_i$ from this element can then be added to the matrix A (which is a sparse matrix provided by the Boost library) as follows:

```
A(i,j)+=(gbas[0]*gphi[0]+gbas[1]*gphi[1])*elem->dx();
```

Similarly, the value of the test function is evaluated to the variable `phi`, and a contribution is added to the right hand side vector:

```
b(i)+=2.0*phi*elem->dx();
```

The effects of the quadrature weight and the Jacobian determinant of the transformation from local to global coordinates are included in the function `dx()`.

While this is sufficient for solving linear state problems, non-linear problems and shape optimization can benefit from a slightly different approach, which permits the use of automatic differentiation.

4.2 Residual assembly

In Section 3.3 we wrote the state problem in the residual formulation (38) for brevity of notation. As we will see, this formulation is convenient also from the programming point of view, especially when the sensitivity analysis phase is involved.

The Newton iteration to solve the set of algebraic equations (38) can be written as

$$\boldsymbol{q}^{k+1} = \boldsymbol{q}^k - \left(\frac{\partial \boldsymbol{r}(\boldsymbol{q}^k)}{\partial \boldsymbol{q}} \right)^{-1} \boldsymbol{r}(\boldsymbol{q}^k). \tag{52}$$

Here $\boldsymbol{q}^k$ is an approximation to the vector of unknowns at iteration k. Computation of the required Jacobian $\partial \boldsymbol{r}/\partial \boldsymbol{q}$ can be easily implemented as follows. We first create a vector `q` containing the degrees of freedom, and define the components of the vector to be the independent variables from the AD point of view. These variables, and all the variables depending on them, must be represented using the `addouble` type (see articles [PI, PV] for more information). During the assembly phase, we first compute ∇u_h

```
gu[0] +=  gbas[0]*q(i);
gu[1] +=  gbas[1]*q(i);
```

looping over all basis functions in this element. Here `gbas` contains the gradient of the basis function, obtained using `elem.evalBasisGrad()` like previously. Next we evaluate the value and the gradient of each test function, and add the contribution to the residual vector:

```
r(i) += (gu[0]*gphi[0]+gu[1]*gphi[1]-2.0*phi)*elem.dx();
```

Notice the similarity of this piece of code and the mathematical expression (10) of the weak form of the problem.

Here `r` is a *dense vector* containing `addouble` type variables. However, as explained in the article [PI], each `addouble` computes and stores only non-zero partial derivatives. In this case, the variables `gu[0]` and `gu[1]` only have non-zero partial derivatives with respect to a few degrees of freedom. The same holds true for each component of residual, e.g. each element in the vector `r`. Thus the sparsity of the Jacobian $\partial \boldsymbol{r}/\partial \boldsymbol{q}$ is exploited as usually in the finite element method.

We use exactly the same finite element routines than previously, such as `elem.evalBasisGrad()` and `elem.dx()`. The fundamental difference to the previous approach is that we do not assemble the system matrix and the right hand side vector separately. Instead, we only assemble the residual vector, and let the underlying automatic differentiation engine compute and store the partial derivatives with respect to the degrees of freedom $\boldsymbol{q}$. The Jacobian matrix can then be extracted and used to solve the linear system related to the Newton step (52).

In case of the linear model problem considered here, the Jacobian is exactly the same matrix $\boldsymbol{A}$ that is obtained using the previous method, the residual $\boldsymbol{r} = -\boldsymbol{b}$, and the Newton iteration will converge with just one step. Thus in this case, there is not much difference between these approaches from the point of view of solving the state problem. However, this approach is convenient from the shape sensitivity analysis point of view, as we will see next, and in the case of non-linear state problems like the one we will consider in Section 5.

4.3 Sensitivity analysis

The shape sensitivity analysis capabilities are implemented into the library as follows. In the finite element routines, all variables that depend on mesh nodal coordinates are represented using the `addouble` type. Most notably, the gradients of the basis functions and the Jacobian of the mapping from local to global coordinates are affected.

Using this version of the library, computing the partial derivatives with respect to shape is very easy. In case of our model problem we only have to define the mesh nodal positions as a function of the design

```
for (int i=0; i<nnodes; i++) {
  mesh.coord(i,0) =  alpha/alpha0*mesh0.coord(i,0);
  mesh.coord(i,1) =  alpha0/alpha*mesh0.coord(i,1);
}
```

where `alpha` has been declared independent, and `alpha0` is the design corresponding to the reference mesh `mesh0`. After this, exactly the same assembly

procedure described in the previous section is used to compute the partial derivatives $\partial \boldsymbol{r}/\partial \alpha$.

In this case we have only one design variable, and there is no need for any special exploitation of sparsity of geometrical derivatives. However, in cases when there are many design variables, it may be beneficial to perform the differentiation with respect to the mesh nodal coordinates, as explained in Section 3.6.3. This can be done simply by declaring the mesh nodal coordinates as independent variables. We return to the issue of computational efficiency of this approach in Section 5.3.

5 MODEL PROBLEM II

Let us now consider the discrete level sensitivity analysis in the context of a non-linear state problem. As an example we consider the well known Navier-Stokes equations for incompressible flow with a constant viscosity:

$$\begin{cases} -\mu\Delta\bar{u} + \rho(\bar{u}\cdot\nabla\bar{u}) + \nabla p = 0 \\ \nabla\cdot\bar{u} = 0 \end{cases} \tag{53}$$

Here $\bar{u} = (u_1, u_2)^T$ is the velocity, p is the pressure, ρ is the density, and μ is the viscosity of the fluid.

The weak form is obtained by multiplying with a vector valued test function $\boldsymbol{\varphi}$, and performing integration by parts:

$$\begin{aligned} \int_\Omega \mu\nabla u_i\cdot\nabla\varphi_i - \int_{\partial\Omega}\mu\nabla u_i\cdot\bar{n}\varphi_i + \int_\Omega \rho(\bar{u}\cdot\nabla u_i)\varphi_i \\ - \int_\Omega p\nabla\varphi_i + \int_{\partial\Omega} p\varphi_i n_i = 0, \quad i = 1,2 \\ (\nabla\cdot\bar{u}))\,\varphi_3 = 0. \end{aligned} \tag{54}$$

It is known, that when solving the Navier-Stokes equations the elements used to approximate the velocity and the pressure need to be chosen in a special way to obtain good convergence and approximation properties (see e.g. [68]). For this example, the famous Taylor-Hood element combination was chosen, i.e. quadratic triangular elements were used for the velocity and linear triangular elements for the pressure.

We consider the classical problem of a flow past a backward facing step. The geometry of the problem is shown in Figure 6, and the boundary conditions are as follows:

$$\begin{aligned} \bar{u} &= \bar{u}_{in} \quad \text{at } \Gamma_i \\ \bar{u} &= 0 \quad \text{at } \Gamma_w \\ \bar{u}\cdot\bar{t} = 0, \quad \bar{n}\cdot\sigma\cdot\bar{n} &= 0 \quad \text{at } \Gamma_o \end{aligned} \tag{55}$$

where σ is the stress tensor given by

$$\sigma_{ij} = -p\delta_{ij} + 2\mu\epsilon_{ij} = -p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{56}$$

and Γ_i, Γ_w and Γ_o are the inflow, wall and outflow boundaries, respectively.

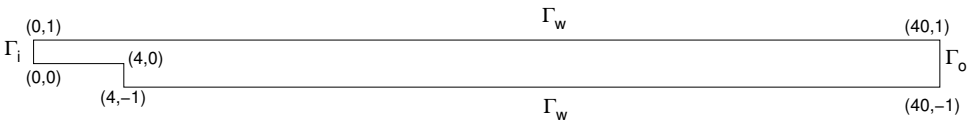

FIGURE 6 The backward facing step geometry.

The Reynolds number Re is a dimensionless number characterizing the nature of the flow. It is defined as

$$Re = \frac{\rho U L}{\mu}, \tag{57}$$

where U and L are the characteristic velocity and length scales of the flow. Low Reynolds number flows are laminar, and can be simulated using the Navier-Stokes equations alone. As the Reynolds number exceed a certain case dependent limit, the flow becomes turbulent. In that case, direct simulation using the Navier-Stokes equations is usually not feasible, because the length scales present in the flow are so small that an extremely dense mesh would be required. Instead, a turbulence model [69] is often used.

5.1 Solution of the state problem

Here we consider only a low Reynolds number case, where the flow is known to be laminar. However, due to their non-linear nature, the Navier-Stokes equations can be complicated to solve even in this case. Therefore, a Picard type linearization is applied, and the convective term is taken to be $\bar{u}^{k-1} \cdot \nabla \bar{u}^k$, where $\bar{u}^{k-1}$ means that the velocity is taken to be a constant from the previous iteration when the residual is linearized. In the context of our finite element library, such a linearization can be realized simply by using a function called `value`, which returns the value of a `addouble` variable as a regular double, thus ignoring its derivatives.

When a reasonably good approximation is achieved, i.e. the residual has dropped below given tolerance, we switch to relaxed Newton iteration. Only in the very end of the process we can use the full Newton iteration without relaxation. These switches happen after 68 and 92 iterations, and are clearly visible in the convergence history shown in Figure 7.

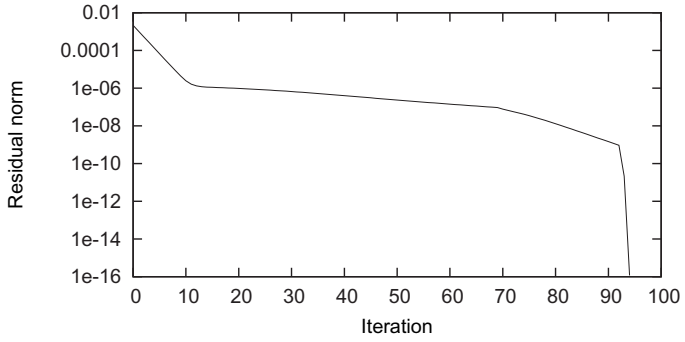

FIGURE 7 Convergence of the non-linear state problem.

The number of degrees of freedom was 16880 for the pressure approximation, and 66386 for both components of the velocity, resulting in the total number of 149652 unknowns. The total computation time needed for the 94 iterations was around 48 minutes on a server having two Intel Xeon Quad Core 2.33GHz processors and 8 gigabytes of RAM. No parallelization was exploited in this computation.

Streamlines of the computed solution are shown in Figure 8. To verify the results, comparison was made to the computations of Keskar et al. [70], and Lee et al. [71], and to the measurements of Lee et al. and Armaly et al. [72]. The present results are qualitatively similar to the referenced works, i.e. vortices similar to the ones shown in Figure 8 are found to be present in the flow. A quantitative comparison can be made using the location of reattachment at the bottom wall x_r and locations of the separation and attachment at the upper wall x_s, x_{rs} (see Figure 8 for illustration). Reasonably good agreement between the results was observed.

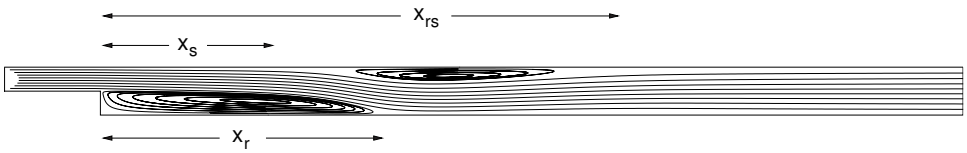

FIGURE 8 Streamlines of the solution.

5.2 Incomplete convergence and sensitivity analysis

The discrete level sensitivity formulas were derived under the assumption that the state problem is solved exactly, i.e. the residual vector is zero. However, in case of difficult non-linear problems it may not always be feasible to solve the

state problem with such a high accuracy. The question is, how does this affect the quality of the sensitivity analysis? We test this by computing the sensitivity of the pressure difference

$$J = \int_{\Gamma_i} p\mathrm{d}x - \int_{\Gamma_o} p\mathrm{d}x \tag{58}$$

with respect to the height of the step.

In Table 1 is shown the sensivitity of the objective functional (58) with respect to the height of the step computed using the discrete adjoint method for various orders of convergence of the state problem. Reference gradient was obtained using the finite difference formula (51), where the step length was 10^{-3}, and the residual norm was below the value 10^{-14} when the function evaluations were performed.

In cases where the relative residual of the state problem is $4.6 \cdot 10^{-4}$ or larger, the obtained value for the sensitivity is not even close to the correct value. When the relative residual has reached the value $4.6 \cdot 10^{-6}$, the deviation from the reference value is less than one per cent. Below this level, an order of magnitude decrease in the residual produces roughly one more correct digit in the sensitivity. That is, to obtain a high level of consistency, the state problem should be solved with relatively high accuracy.

However, in the early stages of the optimization very high level of consistency may not even be needed. Later on during the optimization the changes in the design are often smaller, and a good initial guess for the state problem is therefore available from the previous evaluation, making it easier to solve the state problem with higher precision.

TABLE 1 AD gradient versus order of convergence.

Method	Relative residual	Gradient
AD	$3.1 \cdot 10^{-2}$	$-18.3798000050421491 \cdot 10^{-5}$
	$4.6 \cdot 10^{-4}$	$-3.4392497720377437 \cdot 10^{-5}$
	$4.6 \cdot 10^{-6}$	$-1.4563802649023588 \cdot 10^{-5}$
	$4.3 \cdot 10^{-8}$	$-1.4444250576081891 \cdot 10^{-5}$
	$4.0 \cdot 10^{-10}$	$-1.4443244719085867 \cdot 10^{-5}$
	$4.6 \cdot 10^{-12}$	$-1.4443235472460206 \cdot 10^{-5}$
	$5.4 \cdot 10^{-14}$	$-1.4443235364881294 \cdot 10^{-5}$
HOFD		$-1.4443235364440352 \cdot 10^{-5}$

5.3 Computing large shape gradients

Let us now consider the sensitivity analysis of the objective functional (58) in a case where we have a lot of design variables. As an example we consider sensitivity analysis in a CAD free setting, where the design variables are the x^2 displacements of the boundary nodes on the bottom of channel after the step. In our

discretization this equals $M = 450$ design variables.

Let α_j be the j^{th} design variable, and $b(j)$ be the global index of the corresponding node. We number the design variables so that $b(j)$ and $b(j+1)$ are always neighbouring nodes. The sensitivities of the mesh nodes in $[4, 40] \times [-1, 0]$ are defined analytically as follows. Let the node i be located in this area. It holds $\tilde{x}^1_{b(j)} \leq \tilde{x}^1_i \leq \tilde{x}^1_{b(j+1)}$ for some j, and we can define

$$\frac{\partial x^2_i}{\partial \alpha_j} = (1-t) \cdot (-\tilde{x}^2_i) \qquad \text{and} \qquad \frac{\partial x^2_i}{\partial \alpha_{j+1}} = t \cdot (-\tilde{x}^2_i), \tag{59}$$

where

$$t = (\tilde{x}^1_i - \tilde{x}^1_{b(j)}) / (\tilde{x}^1_{b(j+1)} - \tilde{x}^1_{b(j)}) \in [0, 1]. \tag{60}$$

In Figure 9 are shown the obtained sensitivities of J_h for a part of the channel as a function of the location of the corresponding boundary node.

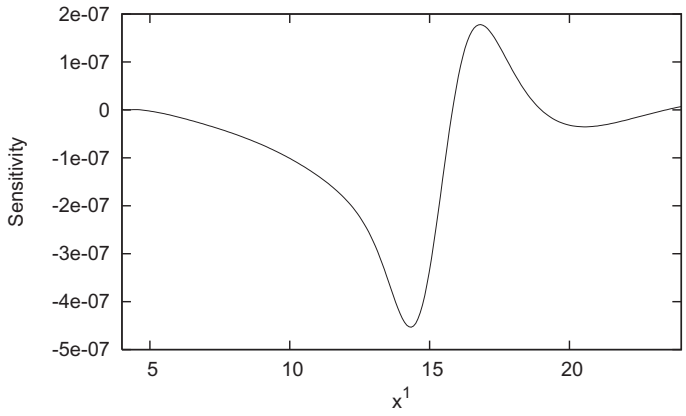

FIGURE 9 Sensitivity of J_h with respect to x^2 displacements of nodes on the bottom of the channel.

Using our implementation, the following computation times were observed:

- One evaluation of the residual vector takes on average about 0.95 seconds. We shall use the abbreviation T_R for this computation time later while comparing some sensitivity analysis approaches.

- Evaluating the residual and the Jacobian $\partial \boldsymbol{r} / \partial \boldsymbol{q}$ takes around 2 seconds (abbreviation T_{R+J}).

- If we simultaneously compute also $\partial \boldsymbol{r} / \partial \boldsymbol{X}$, computation time increases to about 4.5 seconds (abbreviation T_{R+J+X}).

- Solution of a linear system with the Jacobian as the coefficient matrix takes around 23 seconds (abbreviation T_L).

In Table 2 we summarize the computation times to obtain the gradient using different methods. The finite difference approach performs a reanalysis for each design variable. Each reanalysis requires (at least) one iteration using the Newton's method, and a lot of time is consumed to solve the the linear systems. Some improvement could be obtained by reusing the factorization of the Jacobian matrix, but obviously there is no way the finite differences could compete with the adjoint approach.

The second method in the Table is the adjoint approach, where the residual is differentiated directly with respect to all the design variables. We implemented this using finite differences, but performing the differentiation with a dense forward mode AD results in comparable computational complexity. Here, only one linear system is solved, but the computation time of $\partial \boldsymbol{r}/\partial \boldsymbol{\alpha}$ is approximately M times the original residual vector evaluation time.

The last approach is based on the exploitation of the sparsity as explained in Section 3.6.3. That is, we perform differentiation with respect to the mesh nodal coordinates to obtain $\partial \boldsymbol{r}/\partial \boldsymbol{X}$, from which $\partial \boldsymbol{r}/\partial \boldsymbol{\alpha}$ can be easily computed as $\partial \boldsymbol{X}/\partial \boldsymbol{\alpha}$ is known. The total duration of the sensitivity analysis phase using this approach was 32 seconds, only slightly longer than 30.5 seconds that was on average required for one non-linear iteration during the state problem solution phase.

TABLE 2 Total computation times required for the sensitivity analysis using different approaches.

Method	Required operations	Computation time
Finite differences	M x (T_{R+J} + T_L)	3 h 36 min 5 s
Adjoint without sparsity	T_J + T_L + M x T_R	15 min 11 s
Adjoint with sparse AD	T_{R+J+X} + T_L	32 sec

6 AUTHOR'S CONTRIBUTION

Sparse forward mode automatic differentiation has been previously utilized very little, because it has been considered too slow for practical purposes [8]. Article [PI] presents a simple practical implementation of this method, and by computational examples shows that a quite reasonable overhead is achieved, making the method suitable for many applications. The article is mainly written by the author, and the presented techniques and numerical tests are developed by the author. The electromagnetic simulator utilized in the tests was obtained from the research partners, as acknowledged in the article.

As mentioned earlier in this thesis, a mesh deformation procedure is needed to obtain a numerical model for which consistent derivatives can be computed. In the article [PII] written by the author, a particular deformation technique is proposed. The method is a minor modification of an existing technique as referenced in the article. The article also considers an example optimization problem, which is an inverse design problem featuring the Navier-Stokes equations as the state problem. All computations were performed by the author.

The finite element library presented in Chapter 4 has been implemented by the author. The library is otherwise based on quite standard techniques, but it utilizes the sparse forward mode implementation presented in [PI] as a technique enabling sensitivity analysis. The library has been used to study an optimization problem related to fibre orientation control, results of which are presented in the article [PIII]. The author has written the section about automatic differentiation in the article and provided some assistance during the computations.

The author came up with the idea to apply the so called pseudo solid approach to solve Bernoulli free boundary problems, and developed the numerical methods presented in the article [PIV]. The problem formulation and the mathematical analysis were done by the co-authors, who also wrote the related parts of the article. The parts related to discretization, optimization, and numerical results were mainly written by the author, who also performed all computations.

Although there are many published applications of shape optimization in the context of electromagnetics problems [73, 74, 75], gradient based shape optimization is still relatively seldom used in the design of microwave devices [76].

To our knowledge, the article [PV] presents the first work where exact discrete level sensitivity analysis is performed on a method of moments solver. The author applied automatic differentiation techniques presented in the article [PI] to the solver provided by the project partners, and was the main writer of the article [PV], except for the part describing the electromagnetic solver. The author also performed all computations presented in the article.

The article [PVI] is an extension of the article [PV], and deals with the practical problems related to geometry parametrization and sensitivity analysis in case where the geometry is parametrized using splines, but the mesh generator does not provide sensitivity information. The author was the main writer of the article, and performed all computations.

7 CONCLUSIONS AND FUTURE WORK

This thesis has considered methods for computing consistent shape derivatives, i.e. derivatives of the numerical model that are exact within very high accuracy. The methodology used is the classical boundary variation based shape optimization using deformed meshes. While highly useful in many applications, this approach has its limitations. Namely, large changes in the shape usually result in an invalid mesh and deterioration of accuracy in the simulations, unless the mesh is regenerated.

An area of further study would be, how to efficiently deal with the small discontinuities that are caused by the inevitable mesh generations. Namely, a complete restart of the optimization after each mesh regeneration loses valuable information, such as the Hessian approximation. One possibility could be gradient-only optimization methods [77], where the line search is performed by looking at the sign of the projected gradient, and function values are therefore irrelevant.

One problem often encountered while setting shape optimization problems is the need to have a link between the geometry parameterization and the mesh, as discussed in the article [PVI]. If an explicit representation of the shape using e.g. spline curves is not mandatory, free form deformation type strategies could be exploited. Namely, if only the deformation of the mesh is parameterized for the optimization, we do not have to know anything about the parameterization that was used to generate the original mesh.

Methods that permit topological changes, such as level set type methods, are obviously interesting. They allow much more flexible changes in the shape, and creation of completely new, perhaps unexpected, shapes. However, level set methods require a lot of problem dependent manual work, since they are based on the continuous level shape differentiation. It would be interesting to develop methods that utilize AD to automatize some of the tasks related to level set methods.

REFERENCES

[1] F. Muyl, L. Dumas, and V. Herbert, "Hybrid method for aerodynamic shape optimization in automotive industry," *Computers & Fluids*, vol. 33, pp. 849–858, 2004.

[2] R. M. Hicks and P. A. Henne, "Wing design by numerical optimization," *Journal of Aircraft*, vol. 15, pp. 407–412, 1978.

[3] Y. W. Chun and E. J. Haug, "Two-dimensional shape optimal design," *International Journal for Numerical Methods in Engineering*, vol. 13, pp. 311–336, 1978.

[4] E. J. Haug, *Applied optimal design : mechanical and structural systems*. Wiley, 1979.

[5] J. Cea, "Numerical methods of shape optimal design," in *Optimization of Distributed Parameter Structures* (E. J. Haug and J. Cea, eds.), pp. 1049–1087, Sijthoff & Noordholl, Alphen aan den Rijn, Netherlands, 1980.

[6] J. P. Webb, "Finite-element analysis of the effect of geometric tolerances on performance over a frequency band," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, pp. 306–310, 2004.

[7] K. Kärkkäinen, *Shape Sensitivity Analysis for Numerical Solution of Free Boundary Problems*. PhD thesis, University of Jyväskylä, 2005.

[8] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, 2nd ed., 2008.

[9] N. K. Nikolova, J. W. Bandler, and M. H. Bakr, "Adjoint techniques for sensitivity analysis in high-frequency structure CAD," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, pp. 403–419, 2004.

[10] K. Miettinen, *Nonlinear multiobjective optimization*. Kluwer Academic Publishers, Boston, 1999.

[11] B. Mohammadi and O. Pironneau, "Applied optimal shape design," *Journal of Computational and Applied Mathematics*, vol. 149, pp. 193–205, 2002.

[12] J. M. Johnson and Y. Rahmat-Samii, "Genetic algorithms and method of moments (GA/MOM) for the design of integrated antennas," *IEEE Transactions on Antennas and Propagation*, vol. 47, pp. 1606–1614, 1999.

[13] M. P. Bendsøe, "Optimal shape design as a material distribution problem," *Structural Optimization*, vol. 1, pp. 192–202, 1989.

[14] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag, 2003.

[15] B. Szabó and I. Babuška, *Finite Element Analysis*. Wiley Interscience, 1991.

[16] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu, *The finite element method: its basis and fundamentals*. Butterworth-Heinemann, 2005.

[17] R. J. LeVeque, *Finite-Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[18] C. Pozrikidis, *A practical guide to boundary element methods with the software library BEMLIB*. CRC Press, 2002.

[19] I. Babuška, U. Banerjee, and J. E. Osborn, "Survey of meshless and generalized finite element methods: A unified approach," *Acta Numerica*, vol. 12, pp. 1–125, 2003.

[20] N. H. Kim, K. K. Choi, and M. E. Botkin, "Numerical method for shape optimization using meshfree method," *Structural and Multidisciplinary Optimization*, vol. 24, pp. 418–429, 2003.

[21] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs, "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement," *Computer Methods in Applied Mechaniccs and Engineering*, vol. 194, pp. 4135–4195, 2005.

[22] W. A. Wall, M. A. Frenzel, and C. Cyron, "Isogeometric structural shape optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, pp. 2976–2988, 2008.

[23] S. Cho and S.-H. Ha, "Isogeometric shape design optimization: exact geometry and enhanced sensitivity," *Structural and Multidisciplinary Optimization*, vol. 38, pp. 53–70, 2007.

[24] J. Toivanen, *Fictitious domain method applied to shape optimization*. PhD thesis, University of Jyväskylä, 1997.

[25] J. Daňková and J. Haslinger, "Numerical realization of a fictitious domain approach used in shape optimization. Part I: distributed controls," *Applications of Mathematics*, vol. 41, no. 2, pp. 123–147, 1996.

[26] R. Glowinski, T.-W. Pan, and J. Periaux, "A fictitious domain method for Dirichlet problem and applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 111, pp. 283–303, 1994.

[27] B. Mohammadi and O. Pironneau, *Applied Shape Optimization for Fluids*. Oxford University Press, 2001.

[28] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[29] C. R. Reeves and J. E. Rowe, *Genetic algorithms: principles and perspectives. A guide to GA theory*. Kluwer Academic Publishers, 2003.

[30] D.-H. Li and M. Fukushima, "A modified BFGS method and its global convergence in nonconvex minimization," *Journal of Computational and Applied Mathematics*, vol. 129, pp. 15–35, 2001.

[31] P. K. Gudla and R. Ganguli, "An automated hybrid genetic-conjugate gradient algorithm for multimodal optimization problems," *Applied Mathematics and Computation*, vol. 167, pp. 1457–1474, 2004.

[32] P. L. George, *Automatic Mesh Generation: Application to Finite Element Methods*. John Wiley & Sons, 1991.

[33] J. F. Thompson, B. K. Soni, and N. P. Weatherhill, eds., *Handbook of Grid Generation*. CRC Press, 1999.

[34] D. Field, "Laplacian smoothing and Delaunay triangulations," *Communications in Applied Numerical Methods*, vol. 4, pp. 709–712, 1988.

[35] J. J. Ródenas, F. J. Fuenmayor, and J. E. Tarancón, "A numerical methodology to assess the quality of the design velocity field computation methods in shape sensitivity analysis," *International Journal for Numerical Methods in Engineering*, vol. 59, pp. 1725 – 1747, 2004.

[36] C. L. Bottasso, D. Detomi, and R. Serra, "The ball-vertex method: a new simple spring analogy method for unstructured dynamic meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, pp. 4244–4264, 2005.

[37] A. A. Johnson and T. E. Tezduyar, "Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces," *Computer Methods in Applied Mechanics and Engineering*, vol. 119, pp. 73–94, 1994.

[38] B. T. Helenbrook, "Mesh deformation using the biharmonic operator," *International Journal for Numerical Methods in Engineering*, vol. 56, pp. 1007–1021, 2003.

[39] A. de Boer, M. S. van der Schoot, and H. Bijl, "Mesh deformation based on radial basis function interpolation," *Computers and Structures*, vol. 85, pp. 784–795, 2007.

[40] S. Jakobsson and O. Amoignon, "Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization," *Computers & Fluids*, vol. 36, pp. 1119–1136, 2007.

[41] J. Haslinger and R. A. E. Mäkinen, *Introduction to Shape Optimization: Theory, Approximation, and Computation*. SIAM, Philadelphia, 2003.

[42] J. Sokolowski and J. Zolésio, *Introduction to shape optimization: Shape Sensitivity Analysis*. Springer-Verlag, 1992.

[43] M. Souli and J. P. Zolesio, "Shape derivative of discretized problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 108, pp. 187–199, 1993.

[44] J. Haslinger and P. Neittaanmäki, *Finite Element Approximation for Optimal Shape Design: Theory and Applications*. John Wiley and Sons, Chichester, 1988.

[45] E. J. Nielsen and M. A. Park, "Using an adjoint approach to eliminate mesh sensitivities in computational design," in *Proceedings of 43rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada*, 2005. AIAA Paper 2005-0491.

[46] D. A. Tortorelli and P. Michaleris, "Design sensitivity analysis: Overview and review," *Inverse Problems in Engineering*, vol. 1, pp. 71–103, 1994.

[47] J. C. Newman III, "Overview of sensitivity analysis and shape optimization for complex aerodynamical configurations," *Journal of Aircraft*, vol. 36, no. 1, pp. 87–96, 1999.

[48] F. v. Keulen, R. T. Hafka, and N. H. Kim, "Review of options for structural design sensitivity analysis. Part1: Linear systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, pp. 3213–3243, 2005.

[49] J. E. V. Peter and R. P. Dwight, "Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches," *Computers & Fluids*, vol. 39, pp. 373–391, 2010.

[50] P. Neittaanmäki and T. Tiihonen, "Sensitivity analysis for a class of optimal shape design problems." Report 29, Department of Mathematics, University of Jyväskylä, 1985.

[51] J. Burkardt, M. Gunzburger, and J. Peterson, "Insensitive functionals, inconsistent gradients, spurious minima, and regularized functionals in flow optimization problems," *International Journal of Computational Fluid Dynamics*, vol. 16, no. 3, pp. 171–185, 2002.

[52] S. K. Nadarajah and A. Jameson, "A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization." AIAA Paper 2000-0067, 2000.

[53] M. Gunzburger, "Adjoint equation-based methods for control problems in incompressible, viscous flows," *Flow, Turbulence and Combustion*, vol. 65, pp. 249–272, 2000.

[54] M. B. Giles and N. A. Pierce, "An introduction to the adjoint approach to design," *Flow, Turbulence and Combustion*, vol. 65, pp. 393–415, 2000.

[55] H.-G. Seong and K. K. Choi, "Boundary-layer approach to shape design sensitivity analysis," *Mechanics of Structures and Machines*, vol. 15, no. 2, pp. 241–263, 1987.

[56] G. Haase, U. Langer, W. Lindner, and W. Mühlhuber, "Optimal sizing of industrial structural mechanics problems using AD," in *Automatic differentiation of algorithms: from simulation to optimization* (G. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, eds.), pp. 181–188, Springer, 2002.

[57] B. Mohammadi, J.-M. Malé, and N. Rostaing-Schmidt, "Automatic differentiation in direct and reverse modes: Application to optimum shapes design in fluid mechanics," in *Computational Differentiation: Techniques, Applications, and Tools* (M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds.), pp. 309–318, SIAM, 1996.

[58] F. Courty, A. Dervieux, B. Koobus, and L. Hascoët, "Reverse automatic differentiation for optimum design: from adjoint state assembly to gradient computation," *Optimization Methods and Software*, vol. 18, no. 5, pp. 615–627, 2003.

[59] R. A. E. Mäkinen, K. Hiltunen, M. Laitinen, and A. Niemistö, "Exact linearization in numerical simulation and optimization," in *Proceedings of the VIII Finnish Mechanics Days*, 2003.

[60] C. H. Bischof, P. M. Khademi, A. Bouaricha, and A. Carle, "Efficient computation of gradients and Jacobians by dynamic exploitation of sparsity in automatic differentiation," *Optimization Methods and Software*, vol. 7, pp. 1–39, 1996.

[61] C. H. Bischof, H. M. Bücker, and A. Rasch, "Sensitivity analysis of turbulence models using automatic differentiation," *SIAM Journal on Scientific Computing*, vol. 26, pp. 510–522, 2004.

[62] N. K. Nikolova, J. Zhu, D. Li, M. H. Bakr, and J. W. Bandler, "Sensitivity analysis of network parameters with electromagnetic frequency-domain simulators," *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, no. 2, pp. 670–681, 2006.

[63] D.-H. Kim, K. S. Ship, and J. K. Sykulski, "Applying continuum design sensitivity analysis combined with standard EM software to shape optimization in magnetostatic problems," *IEEE Transactions on Magnetics*, vol. 40, no. 2, pp. 1156–1159, 2004.

[64] L. Holzleitner and K. G. Mahmoud, "Structural shape optimization using MSC/NASTRAN and sequential quadratic programming," *Computers and Structures*, vol. 70, pp. 487–514, 1999.

[65] J. Haataja, J. Heikonen, Y. Leino, J. Rahola, J. Ruokolainen, and V. Savolainen, *Numeeriset menetelmät käytännössä*. CSC - Tieteellinen laskenta Oy, Helsinki, 1999.

[66] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, pp. 1309 – 1331, 2009.

[67] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, "A supernodal approach to sparse partial pivoting," *SIAM J. Matrix Analysis and Applications*, vol. 20, no. 3, pp. 720–755, 1999.

[68] P. M. Gresho, R. L. Sani, and M. S. Engelman, *Incompressible Flow and the Finite Element Method*. John Wiley & Sons, 1998.

[69] D. C. Wilcox, *Turbulence modelling for CFD*. DCW Industries, La Cañada, 1993.

[70] J. Keskar and D. A. Lyn, "Computations of laminar backward-facing step flow at Re=800 with a spectral domain decomposition method," *International Journal for Numerical Methods in Fluids*, vol. 29, pp. 411–427, 1999.

[71] T. Lee and D. Mateescu, "Experimental and numerical investigation of 2-D backward-facing step flow," *Journal of Fluids and Structures*, vol. 12, pp. 703–716, 1998.

[72] B. F. Armaly, F. Durst, J. C. F. Pereira, and B. Schonung, "Experimental and theoretical investigation of backward-facing step flow," *Journal of Fluid Mechanics*, vol. 127, pp. 473–496, 1983.

[73] A. Marrocco and O. Pironneau, "Optimum design with Lagrangian finite elements: Design of an electromagnet," *Computer Methods in Applied Mechanics and Engineering*, vol. 15, pp. 277–308, 1978.

[74] P. Neittaanmäki, M. Rudnicki, and A. Savini, *Inverse Problems and Optimal Design in Electricity and Magnetism*. Oxford University Press, Oxford, 1996.

[75] Y.-S. Chung, C. Cheon, I. Park, and S. Hahn, "Optimal shape design of microwave device using FDTD and design sensitivity analysis," *IEEE Transactions on Microwave Theory and Techniques*, vol. 48, pp. 2289–2296, 2000.

[76] A. Bondeson, Y. Yang, and P. Weinerfelt, "Shape optimization for radar cross sections by a gradient method," *International Journal for Numerical Methods in Engineering*, vol. 61, pp. 687–715, 2004.

[77] D. N. Wilke, S. Kok, and A. A. Groenwold, "The application of gradient-only optimization methods for problems discretized using non-constant methods," *Structural and Multidisciplinary Optimization*, vol. 40, pp. 433–451, 2010.

YHTEENVETO (FINNISH SUMMARY)

Tässä työssä, nimeltään *Muodon optimointia konsistenttien herkkyyksien avulla*, sovelletaan gradienttipohjaisia menetelmiä muodonoptimointitehtäviin, joissa tilayhtälö on osittaisdiffentiaaliyhtälö (ODY). Automaattinen derivointi (AD) tarjoaa suoraviivaisen tavan laskea kohdefunktion gradientti, kun käytettävän ratkaisijan (simulaattorin) lähdekoodi on saatavilla. Tuloksena saadaan numeerisen mallin tarkka gradientti, jota kutsutaan tässä yhteydessä nimellä konsistentti gradientti.

Tässä työssä esitetään uusi toteutus niin sanotulle harvalle etenevälle automaattiselle derivoinnille. Tekniikan avulla voidaan automaattisesti käyttää hyväksi derivaattavektorien harvuusominaisuuksia, mikä voi johtaa merkittäviin säästöihin laskenta-ajassa. Mikäli laskettavat derivaattavektorit eivät ole harvoja, toteutus on ainoastaan hivenen raskaampi kuin perinteinen etenevän AD:n toteutus.

Menetelmän avulla voidaan laskea vektorifunktioiden suuria mutta harvoja Jacobin matriiseja siten, että laskentakoodiin tarvitsee tehdä ainoastaan minimaalisia muutoksia. Yhdessä diskreetin liittotilatekniikan kanssa mentelmä tarjoaa tehokkaan tavan laskea muotogradientteja myös siinä tapauksessa että suunnittelumuuttujia on paljon.

Esitettyä tekniikkaa sovelletaan muotoherkkyysanalyysin toteuttamiseksi olemassaolevaan sähkömagneettiseen ratkaisijaan, joka perustuu momenttimenetelmään. Muokattua ratkaisijaa käytetään useiden antennisuunnittelluun liittyvien muodonoptimointitehtävien ratkaisemiseen.

Herkkyysanalyysi toteutetaan myös elementtimenetelmän yhteydessä. Toteutusta sovelletaan muun muassa kontrollitehtävään, joka mallintaa kuituorientaatiota yksinkertaistetussa paperikoneen perälaatikossa. Lisäksi käsitellään Bernoullin vapaan reunan ongelmaan liittyvää muodon optimointitehtävää. Tilayhtälölle kehitetään niin sanottuun pseudosolidi-lähestymistapaan perustuva ratkaisija, joka mahdollistaa vapaan reunan tehtävän tehokkaan ratkaisemisen ja muotoherkkyysanalyysin.

JYVÄSKYLÄ STUDIES IN COMPUTING

1 Ropponen, Janne, Software risk management - foundations, principles and empirical findings. 273 p. Yhteenveto 1 p. 1999.
2 Kuzmin, Dmitri, Numerical simulation of reactive bubbly flows. 110 p. Yhteenveto 1 p. 1999.
3 Karsten, Helena, Weaving tapestry: collaborative information technology and organisational change. 266 p. Yhteenveto 3 p. 2000.
4 Koskinen, Jussi, Automated transient hypertext support for software maintenance. 98 p. (250 p.) Yhteenveto 1 p. 2000.
5 Ristaniemi, Tapani, Synchronization and blind signal processing in CDMA systems. - Synkronointi ja sokea signaalinkäsittely CDMA järjestelmässä. 112 p. Yhteenveto 1 p. 2000.
6 Laitinen, Mika, Mathematical modelling of conductive-radiative heat transfer. 20 p. (108 p.) Yhteenveto 1 p. 2000.
7 Koskinen, Minna, Process metamodelling. Conceptual foundations and application. 213 p. Yhteenveto 1 p. 2000.
8 Smolianski, Anton, Numerical modeling of two-fluid interfacial flows. 109 p. Yhteenveto 1 p. 2001.
9 Nahar, Nazmun, Information technology supported technology transfer process. A multi-site case study of high-tech enterprises. 377 p. Yhteenveto 3 p. 2001.
10 Fomin, Vladislav V., The process of standard making. The case of cellular mobile telephony. - Standardin kehittämisen prosessi. Tapaustutkimus solukkoverkkoon perustuvasta matkapuhelintekniikasta. 107 p. (208 p.) Yhteenveto 1 p. 2001.
11 Päivärinta, Tero, A genre-based approach to developing electronic document management in the organization. 190 p. Yhteenveto 1 p. 2001.
12 Häkkinen, Erkki, Design, implementation and evaluation of neural data analysis environment. 229 p. Yhteenveto 1 p. 2001.
13 Hirvonen, Kullervo, Towards better employment using adaptive control of labour costs of an enterprise. 118 p. Yhteenveto 4 p. 2001.
14 Majava, Kirsi, Optimization-based techniques for image restoration. 27 p. (142 p.) Yhteenveto 1 p. 2001.
15 Saarinen, Kari, Near infra-red measurement based control system for thermo-mechanical refiners. 84 p. (186 p.) Yhteenveto 1 p. 2001.
16 Forsell, Marko, Improving component reuse in software development. 169 p. Yhteenveto 1 p. 2002.
17 Virtanen, Pauli, Neuro-fuzzy expert systems in financial and control engineering. 245 p. Yhteenveto 1 p. 2002.
18 Kovalainen, Mikko, Computer mediated organizational memory for process control. Moving CSCW research from an idea to a product. 57 p. (146 p.) Yhteenveto 4 p. 2002.
19 Hämäläinen, Timo, Broadband network quality of service and pricing. 140 p. Yhteenveto 1 p. 2002.
20 Martikainen, Janne, Efficient solvers for discretized elliptic vector-valued problems. 25 p. (109 p.) Yhteenveto 1 p. 2002.
21 Mursu, Anja, Information systems development in developing countries. Risk management and sustainability analysis in Nigerian software companies. 296 p. Yhteenveto 3 p. 2002.
22 Seleznyov, Alexandr, An anomaly intrusion detection system based on intelligent user recognition. 186 p. Yhteenveto 3 p. 2002.
23 Lensu, Anssi, Computationally intelligent methods for qualitative data analysis. 57 p. (180 p.) Yhteenveto 1 p. 2002.
24 Ryabov, Vladimir, Handling imperfect temporal relations. 75 p. (145 p.) Yhteenveto 2 p. 2002.
25 Tsymbal, Alexey, Dynamic integration of data mining methods in knowledge discovery systems. 69 p. (170 p.) Yhteenveto 2 p. 2002.
26 Akimov, Vladimir, Domain decomposition methods for the problems with boundary layers. 30 p. (84 p.). Yhteenveto 1 p. 2002.
27 Seyukova-Rivkind, Ludmila, Mathematical and numerical analysis of boundary value problems for fluid flow. 30 p. (126 p.) Yhteenveto 1 p. 2002.
28 Hämäläinen, Seppo, WCDMA Radio network performance. 235 p. Yhteenveto 2 p. 2003.
29 Pekkola, Samuli, Multiple media in group work. Emphasising individual users in distributed and real-time CSCW systems. 210 p. Yhteenveto 2 p. 2003.
30 Markkula, Jouni, Geographic personal data, its privacy protection and prospects in a location-based service environment. 109 p. Yhteenveto 2 p. 2003.
31 Honkaranta, Anne, From genres to content analysis. Experiences from four case organizations. 90 p. (154 p.) Yhteenveto 1 p. 2003.
32 Raitamäki, Jouni, An approach to linguistic pattern recognition using fuzzy systems. 169 p. Yhteenveto 1 p. 2003.
33 Saalasti, Sami, Neural networks for heart rate time series analysis. 192 p. Yhteenveto 5 p. 2003.
34 Niemelä, Marketta, Visual search in graphical interfaces: a user psychological approach. 61 p. (148 p.) Yhteenveto 1 p. 2003.
35 You, Yu, Situation Awareness on the world wide web. 171 p. Yhteenveto 2 p. 2004.
36 Taatila, Vesa, The concept of organizational competence – A foundational analysis. - Perusteanalyysi organisaation kompetenssin käsitteestä. 111 p. Yhteenveto 2 p. 2004.

37 LYYTIKÄINEN, VIRPI, Contextual and structural metadata in enterprise document management. - Konteksti- ja rakennemetatieto organisaation dokumenttien hallinnassa. 73 p. (143 p.) Yhteenveto 1 p. 2004.
38 KAARIO, KIMMO, Resource allocation and load balancing mechanisms for providing quality of service in the Internet. 171 p. Yhteenveto 1 p. 2004.
39 ZHANG, ZHEYING, Model component reuse. Conceptual foundations and application in the metamodeling-based systems analysis and design environment. 76 p. (214 p.) Yhteenveto 1 p. 2004.
40 HAARALA, MARJO, Large-scale nonsmooth optimization variable metric bundle method with limited memory. 107 p. Yhteenveto 1 p. 2004.
41 KALVINE, VIKTOR, Scattering and point spectra for elliptic systems in domains with cylindrical ends. 82 p. 2004.
42 DEMENTIEVA, MARIA, Regularization in multistage cooperative games. 78 p. 2004.
43 MAARANEN, HEIKKI, On heuristic hybrid methods and structured point sets in global continuous optimization. 42 p. (168 p.) Yhteenveto 1 p. 2004.
44 FROLOV, MAXIM, Reliable control over approximation errors by functional type a posteriori estimates. 39 p. (112 p.) 2004.
45 ZHANG, JIAN, Qos- and revenue-aware resource allocation mechanisms in multiclass IP networks. 85 p. (224 p.) 2004.
46 KUJALA, JANNE, On computation in statistical models with a psychophysical application. 40 p. (104 p.) 2004.,
47 SOLBAKOV, VIATCHESLAV, Application of mathematical modeling for water environment problems. 66 p. (118 p.) 2004.
48 HIRVONEN, ARI P., Enterprise architecture planning in practice. The Perspectives of information and communication technology service provider and end-user. 44 p. (135 p.) Yhteenveto 2 p. 2005.
49 VARTIAINEN, TERO, Moral conflicts in a project course in information systems education. 320 p. Yhteenveto 1p. 2005.
50 HUOTARI, JOUNI, Integrating graphical information system models with visualization techniques. - Graafisten tietojärjestelmäkuvausten integrointi visualisointitekniikoilla. 56 p. (157 p.) Yhteenveto 1p. 2005.
51 WALLENIUS, EERO R., Control and management of multi-access wireless networks. 91 p. (192 p.) Yhteenveto 3 p. 2005.
52 LEPPÄNEN, MAURI, An ontological framework and a methodical skeleton for method engineering – A contextual approach. 702 p. Yhteenveto 2 p. 2005.
53 MATYUKEVICH, SERGEY, The nonstationary Maxwell system in domains with edges and conical points. 131 p. Yhteenveto 1 p. 2005.
54 SAYENKO, ALEXANDER, Adaptive scheduling for the QoS supported networks. 120 p. (217 p.) 2005.
55 KURJENNIEMI, JANNE, A study of TD-CDMA and WCDMA radio network enhancements. 144 p. (230 p.) Yhteenveto 1 p. 2005.
56 PECHENIZKIY, MYKOLA, Feature extraction for supervised learning in knowledge discovery systems. 86 p. (174 p.) Yhteenveto 2 p. 2005.
57 IKONEN, SAMULI, Efficient numerical methods for pricing American options. 43 p. (155 p.) Yhteenveto 1 p. 2005.
58 KÄRKKÄINEN, KARI, Shape sensitivity analysis for numerical solution of free boundary problems. 83 p. (119 p.) Yhteenveto 1 p. 2005.
59 HELFENSTEIN, SACHA, Transfer. Review, reconstruction, and resolution. 114 p. (206 p.) Yhteenveto 2 p. 2005.
60 NEVALA, KALEVI, Content-based design engineering thinking. In the search for approach. 64 p. (126 p.) Yhteenveto 1 p. 2005.
61 KATASONOV, ARTEM, Dependability aspects in the development and provision of location-based services. 157 p. Yhteenveto 1 p. 2006.
62 SARKKINEN, JARMO, Design as discourse: Representation, representational practice, and social practice. 86 p. (189 p.) Yhteenveto 1 p. 2006.
63 ÄYRÄMÖ, SAMI, Knowledge mining using robust clustering. 296 p. Yhteenveto 1 p. 2006.
64 IFINEDO, PRINCELY EMILI, Enterprise resource planning systems success assessment: An integrative framework. 133 p. (366 p.) Yhteenveto 3 p. 2006.
65 VIINIKAINEN, ARI, Quality of service and pricingin future multiple service class networks. 61 p. (196 p.) Yhteenveto 1 p. 2006.
66 WU, RUI, Methods for space-time parameter estimation in DS-CDMA arrays. 73 p. (121 p.) 2006.
67 PARKKOLA, HANNA, Designing ICT for mothers. User psychological approach. – Tieto- ja viestintätekniikoiden suunnittelu äideille. Käyttäjäpsykologinen näkökulma. 77 p. (173 p.) Yhteenveto 3 p. 2006.
68 HAKANEN, JUSSI, On potential of interactive multiobjective optimization in chemical process design. 75 p. (160 p.) Yhteenveto 2 p. 2006.
69 PUTTONEN, JANI, Mobility management in wireless networks. 112 p. (215 p.) Yhteenveto 1 p. 2006.
70 LUOSTARINEN, KARI, Resource , management methods for QoS supported networks. 60 p. (131 p.) 2006.
71 TURCHYN, PAVLO, Adaptive meshes in computer graphics and model-based simulation. 27 p. (79 p.) Yhteenveto 1 p.
72 ZHOVTOBRYUKH, DMYTRO, Context-aware web service composition. 290 p. Yhteenveto 2 p. 2006.

73 KOHVAKKO, NATALIYA, Context modeling and utilization in heterogeneous networks. 154 p. Yhteenveto 1 p. 2006.
74 MAZHELIS, OLEKSIY, Masquerader detection in mobile context based on behaviour and environment monitoring. 74 p. (179 p). Yhteenveto 1 p. 2007.
75 SILTANEN, JARMO, Quality of service and dynamic scheduling for traffic engineering in next generation networks. 88 p. (155 p.) 2007.
76 KUUVA, SARI, Content-based approach to experiencing visual art. - Sisältöperustainen lähestymistapa visuaalisen taiteen kokemiseen. 203 p. Yhteenveto 3 p. 2007.
77 RUOHONEN, TONI, Improving the operation of an emergency department by using a simulation model. 164 p. 2007.
78 NAUMENKO, ANTON, Semantics-based access control in business networks. 72 p. (215 p.) Yhteenveto 1 p. 2007.
79 WAHLSTEDT, ARI, Stakeholders' conceptions of learning in learning management systems development. - Osallistujien käsitykset oppimisesta oppimisympäristöjen kehittämisessä. 83 p. (130 p.) Yhteenveto 1 p. 2007.
80 ALANEN, OLLI, Quality of service for triple play services in heterogeneous networks. 88 p. (180 p.) Yhteenveto 1 p. 2007.
81 NERI, FERRANTE, Fitness diversity adaptation in memetic algorithms. 80 p. (185 p.) Yhteenveto 1 p. 2007.
82 KURHINEN, JANI, Information delivery in mobile peer-to-peer networks. 46 p. (106 p.) Yhteenveto 1 p. 2007.
83 KILPELÄINEN, TURO, Genre and ontology based business information architecture framework (GOBIAF). 74 p. (153 p.) Yhteenveto 1 p. 2007.
84 YEVSEYEVA, IRYNA, Solving classification problems with multicriteria decision aiding approaches. 182 p. Yhteenveto 1 p. 2007.
85 KANNISTO, ISTO, Optimized pricing, QoS and segmentation of managed ICT services. 45 p. (111 p.) Yhteenveto 1 p. 2007.
86 GORSHKOVA, ELENA, A posteriori error estimates and adaptive methods for incompressible viscous flow problems. 72 p. (129 p.) Yhteenveto 1 p. 2007.
87 LEGRAND, STEVE, Use of background real-world knowledge in ontologies for word sense disambiguation in the semantic web. 73 p. (144 p.) Yhteenveto 1 p. 2008.
88 HÄMÄLÄINEN, NIINA, Evaluation and measurement in enterprise and software architecture management. - Arviointi ja mittaaminen kokonais- ja ohjelmistoarkkitehtuurien hallinnassa. 91 p. (175 p.) Yhteenveto 1 p. 2008.
89 OJALA, ARTO, Internationalization of software firms: Finnish small and medium-sized software firms in Japan. 57 p. (180 p.) Yhteenveto 2 p. 2008.
90 LAITILA, ERKKI, Symbolic Analysis and Atomistic Model as a Basis for a Program Comprehension Methodology. 321 p. Yhteenveto 3 p. 2008.
91 NIHTILÄ, TIMO, Performance of Advanced Transmission and Reception Algorithms for High Speed Downlink Packet Access. 93 p. (186 p.) Yhteenveto 1 p. 2008.
92 SETÄMAA-KÄRKKÄINEN, ANNE, Network connection selection-solving a new multiobjective optimization problem. 52 p. (111p.) Yhteenveto 1 p. 2008.
93 PULKKINEN, MIRJA, Enterprise architecture as a collaboration tool. Discursive process for enterprise architecture management, planning and development. 130 p. (215 p.) Yhteenveto 2 p. 2008.
94 PAVLOVA, YULIA, Multistage coalition formation game of a self-enforcing international environmental agreement. 127 p. Yhteenveto 1 p. 2008.
95 NOUSIAINEN, TUULA, Children's involvement in the design of game-based learning environments. 297 p. Yhteenveto 2 p. 2008.
96 KUZNETSOV, NIKOLAY V., Stability and oscillations of dynamical systems. Theory and applications. 116 p. Yhteenveto 1 p. 2008.
97 KHRIYENKO, OLEKSIY, Adaptive semantic Web based environment for web resources. 193 p. Yhteenveto 1 p. 2008.
98 TIRRONEN, VILLE, Global optimization using memetic differential evolution with applications to low level machine vision. 98 p. (248 p.) Yhteenveto 1 p. 2008.
99 VALKONEN, TUOMO, Diff-convex combinations of Euclidean distances: A search for optima. 148 p. Yhteenveto 1 p. 2008.
100 SARAFANOV, OLEG, Asymptotic theory of resonant tunneling in quantum waveguides of variable cross-section. 69 p. Yhteenveto 1 p. 2008.
101 POZHARSKIY, ALEXEY, On the electron and phonon transport in locally periodical waveguides. 81 p. Yhteenveto 1 p. 2008.
102 AITTOKOSKI, TIMO, On challenges of simulation-based globaland multiobjective optimization. 80 p. (204 p.) Yhteenveto 1 p. 2009.
103 YALAHO, ANICET, Managing offshore outsourcing of software development using the ICT-supported unified process model: A cross-case analysis. 91 p. (307 p.) Yhteenveto 4 p. 2009.
104 KOLLANUS, SAMI, Tarkastuskäytänteiden kehittäminen ohjelmistoja tuottavissa organisaatioissa. - Improvement of inspection practices in software organizations. 179 p. Summary 4 p. 2009.
105 LEIKAS, JAANA, Life-Based Design. 'Form of life' as a foundation for ICT design for older adults. - Elämälähtöinen suunnittelu. Elämänmuoto ikääntyville tarkoitettujen ICT tuotteiden ja palvelujen suunnittelun lähtökohtana. 218 p. (318 p.) Yhteenveto 4 p. 2009.

106 VASILYEVA, EKATERINA, Tailoring of feedback in web-based learning systems: Certitude-based assessment with online multiple choice questions. 124 p. (184 p.) Yhteenveto 2 p. 2009.

107 KUDRYASHOVA, ELENAV., Cycles in continuous and discrete dynamical systems. Computations, computer assisted proofs, and computer experiments. 79 p. (152 p.) Yhteenveto 1 p. 2009.

108 BLACKLEDGE, JONATHAN, Electromagnetic scattering and inverse scattering solutions for the analysis and processing of digital signals and images. 297 p. Yhteenveto 1 p. 2009.

109 IVANNIKOV, ANDRIY, Extraction of event-related potentials from electroencephalography data. - Herätepotentiaalien laskennallinen eristäminen EEG-havaintoaineistosta. 108 p. (150 p.) Yhteenveto 1 p. 2009.

110 KALYAKIN, IGOR, Extraction of mismatch negativity from electroencephalography data. - Poikkeavuusnegatiivisuuden erottaminen EEG-signaalista. 47 p. (156 p.) Yhteenveto 1 p. 2010.

111 HEIKKILÄ, MARIKKA, Coordination of complex operations over organisational boundaries. 265 p. Yhteenveto 3 p. 2010.

112 FEKETE, GÁBOR, Network interface management in mobile and multihomed nodes. 94 p. (175 p.) Yhteenveto 1 p. 2010.

113 KUJALA, TUOMO, Capacity, workload and mental contents - Exploring the foundations of driver distraction. 146 p. (253 p.) Yhteenveto 2 p. 2010.

114 LUGANO, GIUSEPPE, Digital community design - Exploring the role of mobile social software in the process of digital convergence. 253 p. (316 p.) Yhteenveto 4 p. 2010.

115 KAMPYLIS, PANAGIOTIS, Fostering creative thinking. The role of primary teachers. - Luovaa ajattelua kehittämässä. Alakoulun opettajien rooli. 136 p. (268 p.) Yhteenveto 2 p. 2010.

116 TOIVANEN, JUKKA, Shape optimization utilizing consistent sensitivities. - Muodon optimointia konsistenttien herkkyyksien avulla. 55 p. (130 p.)Yhteenveto 1 p. 2010.