J. Math. Study
doi: 10.4208/jms.v52n3.19.01

Vol. **52**, No. 3, pp. 227-243
September 2019

# Computation of Shape Derivatives in Electromagnetic Shaping by Algorithmic Differentiation

Karsten Eppler[1], Helmut Harbrecht[2,*], Sebastian Schlenkrich[3] and Andrea Walther[4]

[1] *Institute of Numerical Mathematics, Technische Universität Dresden, 01062 Dresden, Germany.*
[2] *Department of Mathematics and Computer Science, University of Basel, 4001 Basel, Switzerland.*
[3] *d-fine GmbH, Opernplatz 2, 60313 Frankfurt am Main, Germany.*
[4] *Institute of Mathematics, Paderborn University, 33098 Paderborn, Germany.*

Received September 24, 2018; Accepted February 26, 2019

**Abstract.** Shape optimization based on analytical shape derivatives is meanwhile a well-established tool in engineering applications. For an appropriate discretization of the underlying problem, the technique of algorithmic differentiation can also be used to provide a discrete analogue of the analytic shape derivative. The present article is concerned with the comparison of both types of gradient calculation and their effects on a gradient-based optimization method with respect to accuracy and performance, since so far only a few attempts have been made to compare these approaches. For this purpose, the article discusses both techniques and analyses the obtained numerical results for a generic test case from electromagnetic shaping. Since good agreement of both methods is found, algorithmic differentiation seems to be worthwhile to be used also for more demanding shape optimization problems.

**AMS subject classifications**: 49M25, 49Q10, 78M15

**Key words**: Exterior electromagnetic shaping, shape optimization, boundary element method, automatic differentiation.

## 1 Introduction and motivation

Gradient-based optimization methods are frequently used in engineering applications. In particular, shape optimization is quite indispensable for designing and constructing in-

*Corresponding author. *Email addresses:* `karsten.eppler@tu-dresden.de` (K. Eppler), `helmut.harbrecht@unibas.ch` (H. Harbrecht), `sebastian.schlenkrich@d-fine.de` (S. Schlenkrich), `andrea.walther@uni-paderborn.de` (A. Walther) dustrial components. Many problems that arise in application, particularly in structural mechanics and in the optimal control of distributed parameter systems, can be formulated as the minimization of functionals defined over a class of admissible domains.

Analytic shape optimization methods are known to be an efficient numerical tool for free boundary computations in electromagnetic shaping, see e.g. [4, 6, 11, 29, 34–36]. In [15, 16], the first two authors developed first and second order algorithms for elliptic shape optimization problems with additional functional constraints. A wavelet-based boundary element method was used for the computation of the objective and related first and second order shape derivatives. These algorithms have successfully been applied in [17] for exterior electromagnetic shaping.

As an alternative approach to provide derivatives one may use Algorithmic Differentiation (AD). AD yields exact derivative information for a function evaluation given as computer code. A comprehensive introduction to AD can be found in [23].

In the context of optimal control problems, the forward mode of AD can be seen as a discrete version of the sensitivity approach. Conversely, the reverse mode of AD involves a discrete adjoint somehow related to the continuous adjoint equation. Despite the fact that these parallels have already been hinted at in [22], a detailed theoretical analysis of the relations between the exact discrete derivatives provided by AD and the corresponding continuous derivative formulation is only available for optimal control problems based on ODEs, see e.g. [19, 41]. The influences of the different derivative information, i.e., either the exact discrete derivatives of the evaluation program, provided by AD, or the continuous derivatives provided by the adjoint equations, on the whole optimization problem were studied in [21] for small ODE-based optimization problems. In [31], the gradient computation for a rather small two-dimensional optimization problem based on Navier-Stokes equations was considered using a Taylor-Hood finite element discretization in space and an implicit Euler scheme in time.

The aim of the present article is to use first the analytic gradient derived from the analytical setting. For the implementation of both, the objective and the analytical gradient, a wavelet-based boundary element method is chosen. Subsequently, it will be shown that also the application of AD is feasible for this problem of medium to large-scale size. Finally, the whole optimization process obtained with the analytic gradient is compared with similar computations based on gradients generated by the AD tool ADOL-C [42].

This article is organized as follows. Section 2 repeats the main aspects about the underlying model and analytic gradients in electromagnetic shaping. The discretization of the unknown shape, the numerical solution of the boundary integral equation by a wavelet-based boundary element method, and the optimization method for the resulting finite dimensional optimization problem are discussed in Section 3. In Section 4, we recall basic facts about the concept of algorithmic differentiation and its implementation. Finally, Section 5 summarizes our comparison tests with various respects. In Section 6, we state concluding remarks.

## 2 Modelling and analytic shape calculus

### 2.1 Physical model

Let a cylindric vertical column of molten liquid metal with planar, simply connected cross section $\Omega\subset\mathbb{R}^2$ and fixed area $V_0$ falling down in an electromagnetic field which is created by vertical conductors. The frequency of the imposed current is very high so that the magnetic field does not penetrate into the metal and the electromagnetic forces are reduced to the magnetic pressure acting on the interface. They are in equilibrium with surface tension forces of the matter. Following e.g. [4], the unknown equilibrium shape is characterized by an overdetermined boundary value problem for the magnetostatic Maxwell equation, see also [17,29,36] for more details.

In order to state the respective boundary value problem, we shall introduce the following quantities. Geometric quantities are the outward unit normal $\mathbf{n}=[n_1,n_2]^\top$ of $\Omega$, the curvature $\kappa$ of $\Gamma:=\partial\Omega$, and the volume of the domain $V(\Omega)=|\Omega|$. Physical quantities are the magnetic permeability $\mu_0$, the surface tension of the liquid $\sigma$, the magnetic field $\mathbf{B}$, and the density current vector $\mathbf{j}_0$. As one readily verifies, in the situation under consideration, the density current vector takes the form $\mathbf{j}_0(\mathbf{x})=[0,0,j(\mathbf{x})]^\top$ with $\mathbf{x}=[x_1,x_2]^\top\in\mathbb{R}^2$, and, hence, $\mathbf{B}(\mathbf{x})=[B_1(\mathbf{x}),B_2(\mathbf{x}),0]^\top$.

In case of the equilibrium, the cross section $\Omega$ of the liquid metal and the related magnetic field $\mathbf{B}$ satisfy the system of equations

$$\nabla\times\mathbf{B}=\mu_0\mathbf{j}_0 \qquad \text{in } (\mathbb{R}^2\setminus\overline{\Omega})\times\mathbb{R}, \tag{2.1}$$

$$\nabla\cdot\mathbf{B}=0 \qquad \text{in } (\mathbb{R}^2\setminus\overline{\Omega})\times\mathbb{R}, \tag{2.2}$$

$$\mathbf{B}\cdot\begin{bmatrix}\mathbf{n}\\0\end{bmatrix}=0 \qquad \text{on } \Gamma\times\mathbb{R}, \tag{2.3}$$

$$\frac{\|\mathbf{B}\|^2}{2\mu_0}+\sigma\kappa=p \qquad \text{constant on } \Gamma\times\mathbb{R}, \tag{2.4}$$

$$\mathbf{B}=\mathcal{O}(\|\mathbf{x}\|^{-2}) \qquad \text{as } \|\mathbf{x}\|\to\infty, \tag{2.5}$$

provided that $\Omega$ fullfils the volume constraint $V(\Omega)=V_0$. The unknowns of the problem are the boundary $\Gamma=\partial\Omega$ and the pressure constant $p$ in the equilibrium condition (2.4).

### 2.2 Shape optimization problem

Introducing a scalar potential $u:\Omega\mapsto\mathbb{R}$, condition (2.2) implies the ansatz

$$\mathbf{B}(\mathbf{x})=\mu_0\nabla\times\begin{bmatrix}0\\0\\u(\mathbf{x})\end{bmatrix}=\mu_0\begin{bmatrix}\frac{\partial u}{\partial x_2}(\mathbf{x})\\-\frac{\partial u}{\partial x_1}(\mathbf{x})\\0\end{bmatrix}.$$

Hence, in view of (2.1) and (2.5), the scalar potential $u$ has to satisfy the following boundary value problem in the exterior domain $\Omega^c := \mathbb{R}^2 \setminus \overline{\Omega}$:

$$\begin{aligned} -\Delta u &= j && \text{in } \Omega^c, \\ u &= 0 && \text{on } \Gamma, \\ u &= \mathcal{O}(1) && \text{as } \|\mathbf{x}\| \to \infty, \\ \|\nabla u\| &= \mathcal{O}(\|\mathbf{x}\|^{-2}) && \text{as } \|\mathbf{x}\| \to \infty. \end{aligned} \tag{2.6}$$

After rescaling physical constants, determing the free surface $\Gamma^\star = \partial\Omega^\star$ is equivalent to the shape optimization problem

$$E(\Omega) \to \min_{\Omega \subset \mathbb{R}^2} \quad \text{subject to} \quad V(\Omega) = V_0, \tag{2.7}$$

where the (scaled) total energy is given by

$$E(\Omega) = J(\Omega) + AP(\Omega).$$

Here, the volume and the perimeter of a domain are defined as usual

$$V(\Omega) = \int_\Omega \mathrm{d}\mathbf{x}, \qquad P(\Omega) = \int_\Gamma \mathrm{d}\sigma_{\mathbf{x}},$$

while the Dirichlet energy computes as

$$J(\Omega) = -\int_\Omega \|\nabla u\|^2 \mathrm{d}\mathbf{x}$$

with $u$ being the solution of (2.6).

The magnetic field is assumed to be generated by a current flow in vertical live wires at position $\mathbf{x}_i$ and of amperage $\alpha_i$ each, where $i = 1,\ldots,M$. The sign of $\alpha_i$ is determined by the direction of the currents in the particular wires. We model the wires by constant densities on circular conductors with finite diameter $\varepsilon > 0$, that is

$$j(\mathbf{x}) = \sum_{i=1}^{M} \frac{\alpha_i}{\pi\varepsilon^2} \chi_{B_\varepsilon(\mathbf{x}_i)}, \qquad \mathbf{x} \in \mathbb{R}^2, \tag{2.8}$$

where

$$B_\varepsilon(\mathbf{x}_i) := \left\{ \mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x} - \mathbf{x}_i\| \le \varepsilon \right\}.$$

Motivated from a physical point of view, we introduce the additional assumption $\sum_{i=1}^M \alpha_i = 0$, which means that the overall flow in both directions is equilibrated.

## 2.3 Computation of the objective

The exterior boundary value problem with respect to $u$ can be reformulated as a boundary integral equation, provided that we find an appropriate Newton potential $N_j \in C^{1,1}(\mathbb{R})$ which satisfies the equation

$$\begin{aligned} -\Delta N_j &= j && \text{in } \mathbb{R}^2, \\ N_j &= \mathcal{O}(1) && \text{as } \|\mathbf{x}\| \to \infty, \\ \|\nabla N_j\| &= \mathcal{O}(\|\mathbf{x}\|^{-2}) && \text{as } \|\mathbf{x}\| \to \infty. \end{aligned} \tag{2.9}$$

Then, the ansatz

$$u = N_j + v \tag{2.10}$$

would yield the problem of seeking a harmonic function $v$ which satisfies

$$\begin{aligned} \Delta v &= 0 && \text{in } \Omega, \\ v &= -N_j && \text{on } \Gamma, \\ v &= \mathcal{O}(1) && \text{as } \|\mathbf{x}\| \to \infty, \\ \|\nabla v\| &= \mathcal{O}(\|\mathbf{x}\|^{-2}) && \text{as } \|\mathbf{x}\| \to \infty. \end{aligned} \tag{2.11}$$

We recall the following result from [17].

**Lemma 2.1.** *Let the density $j$ be modeled in accordance with* (2.8). *Then, a Newton potential $N_j \in C^{1,1}(\mathbb{R})$ which satisfies* (2.9) *is given by*

$$N_j = \sum_{i=1}^{M} N_{i,j}, \quad \text{where} \quad N_{i,j}(\mathbf{x}) = \begin{cases} \frac{\alpha_i}{4\pi}\left[1 - \frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{\varepsilon^2} - 2\log\varepsilon\right], & \mathbf{x} \in B_\varepsilon(\mathbf{x}_i), \\ -\frac{\alpha_i}{2\pi}\log\|\mathbf{x}-\mathbf{x}_i\|, & \mathbf{x} \notin B_\varepsilon(\mathbf{x}_i). \end{cases} \tag{2.12}$$

*In particular, there holds*

$$J(\Omega) = \int_\Omega -j(\mathbf{x})u(\mathbf{x})\mathrm{d}\mathbf{x} = \int_{\operatorname{supp} j} -j(\mathbf{x})N_j(\mathbf{x})\mathrm{d}\mathbf{x} + \int_\Gamma N_j(\mathbf{x})\frac{\partial u}{\partial \mathbf{n}}(\mathbf{x})\mathrm{d}\sigma. \tag{2.13}$$

**Remark 2.1.** By Lemma 2.1, the objective becomes numerically computable. Especially, the behaviour of $N_j$ at infinity as well as in a neighbourhood of $\Gamma$ is *shape independent* of $\varepsilon$ provided that $\min_{i=1}^{M}\operatorname{dist}(\Omega,\mathbf{x}_i) > \varepsilon$. This implies that the integral over $\operatorname{supp} j$ on the right-hand side of (2.13) is independent of the particular domain $\Omega$ and can be neglected in computations.

## 2.4 Computation of the shape gradient

We next recall the shape derivatives of all functionals which appear in problem (2.7). Applying, for example, the perturbation of identity approach by a smooth perturbation field $\mathbf{U} \in \left[C^2(\mathbb{R})\right]^2$, i.e., using

$$\Omega_t = \left\{\mathbf{x} + t\mathbf{U}(\mathbf{x}) : \mathbf{x} \in \Omega\right\},$$

the solution $u_t$ and the shape functionals $J(\Omega_t)$, $V(\Omega_t)$, $P(\Omega_t)$ are well defined provided that $t$ is sufficiently small. By using standard results (cf. [39]), we arrive at the following expression for the directional derivatives:

$$\begin{aligned} \delta J(\Omega)[\mathbf{U}] &= \int_\Gamma \langle \mathbf{U},\mathbf{n}\rangle \|\nabla u\|^2 \mathrm{d}\sigma = \int_\Gamma \langle \mathbf{U},\mathbf{n}\rangle \left[\frac{\partial u}{\partial \mathbf{n}}\right]^2 \mathrm{d}\sigma, \\ \delta P(\Omega)[\mathbf{U}] &= \int_\Gamma \langle \mathbf{U},\mathbf{n}\rangle \kappa \,\mathrm{d}\sigma, \quad \delta V(\Omega)[\mathbf{U}] = \int_\Gamma \langle \mathbf{U},\mathbf{n}\rangle \mathrm{d}\sigma. \end{aligned} \tag{2.14}$$

We like to emphasize that only the Neumann data $\partial u/\partial \mathbf{n}$ of the state enter the shape gradient of the Dirichlet energy $J(\Omega)$. They need also to be computed for evaluating the Dirichlet energy itself by (2.13), compare also Remark 2.1. Therefore, the additional effort to compute the shape gradient is negligible.

Applying a Lagrange formalism by means of

$$L(\Omega,\lambda) := E(\Omega) - \lambda\big(V(\Omega) - V_0\big)$$

yields immediately the following theorem.

**Theorem 2.1** (Necessary condition). *If $\Omega^\star$ is optimal, then there exists a $\lambda^\star \neq 0$ such that*

$$\left\{\|\nabla u\|^2 + A\kappa - \lambda^\star\right\}\Big|_{\Gamma^\star} \equiv 0 \quad \textit{and} \quad |\Omega^\star| = V(\Omega^\star) = V_0.$$

**Remark 2.2.** Theorem 2.1 is the key theorem to establish the equivalence of the free surface problem (2.1)–(2.5) to the shape optimization problem (2.7): The necessary condition is just the additional equilibrium condition for the overdetermined boundary value problem. In particular, the Lagrange multiplier $\lambda^\star$ determines the (a-priori) unknown equilibrium pressure, which is constant along $\Gamma^\star$.

## 3 Discretization and optimization methods

### 3.1 Boundary variation and finite approximation of the shapes

The outer security set of the shape problem (2.7) is given as $D = \mathbb{R}^2$ (the *hold all* set). As admissible domains, we consider all bounded domains $\Omega \subset D \subset \mathbb{R}^2$ that are star-shaped with respect to $\mathbf{0}$ and of smoothness $C^2$.

Our setup enables us to identify the domain $\Omega \in C^2$ with a parametric representation of its boundary $\Gamma = \partial\Omega$. Due to the star-shapedness, this parametric representation can be chosen in polar coordinates

$$\Gamma := \left\{\gamma(\phi) = r(\phi)\mathbf{e}_r(\phi) : \ \phi \in [0,2\pi]\right\},$$

where $r \in C^2_{\text{per}}([0,2\pi])$ is a positive function with $r(\phi) \geq \delta > 0$ for all $\phi \in [0,2\pi]$,

$$C^2_{\text{per}}([0,2\pi]) = \left\{r \in C^2([0,2\pi]) : \ r^{(i)}(0) = r^{(i)}(2\pi), \ i = 0,1,2\right\}, \tag{3.1}$$

and $\mathbf{e}_r(\phi)=[\cos\phi,\sin\phi]^\top$ denotes the unit vector in the radial direction with respect to $\Omega$.

As standard variation for perturbed domains $\Omega_\varepsilon$ and boundaries $\Gamma_\varepsilon$, respectively, we introduce a function $\delta r\in C^2_{\text{per}}([0,2\pi])$ and set

$$r_\varepsilon(\phi)=r(\phi)+\varepsilon\delta r(\phi),$$

where $\gamma_\varepsilon(\phi)=r_\varepsilon(\phi)\mathbf{e}_r(\phi)$ is always a Jordan curve.

**Remark 3.1.** The main advantage of the present setup is an embedding of the shape problem into a Banach space setting. That is *both*, the shapes and their increments, can be viewed as elements of $C^2_{\text{per}}([0,2\pi])$. Notice that the analytical shape gradients (2.14) can be expressed in polar coordinates, too (cf. [17]).

We shall next proceed with a Ritz-Galerkin approximation of the shape optimization problem (2.7) under consideration. Namely, we can express the smooth function $r\in C^2_{\text{per}}([0,2\pi])$ by the Fourier series

$$r(\phi)=a_0+\sum_{n=1}^{\infty}a_n\cos(n\phi)+a_{-n}\sin(n\phi).$$

Hence, it is reasonable to take the truncated Fourier series

$$r_N(\phi)=a_0+\sum_{n=1}^{N}a_n\cos(n\phi)+a_{-n}\sin(n\phi) \tag{3.2}$$

as approximation of $r$. We mention that also other boundary representations like B-splines might be considered as well. The advantages of our approach is an exponential convergence $r_N\to r$ if the shape is analytical, see also [18].

## 3.2 Augmented Lagrangian method

Since $r_N$ from (3.2) has the $2N+1$ degrees of freedom $a_{-N},a_{1-N},\ldots,a_N$, we arrive at a finite dimensional optimization problem in the open set

$$A_N:=\{a_{-N},a_{1-N},\ldots,a_N\in\mathbb{R}:\ r_N(\phi)>0,\phi\in[0,2\pi]\}\subset\mathbb{R}^{2N+1}.$$

Thus, via the identification $r_N\Longleftrightarrow\Omega_N$, the finite dimensional approximation of the shape optimization problem (2.7) reads as

$$E(\Omega_N)\to\min\quad\text{subject to}\quad V(\Omega_N)=V_0. \tag{3.3}$$

The associated gradients have to be computed with respect to all directions

$$\delta r=\cos(N\phi),\cos\big((N-1)\phi\big),\ldots,\sin\big((N-1)\phi\big),\sin(N\phi).$$

For the solution of the discrete shape optimization (3.3), we consider an augmented Lagrangian method for equality constraints

$$L_c(\Omega,\lambda)=E(\Omega)-\lambda\big(V(\Omega)-V_0\big)+\frac{c}{2}\big(V(\Omega)-V_0\big)^2, \tag{3.4}$$

where the constant $c$ is appropriately chosen (we use $c=10$ in our numerical experiments). The optimization algorithm for the augmented Lagrangian method reads as follows:

- initialization: choose initial guesses $\lambda^{(0)}$ and $\Omega^{(0)}$ for $\lambda^\star$ and $\Omega^\star$, respectively,
- inner iteration: solve
$$\Omega^{(n+1)}=\operatorname{argmin} L_c(\Omega,\lambda^{(n)})$$
with initial guess $\Omega^{(n)}$,
- outer iteration: update
$$\lambda^{(n+1)}=\lambda^{(n)}-c\big(V(\Omega)-V_0\big).$$

In the inner iteration, we employ a quasi-Newton scheme, which is updated by the inverse BFGS-rule without damping. The step size is determined by a quadratic line-search. We refer the reader to, e.g., [10, 25] for more details on the used numerical optimization algorithms.

## 3.3 Boundary integral equation

In view of (2.13) and (2.14), it suffices to provide the normal derivative $\partial u/\partial\mathbf{n}$ of the state in order to compute the objective and its gradient, i.e., for performing a first order optimization method.

The ansatz (2.10) leads to the normal derivative $\partial u/\partial\mathbf{n}$ according to

$$\frac{\partial u}{\partial\mathbf{n}}=\frac{\partial v}{\partial\mathbf{n}}+\frac{\partial N_j}{\partial\mathbf{n}}$$

with the Newton potential $N_j$ defined in (2.12) and $v$ satisfying the boundary value problem (2.11). We introduce the modified *single layer operator* $\mathcal{V}$ and the *double layer operator* $\mathcal{K}$ defined by

$$(\mathcal{V}u)(\mathbf{x}):=-\frac{1}{2\pi}\int_\Gamma \log\|\mathbf{x}-\mathbf{y}\|\left\{u(\mathbf{y})-\int_\Gamma\frac{u(\mathbf{z})}{|\Gamma|}\mathrm{d}\sigma_{\mathbf{z}}\right\}\mathrm{d}\sigma_{\mathbf{y}}+\int_\Gamma\frac{u(\mathbf{z})}{|\Gamma|}\mathrm{d}\sigma_{\mathbf{z}},$$
$$(\mathcal{K}u)(\mathbf{x}):=\frac{1}{2\pi}\int_\Gamma\frac{\langle\mathbf{n}_{\mathbf{y}},\mathbf{x}-\mathbf{y}\rangle}{\|\mathbf{x}-\mathbf{y}\|^2}u(\mathbf{y})\mathrm{d}\sigma_{\mathbf{y}}.$$

Then, the normal derivative of $v$ is given by the Dirichlet-to-Neumann map

$$\mathcal{V}\frac{\partial v}{\partial\mathbf{n}}=\Big(\frac{1}{2}-\mathcal{K}\Big)N_j. \tag{3.5}$$

Notice that $\mathcal{V}: H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma)$ is a symmetric, elliptic, and continuous operator while $\frac{1}{2} - \mathcal{K}: H^{1/2}(\Gamma) \to H^{1/2}(\Gamma)$ is continuous. We refer to, e.g., the books [37,40] for the variational formulation and further details of the operators $\mathcal{V}$ and $\mathcal{K}$.

## 3.4 Boundary element method

Boundary element methods provide a common tool for the solution of boundary integral equations, see, for example, [32, 33]. In general, cardinal B-splines are used as ansatz functions in the Galerkin formulation. But discretizing the boundary integral equation (3.5) with respect to such *single-scale bases* yields densely populated system matrices. In combination with the ill-posedness of the single layer operator, the numerical solution is of at least quadratic complexity.

The crucial idea of the wavelet-based boundary element method is a change of bases, i.e., applying appropriate wavelet bases instead of the traditional single-scale bases. On the one hand, based on the well-known norm equivalences of wavelet bases, the diagonals of the system matrices define optimal preconditioners, cf. [7, 9, 38]. On the other hand, the resulting quasi-sparse system matrices can be compressed without loss of accuracy such that the complexity for the solution of the boundary integral equations becomes of linear complexity, cf. [8,26].

For all the details of the current implementation, including error estimates, we refer the reader to [17, 26]. In particular, to achieve the optimal rate of convergence, we discretize the Neumann data by piecewise constant wavelets and the Dirichlet data by piecewise linear wavelets. Finally, we end up with a discretized system of equations

$$\mathbf{V}_L \mathbf{v}_L = \Big(\frac{1}{2}\mathbf{M}_L - \mathbf{K}_L\Big)\mathbf{G}_L^{-1}\mathbf{u}_L,$$

where $L$ refers to the discretization level corresponding to $2^L$ boundary elements. This system can be solved by e.g. an iterative conjugate gradient method. We like to stress that the stiffness matrices $\mathbf{V}_L$, $\mathbf{K}_L$, the mass matrix $\mathbf{G}_L$ with respect to piecewise linear ansatz and test functions and the mass matrix $\mathbf{M}_L$ with respect to piecewise linear ansatz and piecewise constant test functions, as well as the right-hand side $\mathbf{u}_L$ depend on the actual shape.

**Remark 3.2.** We want to mention that, in the present approach, the discretization of the shape and the discretization of the state equation are decoupled. Consequently, they can be chosen independently according to accuracy requirements.

# 4 Basics on algorithmic differentiation and ADOL-C

Suppose that the function $\mathbf{F}: \mathbb{R}^n \mapsto \mathbb{R}^m$, $\mathbf{y} = \mathbf{F}(\mathbf{x})$ which describes an arbitrary algebraic mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$ is defined by an evaluation procedure in a high-level computer language like Fortran or C. In the application considered here, **F** is equal to the augmented Lagrangian function $L_c(\Omega,\lambda)$, including the computation of the discretized total energy $E(\Omega)$ and of the discretized volume $V(\Omega)$. Therefore, $n=2N+1$ and $m=1$.

The technique of algorithmic differentiation provides derivative information of arbitrary order for the code segment in the computer that evaluates $\mathbf{F}(\mathbf{x})$ within working accuracy [24]. For this purpose, the basic differentiation rules such as, e.g., the product rule are applied to each statement of the given code segment. This local derivative information is then combined by the chain rule to calculate the overall derivatives. Hence, the code is decomposed into a long sequence of simple evaluations, e.g., additions, multiplications, and calls to elementary functions such as $\sin(x)$ or $\exp(x)$, the derivatives of which can easily be calculated. Exploiting the chain rule yields the derivatives of the whole sequence of statements with respect to the input variables.

The so-called forward mode of AD propagates the derivatives together with the function evaluation. Alternatively, one may propagate the derivative information from the dependents **y** to the independents **x**. This yields the so-called reverse mode of AD. Over the last decades, extensive research activities led to a thoroughly understanding and analysis of the two basic modes of AD, where the complexity results are based on the operation count $\mathcal{O}_\mathbf{F}$ of the underlying vector function **F**. Using the forward mode of AD, one *column* of the Jacobian $\nabla\mathbf{F}$ can be calculated at no more than five times $\mathcal{O}_\mathbf{F}$ [23]. One *row* of $\nabla\mathbf{F}$, e.g. the gradient of a scalar-valued component function of **F**, is theoretically obtained also at no more than five times $\mathcal{O}_\mathbf{F}$ by using the reverse mode in its basic form [23]. It is important to note that this bound for the reverse mode is completely independent of the number $n$ of input variables. Hence, for the application considered here, the reverse mode is especially attractive since we have a scalar-valued target function with $m=1$.

Beside the theoretical foundation, numerous AD tools have been developed, e.g. TAF [20], ADOL-C [42], Tapenade [27], for the Automatic Differentiation of Fortran and C/C++ codes. Several AD tools have matured over the past years to a state that they are able to produce tangent and adjoint linearisations of large and unstructured Fortran and C/C++ codes. Additional information about tools and literature on AD can be found on the web-page of the AD-community `www.autodiff.org`.

So far the AD tools use either the technique of source transformation or the approach of operator overloading. In source transformation, given a source code for evaluating a function, the AD tool generates a new source code for evaluating the required derivative information. This technique is especially well suited for the automatic differentiation of Fortran 77 codes because this programming language allows an appropriate analysis of the code. Subsequently, a new source file is generated for computing the desired derivative objects. It is used, for example, by the packages TAF and Tapenade. It is considerably more difficult to obtain the required information for an efficient AD implementation, e.g., a dependency analysis, for C/C++ codes, due to language features like pointers, class definitions, and templates. Therefore, currently an implementation of AD based on operator overloading is better suited for programs written in C or C++.

The key ingredient of AD by overloading is the concept of an *active variable* that contains the usual real part that corresponds to the double computation performed during the function evaluation and a logging facility. All variables that may be considered as differentiable quantities at some time during the program execution must be of an active type. These are all variables that lie on the way from the input variables, i.e., the independents, to the output variable. In data flow terminology, the set of active variable names must contain all its successors in the dependency graph. Variables that do not depend on the independent variables but enter the calculation, for example, as parameters, may remain one of the *passive* types `double`, `float`, or `int`.

The derivative calculation is then based on an internal function representation, which is created during a separate so-called tracing phase of the function evaluation. All calculations, involving active variables that occur during the tracing, are recorded on a sequential data set called *trace*. That is, for each operation, the involved variables, the operation itself and, if required, additional real values are stored. Once this internal function representation is generated, appropriate drivers can be applied for the derivative calculation. The AD-tools ADOL-C [42] and CppAD [2], for example, are based on this technique.

For the optimization results presented in this article, we use the AD-tool ADOL-C to provide the exact discrete derivative information for the evaluation program. Due to the particular discretization of the total energy by a boundary element method, the derivatives computed by ADOL-C, i.e., the gradient of the discretized augmented Lagrangian function $L_c(\Omega,\lambda)$ with respect to the input values $a_{-N},a_{1-N},\dots,a_N$, correspond to the computation of the shape derivatives given in (2.14). Details on the agreement, the corresponding memory requirement, and runtime effects are discussed in the subsequent section.

## 5 Numerical results

### 5.1 Benchmark problem

The numerical shape optimization algorithm has been implemented as C++ program. For our numerical tests, we set $N=32$, which yields 65 design parameters, cf. Subsection 3.1, and $L=10$, which yields 1024 boundary elements, cf. Subsection 3.4.

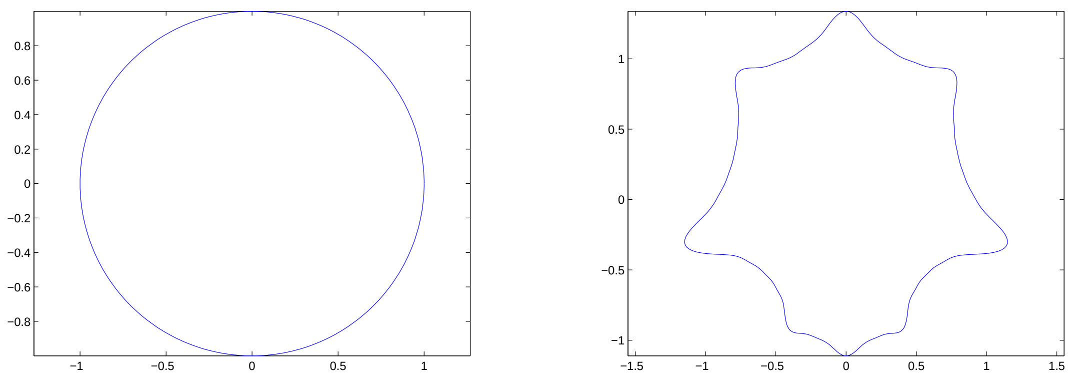
The unit circle is chosen as initial shape as seen in the left plot of Figure 1. For the augmentation of the Lagrangian, the setting $c=10$ leads to satisfying results, cf. (3.4).

Error estimates for the numerically computed objective and shape gradient have been proven in [17]. Namely, it holds

$$\begin{aligned} |J(\Omega)-J_L(\Omega)| &\lesssim h_L^3, \\ |\delta J(\Omega)[\delta r]-\delta J_L(\Omega)[\delta r]| &\lesssim h_L^2, \end{aligned}$$

where $h=2^{-L}$ refers to the mesh size of the boundary element discretization.

Since the shape optimization problem at hand is *well-posed*, see [30], the convergence theory from [18] applies. As known from the literature, optimal solutions of problem



initial shape final shape

Figure 1: Shape deformation of the test problem.

(2.6), (2.7) are analytical shapes, cf. [29]. Thus, the Fourier series (3.2) provides an *exponential* approximation. Consequently, a reasonable overall accuracy can be guaranteed for the particular discretization. Nevertheless, we have to deal with about 90 million operations per function evaluation.

## 5.2 Case studies for optimization runs

We shall compare the analytic shape gradient and the discrete shape gradient provided by algorithmic differentiation. In some cases, the usage of these different derivative objects yields also different numerical results, see, e.g., [1]. For this reason, we compare here the result of the optimization procedure using either analytic gradient information or discrete gradients provided by AD.

The optimization is performed by 50 outer iterations with five inner iterations each. The graph of the optimized shape is displayed in the right plot of Figure 1.

We performed an optimization run with analytic gradient evaluation and another one with gradients generated by AD. The convergence history of the objective function for the optimization run using AD gradients is shown in Figure 2. Here, the objective function value is plotted for all inner and outer iterations. The result is almost equal to the optimization with analytic gradient evaluation. Consequently, the discrepancy of the objective function values, obtained during the optimization process by applying either analytic gradients or AD gradients, almost vanishes as seen in Figure 3. This figure shows the absolute difference of the function value, computed with AD gradients, and the function value, computed with analytic gradients, for all inner and outer iterations. As can be seen, the obtained results during the optimization coincide up to numerical noise.

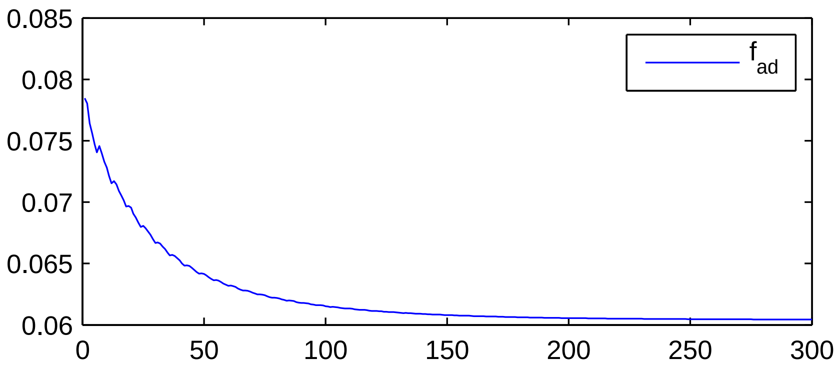


Figure 2: Convergence history of AD-based optimization.

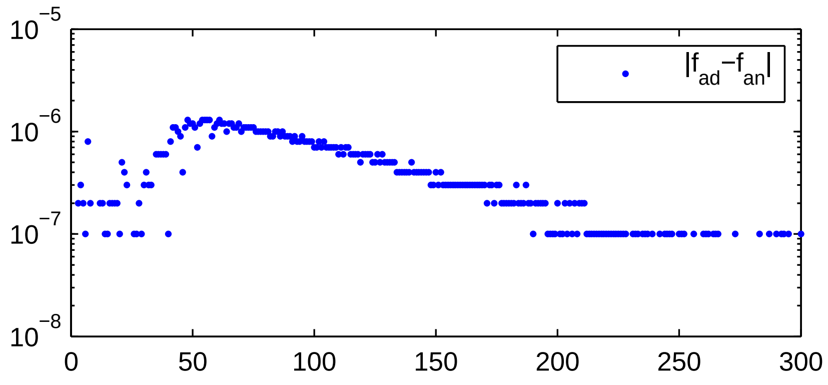


Figure 3: Discrepancy of optimization runs using analytic gradients and AD gradients.

## 5.3 Comparison of gradients

In accordance with (2.13) and (2.14), the main challenge to compute the target function as well as the analytical gradient is the provision of the normal derivative $\partial u/\partial \mathbf{n}$. It is given in its analytic form in (3.5) as

$$\mathcal{V}\frac{\partial v}{\partial \mathbf{n}} = \left(\frac{1}{2} - \mathcal{K}\right) N_j.$$

As described in Subsection 3.3, the discretization of the analytical gradient yields a system of equations given by

$$\mathbf{V}_L \mathbf{v}_L = \left(\frac{1}{2}\mathbf{M}_L - \mathbf{K}_L\right) \mathbf{G}_L^{-1} \mathbf{u}_L, \tag{5.1}$$

where $L$ refers to the discretization level, corresponding to $N = 2^L$ boundary elements. The boundary element matrices $\mathbf{V}_L$, $\mathbf{K}_L$, the mass matrix $\mathbf{G}_L$ with respect to piecewise linear ansatz and test functions and the mass matrix $\mathbf{M}_L$ with respect to piecewise linear ansatz and piecewise constant test functions, as well as the right-hand side $\mathbf{u}_L$ depend on the actual shape. The representation (5.1) can be used for approximating *both*, the objective (2.13) and the analytical shape gradient (2.14), during the optimization.

In case of AD, stating explicitly the dependency on the shape for the discretized version, one obtains as solution

$$\mathbf{v}_L(A_N)=\mathbf{V}_L^{-1}(A_N)\Big(\frac{1}{2}\mathbf{M}_L(A_N)+\mathbf{K}_L(A_N)\Big)\mathbf{G}_L^{-1}(A_N)\mathbf{u}_L(A_N).$$

Thus, the application of algorithmic differentiation, i.e., the chain rule, yields the derivative computation

$$\begin{aligned}\frac{\partial\mathbf{v}_L(A_N)}{\partial A_N}=&\Big(\frac{\partial\mathbf{V}_L^{-1}(A_N)}{\partial A_N}\Big(\frac{1}{2}\mathbf{M}_L(A_N)-\mathbf{K}_L(A_N)\Big)\mathbf{G}_L^{-1}(A_N)\mathbf{u}_L(A_N)\Big)\\&\cdot\Big(\Big(\frac{1}{2}\frac{\partial\mathbf{M}_L(A_N)}{\partial A_N}-\frac{\partial\mathbf{K}_L(A_N)}{\partial A_N}\Big)\mathbf{G}_L^{-1}(A_N)\mathbf{u}_L(A_N)\Big)\\&\cdot\Big(\frac{\partial\mathbf{G}_L^{-1}(A_N)}{\partial A_N}\mathbf{u}_L(A_N)\Big)\cdot\frac{\partial\mathbf{u}_L(A_N)}{\partial A_N}.\end{aligned}$$

One obtains therefore a quite complicated representation of the derivative of the discretized variables $\mathbf{v}_L$ with respect to the design variables $A_N$. Despite this rather sophisticated representation of the derivative information, one finds for our application that the analytic gradient and the discrete gradient, provided by algorithmic differentiation, coincide up to discretization error accuracy. Figure 4 displays the convergence history of the gradient evaluated by AD (left graph) and its discrepancy to the analytic gradient for each outer iteration (right graph).

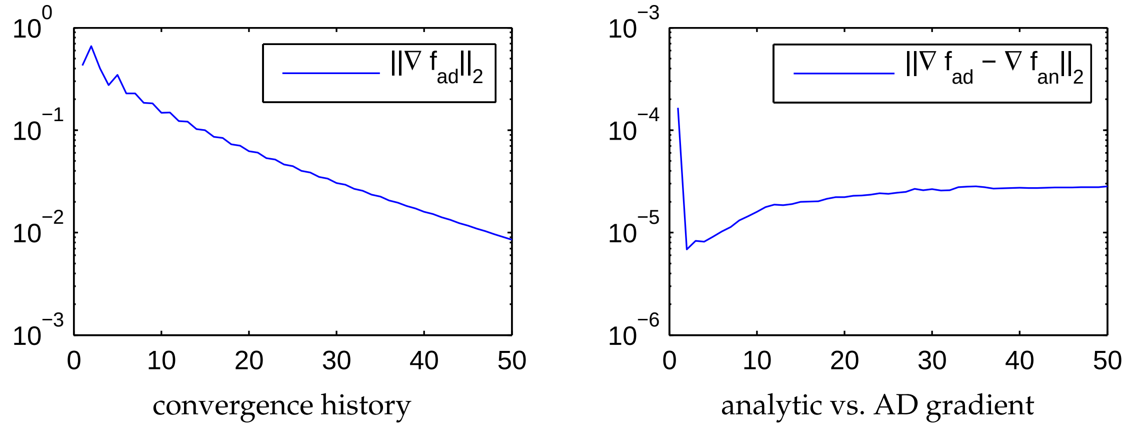


Figure 4: Convergence history and comparison of gradients.

This result also confirms the linear convergence rate of the gradient based optimization method. The evaluation time of the analytical gradient requires the same key ingredient like the computation of the function value. Hence, the computing time of 0.3 seconds for one analytic gradient corresponds to the time required to evaluate the function value itself. The computation of one gradient using ADOL-C requires 5.3 seconds. This yields a ratio of 18 of the computing time for one AD gradient and the computing time of the underlying function. Hence, the gradient computation by ADOL-C is roughly four times slower than the theory outlined in Section 4 predicts.

## 6 Concluding remarks

Due to well-posedness and the considered accuracy level, one achieves a good agreement of the analytically computed shape gradient projected to the finite dimensional ansatz space and the AD gradient. We observed that the difference increase on coarser discretization levels, where the AD gradient provides the 'correct' gradient for the fully discretized problem. This reflects the fact that the discretization error of the analytical shape gradient is larger on coarser levels.

The analytical gradient is computed 'almost for free' in our particular situation. This might be one reason that the ratio between analytical gradient computation and AD gradient computation is a bit worser than predicted by the theory. The bottleneck for the AD-application is the generation of the two stiffness matrices $\mathbf{V}_L$ and $\mathbf{K}_L$ in (5.1), more precisely, the related matrix compression of the wavelet-based boundary element method. The efficient exploitation of the particular structure of the compression pattern will be thus a subject of future work.

**References**

[1] F. Abraham, M. Behr and M. Heinkenschloss. The effect of stabilization in finite element methods for the optimal boundary control of the Oseen equations. *Finite Elem. Anal. Des.*, 41:229–251, 2004.

[2] CppAD: A package for differentiation of C++ algorithms. http://www.coin-or.org/CppAD/.

[3] O. Colaud and A. Henrot. Numerical approximation of a free boundary problem arising in electromagnetic shaping. *SIAM J. Numer. Anal.*, 31:1109–1127, 1994.

[4] M. Crouzeix. Variational approach of magnetic shaping problem. *Eur. J. Mech., B/Fluids*, 10:527–536, 1991.

[5] M. Crouzeix and F.J. Sayas. Asymptotic expansions of the error of spline Galerkin boundary element methods. *Numer. Math.*, 78:523–547, 1998.

[6] M. Crouzeix, Ph. Feat and F.J. Sayas. Theorical and numerical study of a free boundary problem by boundary integral methods. *ESAIM Math. Model. Numer. Anal.*, 35(6):1137–1158, 2001.

[7] W. Dahmen. Wavelet and multiscale methods for operator equations. *Acta Numerica*, 6:55–228, 1997.

[8] W. Dahmen, H. Harbrecht and R. Schneider. Compression techniques for boundary integral equations. Optimal complexity estimates. *SIAM J. Numer. Anal.*, 43(6):2251–2271, 2006.

[9] W. Dahmen and A. Kunoth. Multilevel preconditioning. *Numer. Math.*, 63:315–344, 1992.

[10] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Nonlinear Equations and Unconstrained Optimization Techniques*. Prentice-Hall, Englewood Cliffs, 1983.

[11] J. Descloux. Stability of solutions of the bidimensional magnetic shaping problem in absense of surface tension. *Eur. J. Mech., B/Fluids*, 10:513–526, 1991.

[12] K. Eppler. Boundary integral representations of second derivatives in shape optimization. *Discuss. Math. Differ. Incl. Control Optim.*, 20:63–78, 2000.

[13] K. Eppler. Optimal shape design for elliptic equations via BIE-methods. *J. Appl. Math. Comput. Sci.*, 10:487–516, 2000.

[14] K. Eppler. Second derivatives and sufficient optimality conditions for shape functionals. *Control Cybernet.*, 29:485–512, 2000.

[15] K. Eppler and H. Harbrecht. Numerical solution of elliptic shape optimization problems using wavelet-based BEM. *Optim. Methods Softw.*, 18:105–123, 2003.

[16] K. Eppler and H. Harbrecht. Second order shape optimization using wavelet BEM. *Optim. Methods Softw.*, 21:135–153, 2006.

[17] K. Eppler and H. Harbrecht. Exterior electromagnetic shaping using wavelet BEM. *Math. Meth. Appl. Sci.*, 28:387–405, 2005.

[18] K. Eppler, H. Harbrecht and R. Schneider. On convergence in elliptic shape optimization. *SIAM J. Control Optim.*, 45:61–83, 2007.

[19] Y.G. Evtushenko. Computation of exact gradients in distributed dynamic systems. *Optim. Methods Softw.*, 9(1–3):45–75, 1998.

[20] R. Giering and T. Kaminski. Recipes for adjoint code construction. *ACM Trans. Math. Softw.*, 24:437–474, 1998.

[21] R. Griesse and A. Walther. Evaluating gradients in optimal control — continuous adjoints versus automatic differentiation. *J. Optim. Theory Appl.*, 122(1):63–86, 2004.

[22] A. Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pp. 83–108. Kluwer Academic Publishers, Amsterdam, 1989.

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

[24] A. Griewank, K. Kulshreshtha, and A. Walther, On the numerical stability of algorithmic differentiation, *Computing*, 94(2–4):125–149, 2012.

[25] Ch. Grossmann and J. Terno. *Numerik der Optimierung*. Teubner, Stuttgart, 1993.

[26] H. Harbrecht and R. Schneider. Wavelet Galerkin schemes for 2D-BEM. In J. Elschner *et al.*, editors, *Problems and methods in mathematical physics (Chemnitz, 1999)*, vol. 121 of *Operator Theory: Advances and Applications*, Birkhäuser, Basel (2001).

[27] L. Hascoët and V. Pascual. The Tapenade automatic differentiation tool. Principles, model, and specification. *ACM Trans. Math. Softw.*, 39(3):20, 2013.

[28] J. Haslinger and P. Neitaanmäki. *Finite Element Approximation for Optimal Shape, Material and Topological Design*. Wiley, Chichester, 2nd edition, 1996.

[29] A. Henrot and M. Pierre. About existence of equilibria in electromagnetic casting. *Q. Appl. Math.*, 49(3):563–575, 1991.

[30] A. Henrot, M. Pierre and M. Rihani. Finite dimensional reduction for the positivity of some second shape derivatives. *Methods Appl. Anal.*, 10:457–476, 2003.

[31] M. Hinze and T. Slawig. Adjoint gradients compared to gradients from algorithmic dif-

ferentiation in instantaneous control of the Navier–Stokes equations. *Optim. Methods Softw.* 18(3):299–315, 2003.

[32] G.C. Hsiao and W. Wendland. A finite element method for some integral equations of the first kind. *J. Math. Anal. Appl.*, 58:449–481, 1977.

[33] G.C. Hsiao and W. Wendland. On a boundary integral method for some exterior problems in elasticity. *Mat. Mekh. Astron.*, 18:31–60, 1985.

[34] A. Novruzi. Contribution en optimisation des formes et applications. PHD Thesis, Nancy, 1997.

[35] A. Novruzi and J.R. Roche. Second derivatives, Newton method, application to shape optimization. *INRIA-report No. 2555*, 1995.

[36] M. Pierre and J.-R. Roche. Computation of free surfaces in the electromagnetic shaping of liquid metals by optimization algorithms. *Eur. J. Mech., B/Fluids*, 10:489–500, 1991.

[37] S. Sauter and C. Schwab. *Boundary Element Methods. Springer Series in Computational Mathematics*. Springer, Berlin-Heidelberg, 2010.

[38] R. Schneider. *Multiskalen- und Wavelet-Matrixkompression: Analysisbasierte Methoden zur Lösung großer vollbesetzter Gleichungssyteme*. B.G. Teubner, Stuttgart, 1998.

[39] J. Sokolowski and J.-P. Zolesio. *Introduction to Shape Optimization*. Springer, Berlin, 1992.

[40] O. Steinbach. *Numerical Approximation Methods for Elliptic Boundary Value Problems*, Springer Science & Business, New York, 2008.

[41] A. Walther. Automatic differentiation of explicit Runge-Kutta methods for optimal control. *Comput. Optim. Appl.*, 36:83–108, 2007.

[42] A. Walther and A. Griewank. Getting started with ADOL-C, in *Combinatorial Scientific Computing*, Chapman-Hall CRC Computational Science, pp. 181–202, 2012.