Geometric Inverse Problems, Images and SQP Methods Based on Weak Shape Hessians

Kumulative Habilitation

als Teilleistung zur Erlangung der Lehrbefähigung in Mathematik

Der Fakultät für Mathematik und Informatik der Universität Würzburg vorgelegt von

Dr. rer. nat. Stephan Schmidt
1 Introduction to Shape Optimization

1.1 Geometric Inverse Problems

A geometric inverse problem represents the task of reconstructing a physical object out of measurement data. The unknown geometry is linked to the data via a physical model governed by partial differential equations (PDEs). Typically, inverse problems arise in tomography and medical imaging. Applications in engineering include the identification of impurities in technical components via non-destructive testing or obstacle problems. Finally, elements of remote sensing, e.g. in earth observation or astrophysics [23], also fall into this category.

Mathematically, the underlying model is most often an elliptic PDE, see for instance [9, 10]. However, as part of this work, we are mostly interested in inverse problems governed by hyperbolic conservation laws, meaning we seek numerical algorithms to compute shapes that interact with waves in certain desired ways. To this end let \( \Omega \) be the unknown domain to be found and let \( \mathcal{D} \) be the hold-all domain, a domain to contain all of the iterates of the shape optimization problem, meaning \( \Omega \subset \mathcal{D} \subset \mathbb{R}^d \). Furthermore, \( \Omega \) is assumed to have piecewise smooth boundaries \( \Gamma := \partial \Omega \). Finally, \( \Gamma_{\text{inc}} \subset \Gamma \) denotes the unknown inclusion to be found and \( \Gamma_{i/o} \subset \Gamma \) denotes the boundary where measurements can be taken. Summarizing the above, one arrives at the following problem.

Minimizing

\[
\min_{(\varphi, \Gamma_{\text{inc}})} J(\varphi, \Omega) := \frac{1}{2} \int_0^T \int_{\Gamma_{i/o}} \| B(\varphi - \varphi_{\text{meas}}) \|^2 \, d\mathbf{s} \, dt + \delta R(\Gamma_{\text{inc}}) \tag{1}
\]

subject to

\[
\begin{align*}
\varphi + \text{div} \, F(\varphi) &= 0 \quad \text{in} \quad \Omega \\
F_b(\varphi, n) &= g \quad \text{on} \quad \Gamma_{i/o} \\
F_b(\varphi, n) &= 0 \quad \text{on} \quad \partial \Omega \setminus \Gamma_{i/o}.
\end{align*}
\]  

Here, equation (2) describes the wave propagation of some physical state \( \varphi \) governed by the flux \( F \) during time \([0, T]\). Information enters the domain on \( \Gamma_{i/o} \) via the scanning pulse \( g \) and the boundary flux operator \( F_b(\varphi, n) \), where \( n \) is the outward normal. The objective \( J \) tracks the difference between the measured state \( \varphi_{\text{meas}} \) and the actual state \( \varphi \). Also, \( B \) is a matrix describing the observable parts of \( \varphi \). Furthermore, the objective also contains a regularization term \( R \), often the perimeter of \( \Gamma_{\text{inc}} \) for \( \delta > 0 \). Frequently in inverse problems, measurements are taken everywhere in the domain or, in so-called tomography problems, behind the unknown inclusion. Here, however, measurements are always taken at the inlet boundary, meaning one is essentially controlling the shape \( \Gamma_{\text{inc}} \) such that the echo of the incoming wave \( g \), resp. \( \varphi \), behaves in a desired way. The above problem has been studied in [SS] with respect to computational acoustics and acoustic horn optimization and in [S4] with respect to remote sensing governed by Maxwell’s equations. In all cases, there is a canonical choice for \( B \), namely the upwind splitting operator, which maps the state \( \varphi \) to the outgoing characteristics. This will be discussed alongside the adjoint equations.

Initially, the approach of “optimize-then-discretized” is used, while the reverse is mentioned in Section 3. Classical shape calculus from [15, 40] is used in [S4, S5] to formally shape differentiate the Lagrangian of (1), (2). To this end, a vector field \( V \in C^0_0(\mathcal{D}, \mathbb{R}^d) \) is used to generate perturbed domains

\[
\Omega_\epsilon[V] := \{ x + \epsilon V(x) : x \in \Omega \}
\]

and we seek the one sided limit

\[
dJ(\Omega)[V] := \lim_{\epsilon \to 0^+} \frac{J(\varphi_{\epsilon}, \Omega_\epsilon[V]) - J(\varphi, \Omega)}{\epsilon}. \tag{3}
\]

In computing this limit, two concepts of the variation of the state variable are usually introduced, the material derivative \( d\varphi[V] \) and the local or shape derivative \( \varphi'[V] \), which are defined via

\[
d\varphi[V] := \lim_{\epsilon \to 0^+} \frac{\varphi_{\epsilon}(x) - \varphi(x)}{\epsilon}, \quad \varphi'[V] := d\varphi[V] - D\varphi V. \tag{4}
\]
Assuming \( \varphi \) to be in some finite element space \( P \), adjoint calculus applied to the Lagrangian in weak form leads to the following adjoint problem. Find \( \lambda \in P \) such that

\[
0 = \int_0^T \int_\Omega \left( -\dot{\lambda}, \xi - \langle \xi, D^T F(\varphi) \nabla \lambda \rangle \right) \, dx \, dt + \int_0^T \int_\Gamma \langle \xi, D^T F_b(\varphi, n) \cdot \lambda \rangle \, ds \, dt
\]

\[
+ \int_0^T \int_{\Gamma_{in}} \langle B(\varphi - \varphi_{\text{meas}}), B \xi \rangle \, ds \, dt \quad \forall \xi \in P.
\]

(5)

An important observation from the adjoint equation is a canonical choice for the measurement operator \( B \). Suppose a linear conservation law, where \( F_b(\varphi, n) \) denotes the incoming characteristic. Undoing the adjoint time reversal, the strong form interpretation of the \( \Gamma_{in} \) boundary flux in [5] is

\[
F_b^T(\lambda, n) = B^T B \cdot (\varphi - \varphi_{\text{meas}}) \text{ on } \Gamma_{in}.
\]

Supposing \( F_b \) describes the characteristic entering the primal system, a sensible choice for \( B^T B \) is the incoming characteristic for \( -\lambda \), i.e., the outgoing characteristic for \( \varphi \).

### 1.2 Loss of Regularity

It is worth noting that the boundary representation of the shape derivative as in [7] usually has higher regularity requirements than provided by the primal and dual problem in the continuous setting, leading to poor or mesh dependent convergence behavior. See for example [15, Chap. 10, Sec. 25], where the boundary formulation requires \( H^2(\Omega) \cap H_0^1(\Omega) \) continuity for an elliptic problem otherwise in \( H_0^1(\Omega) \). In applications, this usually necessitates some form of gradient smoothing. Assuming sufficient smoothness, the limit [6] for a volume objective can be computed via one of the following formulations

\[
dJ(\Omega)[V] = \int_\Omega f \text{ div } V + df[V] \, dx = \int_\Omega \text{ div}(fV) + f'[V] \, dx = \int_{\partial \Omega} \langle V, n \rangle f \, ds + \int_\Omega f'[V] \, dx,
\]

(6)

where the latter part summarizes how the boundary or strong form representation can be found by transforming the material derivative into the local derivative. As discussed in Section 3 in more detail, all above formulations are not equally suited for a finite element discretization and in [S2], a similar relationship to (6) is studied for objectives defined on the surface. Thus, the terms weak and strong form are preferable over “boundary” or “surface” formulation. It is worth mentioning that for surface functionals, a rather inconvenient computation of the mesh curvature is also avoided by using the weak form.

As seen above, the existence of the local derivative \( f'[V] \) is essential for finding the strong form representation of the shape derivative and there are analytical results showing their existence and uniqueness for a variety of related mixed hyperbolic problems, which motivates their use in the remainder of this section. See for instance [27, 28, 29, 30] for a discussion on the respective regularity requirements and strong form shape derivatives concerning the compressible Navier–Stokes equations or [11, 12, 25] with respect to Maxwell’s equations. Within this context, the strong form representation of the shape derivative for problem [1] is given by

\[
dJ(\Omega)[V] = \int_{t_b}^{t_f} \int_{\Gamma_{inc}} \langle V, n \rangle \text{ div}(\lambda \cdot F_b(\varphi)) \, ds \, dt,
\]

(7)

provided the Riemann flux is linear in \( n \), i.e., \( F_b(\varphi, n) = \tilde{F}_b(\varphi) \cdot n \). More details can be found in [S4].

For a rigorous treatment of the weak approach, including problems governed by the incompressible Navier–Stokes equations, see [8, 41]. Also, a derivation of the adjoint equation for the compressible Navier–Stokes equations in weak form can be found in [S6] and numerical convergence studies of the strong form with respect to changing the polynomial order of the ansatz functions or mesh refinement can be found in [S5, S6].
1.2.1 Horn Optimization and Computational Acoustics

Objective (1) is very well suited for maximizing transmission power of antennas. By setting $\varphi_{\text{meas}} = 0$, one is interested in shapes $\Gamma_{\text{inc}}$ that do not reflect any power back to the inlet boundary, i.e., the antenna $\Gamma_{i/o}$. Interpreted differently, one could also speak of a stealth problem, that is to cloak the object $\Gamma_{\text{inc}}$ to be invisible for a sensor at $\Gamma_{i/o}$ when scanned by a signal $g$. In [S5], an upwind stabilized discontinuous Galerkin solver based on FEniCS [24] is developed to solve (1) within the context of computational acoustics and the design of horn loudspeakers, which form so-called line arrays [45] in most mid- and high-frequency range public address systems. An exemplified geometry of one such loudspeaker is shown in Figure 1.

A number of preceding studies use scientific computing to design acoustic horns [4, 5, 6, 17, 20, 42, 44]. Contrarily, the emphasis here is on detailed design of the acoustic horn in full 3D using the time-domain model (1). The advantage of this approach is the ability to optimize the horn for multiple working conditions simultaneously by choosing the forcing $g$ to contain all frequencies of interest. It is worth noting that under considerable simplifications, the 1D Webster horn equation [26] of wave motion shows exponential shapes to be optimal if there is no constraint on length.

Acoustic wave propagation is modeled via the linear wave equation. To this end, let $p$ denote the acoustic pressure, $u$ the acoustic momentum density vector, that is, the product of the static air density and the acoustic velocity vector, and $c$ the speed of sound. A study of the characteristics of this problem leads to the objective

$$ J(u, p, \Omega) = \frac{1}{2} \int_0^T \int_{\Gamma_{i/o}} (p + \langle u, n \rangle)^2 \, ds \, dt, \quad (8) $$

where state and adjoint are given by

$$ \frac{\partial u}{\partial t} + \nabla p = 0, \quad \frac{\partial \lambda_u}{\partial t} + \nabla \lambda_p = 0 \quad \text{in } \Omega, $$

$$ \frac{\partial p}{\partial t} + c^2 \text{div} \, u = 0, \quad -\frac{\partial \lambda_p}{\partial t} + c^2 \text{div} \, \lambda_u = 0 \quad \text{in } \Omega, $$

$$ \frac{1}{2} (p - c \langle u, n \rangle) = g, \quad \frac{1}{2} (\lambda_p - c \langle \lambda_u, n \rangle) = \frac{1}{2} (p + c \langle u, n \rangle) \quad \text{on } \Gamma_{i/o}, $$

$$ \frac{1}{2} (p - c \langle u, n \rangle) = 0, \quad \frac{1}{2} (\lambda_p - c \langle \lambda_u, n \rangle) = 0 \quad \text{on } \Gamma_{\text{out}}, $$

$$ \langle u, n \rangle = 0, \quad \langle \lambda_u, n \rangle = 0 \quad \text{on } \Gamma_{\text{wall}}. $$

Additional geometric constraints on length and mouth of the horn are omitted for brevity.

It is worth noting that this problem features two equivalent formulations of the strong form shape derivative, namely

$$ dJ(u, p, \Omega)[V] = 2c \int_0^T \int_{\Gamma_{\text{inc}}} \langle \langle V, n \rangle \rangle \text{div}(\lambda_p u) \, ds \, dt = 2c \int_0^T \int_{\Gamma_{\text{inc}}} \langle \langle V, n \rangle \rangle \text{div}(p \lambda_u) \, ds \, dt. \quad (10) $$
The horn was discretized using 114,417 discontinuous Galerkin elements. Test- and trial-functions are linear polynomials on each tetrahedron, which yields a total of $(3+1) \cdot 4 \cdot 114,417 = 1,830,672$ unknowns per time step. The end time is set to $T = 18$ ms and the time step $dt = 5.0 \cdot 10^{-7}$. Time derivatives are discretized by the trapezoidal method and solved in parallel via ILU preconditioned GMRES. The initial and optimized horn are shown in Figure 2. The performance of a horn loudspeaker can be measured by the reflection spectrum, which indicates the fraction with which a single frequency in the input pulse $g$ is returned to the sender. The optimized horn was found very much superior for a wide range of frequencies, as shown in Figure 3.

![Figure 2: Three-view of the initial (left) and optimized (right) horn geometry](image1)

![Figure 3: Left: Magnitude reflection spectra of the initial and optimized horn. Right: Effective radii.](image2)

### 1.2.2 Shape Reconstruction Using Maxwell’s Equations

A proper shape reconstruction problem governed by Maxwell’s equations with non-zero $\varphi_{\text{meas}}$ is considered in [S4]. The state equation in strong form is given by

$$\mu \frac{\partial H}{\partial t} = -\nabla \times E, \quad \varepsilon \frac{\partial E}{\partial t} = \nabla \times H - \sigma E,$$

where $E(t, \cdot) \in (L^2(\Omega))^3$ is the electric flux strength and $H(t, \cdot) \in (L^2(\Omega))^3$ is the magnetizing field. Furthermore, $\varepsilon$ and $\mu$ are the constant permittivity and permeability and $\sigma$ describes the conductivity of the medium.

To find the objective and appropriate fluxes, a characteristic or upwind decomposition is needed. Computing the eigenvalues of the Maxwell system leads to the characteristic flux for the Riemann problem

$$F_0(H, E, n) = \frac{1}{2} \left[ \begin{array}{cc} \hat{A}_n c & A_n Y c \\ -A_n Z c & \hat{A}_n c \end{array} \right] \cdot \left( \begin{array}{c} H' \\ E' \end{array} \right) + \left[ \begin{array}{cc} -\hat{A}_n c & A_n Y c \\ -A_n Z c & -\hat{A}_n c \end{array} \right] \cdot \left( \begin{array}{c} H' \\ E' \end{array} \right),$$

(11)
where $E^c$ and $E^r$ are the respective local and remote states and $\tilde{A}_n$ is given by

$$
\tilde{A}_n := \begin{bmatrix}
    r_y^2 + r_z^2 & -n_x n_y & -n_x n_z \\
    -n_x n_y & r_x^2 + r_z^2 & -n_y n_z \\
    -n_x n_z & -n_y n_z & r_x^2 + r_y^2
\end{bmatrix}.
$$

Finally, $Y$ and $Z$ are constants related to the wave propagation speed. The shape derivative here is given by

$$
dJ(\Omega)[V] = \int_{t_0}^{t_f} \int_{\Gamma_{inc}} \langle V, n \rangle \text{div}(D_n^T (\lambda \cdot \tilde{F}_b(H, E) \cdot n)) \ ds \ dt = \int_{t_0}^{t_f} \int_{\Gamma_{inc}} \langle V, n \rangle Zc \text{div}(\lambda E \times H) \ ds \ dt, \quad (12)
$$

which is structurally very similar to (10).

A discontinuous Galerkin solver using upwind stabilization stemming from the flux splitting is used to solve an actual shape reconstruction problem as shown in Figure 4. The scanning pulse $g$ is a SINC pulse given by

$$
g(t) = \sin \left( \frac{2\pi f_c(t - t_c)}{2\pi f_s(t - t_c)} \right) \sin \left( 2\pi f_c(t - t_c) \right) w(t, t_c),
$$

which is truncated via the Hamming window function

$$
w(t, t_c) = \begin{cases}
0.54 + 0.46 \cos \left( \frac{\pi(t-t_c)}{t_c} \right), & \text{for } t \in [0, 2t_c], \\
0, & \text{otherwise}.
\end{cases}
$$

The center frequency is $f_c = 8.2 \text{ GHz}$ and the spread is $f_s = 4.1 \text{ GHz}$, meaning we scan using frequencies ranging from 4.1 GHz to 12.3 GHz, i.e., using wavelengths between 2.4 cm and 7.3 cm. The obstacle is given by the imprint function

$$
z_{new} = z + \frac{-W}{2 + 2 \exp \left( 100(\sqrt{x^2 + y^2} - W) \right)},
$$

where $W = \frac{299792458 \text{ m s}^{-1}}{4.1 \text{ GHz}} \approx 7.31 \text{ cm}$. Thus, the imprint has a depth of roughly 3.65 cm, the same order of magnitude as the wavelength of the scanning pulse, which adds to the difficulty of the problem. The reconstructed shape is also shown in Figure 4. The algorithm is based on a smoothed gradient descent scheme, i.e., an approximative Newton scheme, where a Laplace-type smoothing

![Figure 4: Left: Geometry to be reconstructed and polarization of the scanning pulse. Right: Reconstruction started from a cuboid after 373 iterations.](image)
of the shape derivative is used to approximate the Hessian. To properly resolve the scanning pulse, a mesh is used such that there are a total of 862,680 unknowns for the state \((H, E)\) per time-step or 472,748,640 unknowns for one primal simulation trajectory in total. To compute the adjoint forcing subject to time reversal, the primal state is only stored on the boundary \(\Gamma_{i/o}\), which enables computing the gradient in a “store everything on boundary” fashion without checkpointing.

### 1.3 Large Scale Problems and CFD

Besides geometric inverse problems, another natural area of shape optimization is fluid dynamics, in particular finding shapes of minimal drag in compressible aerodynamics. This can be exemplified by the problem

\[
\min_{(U, \Omega)} J(U, \Omega) := \int_{\Gamma_w} (\rho n - \tau n) \cdot \alpha \, d s \\
\text{subject to} \\
0 = -\langle F^c(U) - F^v(U), \nabla \xi \rangle_{\Omega} + \langle n \cdot (F^c(U) - F^v(U)), \xi \rangle_{\Gamma}
\]

for all \(\xi\) in some suitable discretization space. Additional geometric constraints are usually necessary to prevent degenerate shapes. Here, \(U = (\rho, \rho u_1, \rho u_2, \rho u_3, \rho E)^T\) denotes the conserved variables containing density \(\rho\), velocity \(u\) and energy \(E\). Also, \(F^c(U)\) and \(F^v(U)\) denote the respective convective and viscous fluxes for the compressible Euler or Navier–Stokes equations. Pressure \(p\) is given by the perfect gas law, \(p = (\gamma - 1)\rho(E - \frac{1}{2}(u_1^2 + u_2^2 + u_3^2))\), where \(\gamma\) is the adiabatic exponent. Finally, \(\alpha\) is the angle of attack and \(\Gamma_w\) the surface of the aircraft.

Shape derivatives for the above problem are desirable for large-scale industrial problems because they offer one possible alternative to skip the otherwise costly to compute sensitivities of the mesh deformation step or the CAD-parameterization software. For this reason, they have been studied and implemented in the DLR flow solver TAU \[39\] in \[34, 37, 35\] and applied to optimizing a complete blended wing-body aircraft in \[S3\]. While TAU is a finite volume solver, discontinuous Galerkin solvers are typically constructed with great flexibility in regards to the polynomial order \(r\) of their ansatz functions. To study the adjoint in variational form, and also to study the convergence behavior of the loss of regularity problem described in Section \[1.2\] with respect to polynomial degree, the shape derivative of the above problem has been implemented in the DLR solver PADGE \[18\] in \[S6\]. A very good agreement in comparison to finite differences could be observed for \(r \geq 4\).

![Figure 5: Blended wing-body aircraft. Wind-tunnel and CFD model with initial and optimized shapes. Wind tunnel picture courtesy DLR Germany.](image-url)
2 Images on Surfaces, Regularization and Non-Smooth Optimization

2.1 Regularization, Total Variation and Non-Smoothness

So far, there has been limited consideration of the regularization term $R$ in (1). In [S4] the surface area of the inclusion was used, which can be motivated from applications, where inclusions stem from impurities or bubbles trapped during solidification. The shape derivative of this surface area leads to so-called curvature flow problems [16] often used in mesh smoothing and iso-perimeter problems, which are also studied in [S2] in more detail. However, due to polarization in electromagnetic waves, geometries with sharp corners and right angles can have profound effects on the reflection properties. Likewise, kinks and layers are also common in geological applications. This leads to the desire to consider regularization terms that foster the creation of non-smooth geometries.

For instance, [46] motivates the use of the total variation semi-norm of geometric quantities such as the tangential gradient of the normal, while TV regularization of flat images consisting of pixels originates from [13, 32]. Thus, we study non-smooth optimization and total variation denoising on surfaces in [S1]. In particular, the problem

$$\text{Minimize} \quad \frac{1}{2} \int_S |Bu - f|^2 \, ds + \frac{\alpha}{2} \int_S |u|^2 \, ds + \beta \int_S |\nabla u|$$

(13)

is considered, where $S \subset \mathbb{R}^3$ is a smooth, compact, orientable and connected surface without boundary and $u \in BV(S)$ denotes the space of functions of bounded variation on the surface $S$, and $\int_S |\nabla u|$ is the surface analog of the total variation (TV) semi-norm. Assuming $f \in L^2(S)$ to be observed data, e.g., a blurred image on the surface $S$, and $B \in L^2(I(S))$ to be the observation operator, this problem can be seen as the “image on surface” analog to (1), where $\alpha \geq 0$ scales a smooth regularization term and $\beta \geq 0$ scales the TV term.

To deal with the non-smoothness of the total variation term, we transport the pre-dual approach of [19, 22] in [S1] from the flat situation onto surfaces using rotationally invariant TV semi-norm formulations. In particular, the Fenchel dual of

$$\text{Minimize} \quad \frac{1}{2} ||\text{div} \, p + B^* f||_{(\alpha + B^* B)^{-1}}^2$$

subject to $|p|_2 \leq \beta$ a.e. on $S, \ p \in H(\text{div}; S)$

(14)

is equivalent to optimization problem (13). The proof [S1, Theorem 8] is based on showing there is no duality gap. Hence, the non-smoothness of the TV term is transformed into a point-wise almost everywhere 2-norm constraint on $p$. In particular, we establish a rigorous relation between the primal and dual problems in appropriate function spaces. We propose to use an interior point method to solve the constrained problem and establish the well-posedness of the barrier approximations and provide necessary and sufficient optimality conditions. Applicability is ensured by developing a surface finite element solver based on the FEniCS framework using $H(\text{div}; S)$ conforming Raviart-Thomas ($R^T$) elements on surfaces for $p$, see Figure 6. Using such a finite element space is quite unique within imaging as the pixelated nature of flat images normally fosters finite difference schemes. Our solver accepts texture and geometry inputs in the wavefront format, see Figure 7, thereby ensuring interoperability with common 3D scanning and printing devices.

2.2 Selected Numerical Results

The solution approach in [S1] is based on a logarithmic barrier method to deal with the inequality constraints. Consequently, we consider the following family of convex problems for a decreasing sequence of barrier parameters $\mu \searrow 0$:

$$\text{Minimize} \quad \frac{1}{2} ||\text{div} \, p + K^* f||_{B^{-1}}^2 - \mu \int_S \ln \left( \beta^2 - |p|_2^2 \right) \, ds$$

subject to $|p|_2 \leq \beta$ a.e. on $S, \ p \in H(\text{div}; S)$.

(15)
Figure 6: Left: global basis function $p$ from $\mathcal{RT} 1$ associated with the degree of freedom located on the edge $E$ adjacent to both triangles in $\mathbb{R}^2$. One clearly sees that $(p \cdot n)_E$ is constant, i.e., of degree $r = 0$, and continuous, i.e., $[p \cdot n]_E = 0$ holds. On all other edges we have $p \cdot n = 0$. Middle: the same situation on a spherical mesh of topological dimension 2 in $\mathbb{R}^3$. Right: detail.

Figure 7: Left: Texture bitmap as delivered by the scanner software. Right: Texture mapped onto the geometry, which is given as a wireframe consisting of triangles.

The analysis of interior point methods in $L^p$ spaces including a convergence analysis of the central path has been addressed in [31, 43] in the context of optimal control problems. Existence and uniqueness of solutions $p$ of (15) for $\mu > 0$ is shown in [S1]. Notice that the presence of the logarithmic barrier term helps to overcome the lack of strict convexity of the objective in (14).

One test case is denoising, i.e. $K = \text{id}$, of a scanned terracotta duck from Figure 7 but with the texture data converted to gray scale. This texture has sharp interfaces between body, beak and feet. The geometry consists of 354, 330 triangles and 177, 167 vertices. Furthermore, our algorithm was

Figure 8: Duck test case: noisy input and denoising for $\beta = 0.1$ and $\beta = 0.3$. The object was kindly scanned by the Rechenzentrum of Würzburg University.

found to be favorable to the established split Bregman algorithm for total variation denoising in terms
of wall-clock time to solution. The total variation denoising of the geometry is ongoing work.

3 Weak Form Shape Hessians and Symbolic Automatic Differentiation

3.1 Shape Derivatives and Finite Elements

The inherent non-linearity of shape optimization problems can result in slow convergence of gradient based approaches, which fosters the desire to construct higher order methods such as a shape-SQP scheme. Besides the ever-present gaps between the continuous and the discrete approach, there are three major difficulties with “optimize-then-discretize” shape optimization, which need to be overcome. First, a naive “repeated differentiation” approach, i.e.,

$$d^2J(\Omega)[V, W] := d(dJ(\Omega)[V])[W],$$

leads to non-symmetric expressions [14, 15]. The second difficulty is the lack of regularity of the classical boundary representation discussed in Section 1.2 which is typically prohibitive when it comes to numerically evaluating second order shape derivatives. Finally, most shape optimization problems are non-convex away from the optimal solution, which can for example be seen numerically [21] using operator symbol analysis [2, 3]. However, this last issue can be countered by globalization techniques.

Recent works [38] give new theoretical justifications to disregard the asymmetric part of the second directional shape derivative, which results in the same expressions as if the condition $(DV)^T W = 0$ is assumed for the two shape perturbation fields $V$ and $W$. This coincides with the “Lie bracket” condition in [15]. With respect to the lack of regularity, the material derivative $d_\varphi[V]$ in [4] is conform with finite element discretizations using standard elements, whereas the local shape derivative $\varphi'[V]$ is not [7]. This suggests establishing an adjoint approach, where local derivatives are entirely avoided and the adjoint is formally used to eliminate material derivatives. The downside of this process is the non-commutativity of material and spatial derivatives, for instance

$$d(Du)[V] = D(du)[V] = (DuDV)$$
$$d(div u)[V] = d(tr(Du))[V] = div(du[V]) - tr(DuDV).$$

For many applications, in particular compressible flows with turbulence modeling, this leads to a very labor intensive process prone to human error, rising the question if it is not possible to use a computer algebra system as an aid.

3.2 Domain Specific Languages and Automatic Differentiation

The idea of [S2] is to develop a computer algebra system, which can generate all forms of the shape derivative [6] automatically and create a shape Newton scheme with the symmetrized shape Hessian for a variety of problems, including the well-establish test-case of energy dissipation minimization governed by the incompressible Navier–Stokes equations. At first glance, such a software would essentially be a tool for symbolic automatic differentiation (AD). The recent advent of automatic simulation code generation based on domain specific programming languages such as UFL [1] means that a new area between purely symbolic manipulations and source code transformations has opened for AD tools and this area is explored with respect to shape optimization in [S2]. Due to the uniqueness of the UFL languages, symbolic manipulations, i.e., manipulations of the equation tree, correspond to source code changes. Thus, this shape-AD tool essentially behaves like a source code transformation tool for UFL on the outside, but internally actually conducts symbolic equation manipulations of variational forms.

The above computing paradigm managed to construct Newton schemes based on the weak form shape Hessian for a variety of problems, including those governed by the incompressible Navier–Stokes equations. In particular, the flow obstacle optimization problem from [9] was revisited and the
weak form shape derivative, excluding barycenter and volume constraint, was automatically found to be

\[
\frac{dJ}{du}(u, p, \Omega)[V] = \int_{\Omega} \text{div} V \left[ \eta \langle \nabla u, \nabla u \rangle + \eta \langle \nabla \lambda_u, \nabla u \rangle + \rho \langle \lambda_u, Du \cdot u \rangle - \rho \text{div} \lambda_u + \lambda_p \text{div} u \right] \, dx \\
+ \int_{\Omega} 2\eta \langle (DV)^T \nabla u, \nabla u \rangle - \eta \langle (DV)^T \nabla u, \nabla \lambda_u \rangle - \eta \langle \nabla u, (DV)^T \nabla \lambda_u \rangle \, dx \\
+ \int_{\Omega} -\rho \langle \lambda_u, DuDVu \rangle + \rho \text{tr}(D\lambda_uDV) - \lambda_p \text{tr}(DuDV) \, dx,
\]

where \((u, p)\) solves the incompressible Navier–Stokes equations with viscosity \(\eta\) and density \(\rho\). Furthermore, \((\lambda_u, \lambda_p)\) denotes the corresponding adjoint variables.

Subsequently, a symmetrized repeated differentiation was conducted to compute \(d^2J(\Omega)[V, W]\) and a Newton descent direction \(W\) was found by solving

\[
d^2J(\Omega)[V, W] + md(V, W) = -dJ(\Omega)[V]
\]

for all \(V\). To achieve nodal mesh movement, continuous Lagrange finite elements of first order were chosen for \(V\) and \(W\). The actual second order directional derivative \(d^2J(\Omega)[V, W]\) is too lengthy to be presented here. Finally, the term \(md(V, W) := \langle V, W \rangle_{L^2(\Omega)} + 0.1 \langle \nabla V, \nabla W \rangle_{L^2(\Omega)}\) stands for “mesh deformation” and also ensures positive definiteness of the Hessian away from the optimum.

![Figure 9: Initial and optimized obstacle in a Navier–Stokes flow, optimized using shape SQP.](image)

![Figure 10: \(L^2(\Omega)\)-Norm of the Newton direction \(W\) (left) and value of the objective (right). The initial increase of the norm of the update stems from reducing the dampening of the globalization strategy.](image)

Resulting shapes are shown in Figure 9. Figure 10 compares the convergence behavior favourably to approximate Newton Schemes previously developed in [33, 36].
Appendix A: References Contributing to the Habilitation

Peer Reviewed Journal Publications


Appendix B: General References


Abstract. [Lai, Chan (Computer Vision and Image Understanding, 2011)] introduced an analog of the total variation image reconstruction approach [Rudin, Osher, Fatemi (Physica D, 1992)] for images on smooth surfaces. The problem is defined in terms of quantities intrinsic to the surface and it is therefore independent of the parametrization. In this paper, a rigorous analytical framework is developed for this model and its Fenchel predual. It is shown that the predual of the total variation problem is a quadratic optimization problem for the predual vector field $\mathbf{p} \in H(\text{div}; S)$ with pointwise inequality constraints on the surface. As in the flat case, $\mathbf{p}$ serves as an edge detector. A function space interior point method is proposed for the predual problem, which is discretized by conforming Raviart-Thomas finite elements on a triangulation of the surface. Well-posedness of the barrier problems is established. Numerical examples including denoising and inpainting problems with both gray-scale and color images on scanned 3D geometries of considerable complexity are presented.

Key words. total bounded variation, Fenchel predual problem, interior-point methods, image reconstruction, image denoising, image inpainting, surfaces

AMS subject classifications. 94A08, 92C55, 68U10, 49M29, 65K05

1. Introduction. We consider the image reconstruction problem

\[
\begin{aligned}
\text{Minimize} & \quad \frac{1}{2} \int_S |Ku - f|^2 \, ds + \frac{\alpha}{2} \int_S |u|^2 \, ds + \beta \int_S |\nabla u| \\
\text{over} & \quad u \in BV(S)
\end{aligned}
\]

where $S \subset \mathbb{R}^3$ is a smooth, compact, orientable and connected surface without boundary. $BV(S)$ denotes the space of functions of bounded variation on the surface $S$, and $\int_S |\nabla u|$ is...
Furthermore, the observed data \(f \in L^2(S)\), parameters \(\beta > 0\), \(\alpha \geq 0\) and the observation operator \(K \in \mathcal{L}(L^2(S))\) are given. By \(K^* \in \mathcal{L}(L^2(S))\) we denote the Hilbert space adjoint of \(K\). It will be shown that \(BV(S) \hookrightarrow L^2(S)\) so that the integrals in (1) are well defined. We assume throughout that either \(\alpha > 0\) holds, or else that \(K\) is injective and has closed range, i.e., there exists a constant \(\gamma > 0\) such that \(\|Ku\|_{L^2(S)} \geq \gamma \|u\|_{L^2(S)}\) for all \(u \in L^2(S)\). This second case is equivalent to \(K^*K\) being a coercive operator in \(\mathcal{L}(L^2(S))\); see for instance [29, Chapter A.2].

The motivation to study (1) goes back to the seminal work in [51], where the total variation (TV) seminorm \(\int |Du|\) was proposed as a regularizing functional in image reconstruction. Due to the choice of norms in the fidelity and regularization terms, problem (1) is also termed a TV–\(L^2\) model. A large body of literature on this topic has emerged; see for instance [15, 16, 18, 53, 56] and the references therein. The operator \(K\) appearing in (1) expresses available a-priori knowledge about the relation between the image \(u\) to be reconstructed and the observed data \(f\). Common examples include \(K = \text{id}\) for classical image denoising [51], \(K = \text{masking}\) for inpainting problems [20, Chapter 6.5], \(K = \text{blur}\) for deblurring problems [13, 25], and \(K = \text{coarsen}\) for un-zooming problems [42]. While our analysis applies to general operators \(K\), the numerical examples included in this paper comprise only denoising and inpainting problems, for which \(K\) acts pointwise and thus leads to diagonal or block-diagonal matrix structures. We expect that non-local operators may considerably increase the numerical cost of the proposed approach unless suitable preconditioned iterative solvers are employed. This is however not subject of the present study.

The increasing interest in studying image processing problems on surfaces is due to its numerous applications, for instance, in computer vision [37], geophysics [36], and medical imaging [41]. This is accompanied by the ongoing development in 3D scanning, remote sensing and other data acquisition hardware. In the applications mentioned unavoidable sampling errors from the imaging equipment, or the need to compress large-scale images, e.g., for limited-bandwidth internet applications, are potential sources of noise, necessitating post processing. The predominant approach in surface image processing so far is based on extensions of the nonlinear, anisotropic diffusion method going back to [44]. In particular, we mention [3, 21, 22] for surface intrinsic concepts, and [8, 9, 43] for volume-based formulations. We also point out [7] who consider an extension of the Mumford–Shah image segmentation problem using the active contour method on surfaces, with a subsequent restoration phase on the segmented parts driven by linear isotropic diffusion.

As an alternative to diffusion driven image restoration our focus here is on problem (1), which was recently proposed in [39] as an analog of the TV–\(L^2\) reconstruction model for images defined on smooth surfaces. One of the algorithms considered for its solution was Chambolle’s projection method [15], based on the formal convex dual problem of (1), and the split Bregman iteration [31]. The intention of the present paper is to extend the work of [39] in several directions. We establish a rigorous relation between the primal and dual problems in appropriate function spaces. To be precise, we formulate the predual of (1), which is a quadratic convex problem (15) with pointwise bound constraints in \(H(\text{div}; S)\), the analog on surfaces.
of the space of vector-valued \( L^2 \) functions whose divergence is likewise square integrable. The distinction between dual and predual problems is necessary due to \( BV(S) \) being non-reflexive. A similar analysis has previously been pursued in [35] for the ‘flat’ case. Notice however that in [35] the TV-seminorm is defined in a way which is not rotationally invariant but has the advantage of leading to pointwise simple bounds \(-\beta 1 \leq p \leq \beta 1\) for the dual variable \( p \). This structure is particularly amenable to numerical solution via a primal-dual active set method. By contrast, we propose to use an interior-point method, which deals nicely with pointwise nonlinear constraints of the form \(|p|_2 \leq \beta\) arising in the coordinate free setting that naturally comes with surfaces. We establish the well-posedness of the barrier approximations for positive barrier parameter and provide necessary and sufficient optimality conditions in function space. Another distinction from previous work is that we do not introduce additional regularization terms as in [35, Sect. 3], which would lift the predual problem to one in \( H^{1,2}(S; \mathbb{R}^2) \) but add artificial diffusion. Due to the well posedness of our approach in function space, we expect and numerically confirm its independence of the mesh size of the triangulation of the surface. By contrast, the split Bregman [31] and primal-dual methods [17] are naturally posed in a Hilbert space setting and thus they do not possess a function space analog when the primal space is the Banach space of \( BV \) functions. The last difference concerns the type of discretization employed. While Cartesian grids are natural in ‘flat’ image processing tasks and lend themselves to finite difference approximations, surfaces are naturally triangularized, for instance by 3D scanner software. Based on the rigorous formulation of the predual problem we are led to choose a conforming finite element discretization of the space \( H(\text{div}; S) \) by the surface analog of (possibly higher-order) Raviart–Thomas finite element spaces introduced in [47]. Notice that in [39] piecewise linear Lagrangian finite elements were considered, which are also \( H(\text{div}; S) \) conforming but do not exhaust that space.

This paper is organized as follows. In section 2 we introduce the proper functional analytic framework for the discussion of (1) and its predual. In particular, we recall the definition of the spaces \( BV(S) \) and \( H(\text{div}; S) \) on a smooth surface \( S \). Section 3 is devoted to the study of the Fenchel predual problem. In section 4 we formulate a function space interior point approach for the solution of the predual problem, analyze the well-posedness of the barrier approximations, and provide necessary and sufficient optimality conditions. Details concerning the discretization by Raviart–Thomas surface finite elements and the implementation of our method are also given in that section. Subsequently, numerical results are presented in section 5. While the presentation focuses on scalar (gray-scale) image data, an extension to multi-channel (color) images is rather straightforward and is presented, along with numerical results for denoising and inpainting problems, likewise in that section. We compare our approach with the split Bregman iteration in section 6. We end with conclusions and an outlook in section 7.

2. Functional Analytic Framework. In this section we introduce the necessary analytical framework to extend the definition of functions of bounded variation (BV) as well as functions in \( H(\text{div}; S) \) on an open subset of \( \mathbb{R}^n \) to functions defined on smooth surfaces.
2.1. Concepts from Differential Geometry. In a nutshell, a smooth surface $S$ is a two-dimensional manifold of class $C^\infty$ embedded in $\mathbb{R}^3$. In the interest of keeping the paper self-contained, we briefly summarize the required concepts from differential geometry. The interested reader is referred, for instance, to [24, 38, 45] for further background material.

Definition 1. A subset $S \subset \mathbb{R}^3$, endowed with the relative topology of $\mathbb{R}^3$, is a smooth surface if for every point $p \in S$ there exists an open set $V \subset S$ containing $p$, an open set $U \subset \mathbb{R}^2$ and a homeomorphism $x : U \to V$ with the additional properties that $x \in C^\infty(U; \mathbb{R}^3)$ holds and that the Jacobian of $x$ has rank 2 on $U$.

A mapping $x$ as above is called a parametrization at $p$. A collection of parametrizations covering all of $S$ is said to be an atlas of $S$. We will always associate with a smooth surface an atlas of parametrizations, and it will not matter throughout the paper which particular atlas is being used. Coordinates in the parameter domain $U$ of a chart $x$ will be denoted by $\zeta_j$, $j = 1, 2$.

A function $f : S \to \mathbb{R}$ is said to be of class $C^k$ if, for any parametrization $x$ in the atlas, the function $f \circ x : U \subset \mathbb{R}^2 \to \mathbb{R}$ is of class $C^k$. Similarly, this notion can be defined for functions $f$ defined only on an open subset of $S$ by appropriately restricting those parametrizations whose image intersects the domain of definition of $f$.

We continue with the notions of tangent vectors and the tangent space at a point $p \in S$. Consider a differentiable curve $\gamma : (\varepsilon, \varepsilon) \to S$ such that $\gamma(0) = p$. Then $X := \dot{\gamma}(0) \in \mathbb{R}^3$ is said to be the tangent vector to the curve $\gamma$ at $p$. The tangent space at $p$, denoted by $T_p(S)$, consists of all tangent vectors of such curves $\gamma$ at $p$. It is a vector space of dimension 2. If $p$ belongs to the image of some parametrization $x$, then it is easy to verify that $\left\{ \frac{\partial x}{\partial \zeta_1}(x^{-1}(p)), \frac{\partial x}{\partial \zeta_2}(x^{-1}(p)) \right\}$ constitutes a basis for $T_p(S)$. Therefore, any tangent vector $X$ at $p$ can be represented as $X = X^1 \frac{\partial x}{\partial \zeta_1}(x^{-1}(p))$. Here and in the sequel we use Einstein’s summation convention. The coefficients $X^i$ are the components of the tangent vector $X \in T_p(S)$ in the local basis induced by the parametrization $x$.

The tangent bundle of $S$ (as a set) is defined as $T(S) := \bigcup_{p \in S} \{ p \} \times T_p(S)$. A (tangential) vector field$^1$ of class $C^k$ ($k \geq 0$) is a map $X : S \to T(S)$ with the following properties:

(i) $X(p) \in \{ p \} \times T_p(S)$ for all $p \in S$, i.e., $X$ is a section.

(ii) For any parametrization $x : U \to V$, the component functions $p \mapsto X^i(p)$ in the representation

$$(2) \quad X(p) = X^i(p) \frac{\partial x}{\partial \zeta_i}(x^{-1}(p)), \quad p \in V$$

are of class $C^k$ on $V$.

Finally we recall the notion of divergence of a $C^k$ vector field $X$ for $k \geq 1$. Suppose that $X$ has the representation (2) w.r.t. the parametrization $x$. Define the differential operators

$^1$Vector fields on the surface and their corresponding function spaces will be denoted by bold-face symbols.

This manuscript is for review purposes only.
\( (\partial_i \cdot)(p), i = 1, 2, \) by

\[
(\partial_i f)(p) := \frac{\partial (f \circ x)}{\partial \zeta_i} (x^{-1}(p))
\]

for \( C^1 \) functions \( f \) defined in a neighborhood of \( p \in S \). Following [50, Chapter 1.2.3], we have

\[
(\text{div} \, X)(p) := \frac{1}{\sqrt{\det G(p)}} \partial_i \left( X^i \sqrt{\det G} \right)(p), \quad p \in V.
\]

Here \( G \) is the metric tensor for the parametrization \( x \) at \( p \), defined by its entries

\[
(g_{ij})(p) := \begin{pmatrix} \frac{\partial x}{\partial \zeta_i} (x^{-1}(p)) \\ \frac{\partial x}{\partial \zeta_j} (x^{-1}(p)) \end{pmatrix}^\top, \quad p \in V,
\]

where \( a^\top b \) denotes the Euclidean inner product in the ambient space \( \mathbb{R}^3 \). Since the vectors \( \frac{\partial x}{\partial \zeta_i} (x^{-1}(p)), i = 1, 2 \) are linearly independent, \( G(p) \) is positive definite and also symmetric.

Due to the fact that every tangent space to a point \( p \) on the surface inherits the standard inner product from the ambient space \( \mathbb{R}^3 \), we can introduce the pointwise inner product of two \( C^k \) vector fields as \( (X, Y)_2 := X^\top Y \), resulting in a real-valued \( C^k \) function on \( S \). Moreover, the pointwise 2-norm of a vector field will be denoted by \( |X|_2 = (X, X)_2^{1/2} \). When \( X \) and \( Y \) are given by representations of type (2) w.r.t. a parametrization \( x \), then the product \( (X, Y)_2 \) is represented by \( g_{ij} X^i Y^j \). Notice that the notions of tangent space, tangent bundle, vector fields and their divergence, as well as the inner product \( (\cdot, \cdot)_2 \) and the norm \( |\cdot|_2 \) are intrinsic quantities, i.e., independent of the atlas used to describe the surface \( S \).

**Assumption 2.** Throughout this paper we will assume that the smooth surface \( S \subset \mathbb{R}^3 \) is compact and connected.

It can be shown that for smooth surfaces, connectedness implies that any two points can be joined by a smooth path. As a further consequence of Assumption 2 \( S \) is also orientable; cf. [2, Prob. 2.43]. That is, the Jacobian of the transition map \( x^{-1} \circ y \) between any two intersecting parametrizations has positive determinant on its domain of definition.

### 2.2. Sobolev Functions and Functions of Bounded Variation

In this section we recall the notions of Lebesgue and Sobolev spaces \( L^p(S) \) and \( H^{1,p}(S) \) on the surface \( S \), as well as the spaces \( \mathbf{H}(\text{div}; S) \) and \( BV(S) \) required for the subsequent analysis.

For \( m \in \mathbb{N}_0 \), \( C^m(S) \) denotes the space of \( C^m \) functions on the surface \( S \). Moreover, \( C^m(S; T(S)) \) denotes the space of \( C^m \) vector fields. As usual, the support of a function \( f \) is defined as

\[
\text{supp } f := \text{cl} \{ p \in S : f(p) \neq 0 \}
\]

with \( \text{cl} \, C \) denoting the closure of a set \( C \subset S \).

*This manuscript is for review purposes only.*
We begin with the recollection of the spaces $L^p(S)$. Let $f$ be a continuous function on $S$ with support in the range $V$ of a parametrization $x : U \to V$. Then, we have by definition

$$\int_S f \, ds := \int_U f(x(u)) \sqrt{\det G(x(u))} \, du,$$

where the measure $ds$ is defined as $ds = \sqrt{\det G} \circ x \, du$, with $du$ denoting the Lebesgue measure in $\mathbb{R}^2$. This definition of the integral extends to arbitrary continuous functions on $S$ by using a partition of unity; cf. [33, Ch. 1.2]. As it is shown there the integrability of a function and the value of its integral over $S$ depend neither on the atlas nor on the partition of unity used.

For $1 \leq p < \infty$ the space $L^p(S)$ is defined as the completion of $C^\infty(S)$ w.r.t. the norm

$$\|f\|_{L^p(S)} := \left( \int_S |f|^p \, ds \right)^{1/p}.$$

We also recall that $L^\infty(S)$ is defined as the space of functions such that

$$\|f\|_{L^\infty(S)} := \operatorname{ess sup}_{p \in S} |f(p)| < \infty.$$

Naturally, these definitions extend to vector fields $f \in L^p(S; T(S))$. For instance, we have

$$\|f\|_{L^p(S; T(S))} := \left( \int_S |f|^p_2 \, ds \right)^{1/p}.$$

The spaces $L^2(S)$ and $L^2(S; T(S))$ are Hilbert spaces w.r.t. the usual inner products $(\cdot, \cdot)_{L^2(S)}$ and $(\cdot, \cdot)_{L^2(S; T(S))}$.

We are now in the position to define functions of bounded variation on surfaces satisfying Assumption 2. Background material on BV functions on flat domains can be found, for instance, in [28], [58, Ch. 5] or [1, Ch. 10].

Definition 3 (see also [39, Sect. 3.1] or [6, Sect. 4]). A function $u \in L^1(S)$ belongs to $BV(S)$ if the TV-seminorm defined by

$$\int_S |\nabla u| := \sup \left\{ \int_S u \, \operatorname{div} \, \eta \, ds : \eta \in V \right\}$$

is finite, where

$$V := \{ \eta \in C^\infty(S; T(S)) : |\eta(p)|_2 \leq 1 \text{ for all } p \in S \}.$$

We equip the space $BV(S)$ with the norm

$$\|u\|_{BV(S)} = \|u\|_{L^1(S)} + \int_S |\nabla u|, \quad u \in BV(S).$$

This manuscript is for review purposes only.
It is worth remarking that, as in the planar case, \( \int_S |\nabla u| = \int_S |\nabla u| \, ds \) holds for all functions \( u \in C^\infty(S) \) and indeed for \( u \in H^{1,1}(S) \); see [50, p.18] and Definition 4 below. Notice that both contributions to the norm \( \| \cdot \|_{BV(S)} \) are independent of the parametrization. We also remark that the space \( C^\infty(S; T(S)) \) can be replaced by \( C^1(S; T(S)) \) without affecting the definition; compare [1, Def. 10.1.1], [58, p.221] or [6].

According to Definition 3, it is clear that the embedding \( BV(S) \hookrightarrow L^1(S) \) holds. Next, we are going to prove that even \( BV(S) \hookrightarrow L^2(S) \) is valid, as is known for two-dimensional flat domains; see for instance [1, Th. 10.1.3]. This result is essential to establish the well-posedness of (1) in the sequel. Its proof requires the notion of intermediate convergence of \( BV(S) \) functions as well as the concept of first-order Sobolev spaces \( H^{1,p}(S) \). We summarize only the essential concepts and refer the reader to [32, 33] for an in-depth introduction to Sobolev spaces on manifolds.

**Definition 4.** Let \( u \in C^\infty(S) \). We call \( \nabla u : S \to \mathbb{R}^3 \) the gradient, defined locally in terms of any parametrization \( x \) by

\[
\nabla u := g^{ij} (\partial_i u) \frac{\partial x}{\partial \xi^j}.
\]

The gradient assigns to each point \( p \in S \) a vector \( (\nabla u)(p) \) in \( T_p(S) \subset \mathbb{R}^3 \) verifying

\[
((\nabla u)(p), v)_2 = v^i (\partial_i u)(p)
\]

for all \( v = v^i \frac{\partial x}{\partial \xi^i} (x^{-1}(p)) \in T_p(S) \). Here, \( g^{ij} \) are the components of the inverse of the metric tensor \( G = (g_{ij}) \). Furthermore, we define

\[
|\nabla u|_2 := (g^{ij} (\partial_i u) (\partial_j u))^{1/2}.
\]

Now for \( 1 \leq p < \infty \) and a function \( u \in C^\infty(S) \), define the norm

\[
\|u\|_{H^{1,p}(S)} := \left( \|u\|_{L^p(S)}^p + \int_S |\nabla u|_2^p \, ds \right)^{1/p}.
\]

The Sobolev space \( H^{1,p}(S) \) is then given by

\[ H^{1,p}(S) := \text{cl} \left( C^\infty(S) \right) \]

where the closure is w.r.t. the norm (8).

The counterpart of the following definition in the classical framework can be found in [1, Definition 10.1.3].

**Definition 5.** Let \( \{u_n\} \) be a sequence of functions in \( BV(S) \) and suppose \( u \in BV(S) \). We say that \( u_n \to u \) in the sense of intermediate convergence if

\[
(i) \ u_n \to u \text{ strongly in } L^1(S) \text{ and }
\]

This manuscript is for review purposes only.
The following lemma can be proved analogously as in [1, Th. 10.1.2]. The proof uses a partition-of-unity argument as well as mollification.

**Lemma 6.** For any \( u \in BV(S) \), there exists a sequence \( \{u_k\} \subset C^\infty(S) \) with \( u_k \to u \) in the intermediate sense.

**Proposition 7.** The space \( BV(S) \), equipped with the norm (7), is a Banach space and the embedding

\[
BV(S) \hookrightarrow L^p(S)
\]

holds for all \( 1 \leq p \leq 2 \).

**Proof.** The first part of the claim can be shown along the lines of [1, Prop. 10.1.1, Th. 10.1.1] taking into consideration the definition of the \( \|\cdot\|_{L^p(S)} \) given in (5). Hence, let us focus on the proof of the second claim. To this aim, let us define \( \{u_n\} \subset C^\infty(S) \) which converges to \( u \in BV(S) \) in the sense of intermediate convergence. Due to [33, Th. 2.6], we have that the embedding \( H^{1,q}(S) \hookrightarrow L^p(S) \) is continuous for all \( q \in [1,2) \) and \( p = \frac{2q}{2-q} \). In particular, for \( q = 1 \) and \( p = 2 \) we have \( H^{1,1}(S) \hookrightarrow L^2(S) \), so there exists \( c > 0 \) such that

\[
\left( \int_S |u_n|^2 \, ds \right)^{1/2} \leq c \left( \int_S |u_n| \, ds + \int_S |\nabla u_n|^2 \, ds \right)
\]

since \( \{u_n\} \subset C^\infty(S) \subset H^{1,1}(S) \). The sequence on the right hand side is bounded, and thus a subsequence of \( \{u_n\} \) convergences weakly in \( L^2(S) \). Since \( u_n \to u \) in \( L^1(S) \), we must have \( u_n \rightharpoonup u \) in \( L^2(S) \). By weak sequential lower semicontinuity of the norm,

\[
\left( \int_S |u|^2 \, ds \right)^{1/2} \leq \liminf_{n \to \infty} \left( \int_S |u_n|^2 \, ds \right)^{1/2}
\]

\[
\leq \liminf_{n \to \infty} c \left( \int_S |u_n| \, ds + \int_S |\nabla u_n|^2 \, ds \right)
\]

\[
= c \left( \|u\|_{L^1(S)} + \int_S |\nabla u| \right) = c \|u\|_{BV(S)}.
\]

Then, applying [33, Cor. 2.1], we have that \( BV(S) \hookrightarrow L^p(S) \) holds for all \( p \in [1,2] \).

To close this section we introduce the following space of vector fields, which will play a fundamental role throughout the paper,

\[
H(\text{div}; S) := \{ v \in L^2(S; T(S)) : \text{div } v \in L^2(S) \}.
\]

We equip this space with the norm

\[
\|v\|_{H(\text{div}; S)} := \left( \|v\|^2_{L^2(S; T(S))} + \|\text{div } v\|^2_{L^2(S)} \right)^{1/2},
\]
which is induced by the inner product

\[(v, w)_{H(\text{div}; S)} := (v, w)_{L^2(S; T(S))} + (\text{div} v, \text{div} w)_{L^2(S)} \]

\(H(\text{div}; S)\) is a Hilbert space.

3. The Fenchel Predual on Surfaces. The dual problem of TV–\(L^2\) has been stated in various references; see for instance [14, 15, 19]. In particular, it appears in [39] exactly for problem (1) on smooth surfaces. However, the arguments used to derive the dual problem in these references were all formal, and in particular no function space was assigned to the problem. To the best of our knowledge [35] is the only reference where this analysis is made rigorous. Due to the lack of reflexivity of BV spaces, the dual and predual problems are different. As has been shown in [35] the predual, posed as a problem in \(H(\text{div})\), is the appropriate concept.

In this section we adapt this rigorous analysis to problem (1) on the surface \(S\). As expected from [39] the predual problem is a quadratic optimization problem for the predual tangent field \(p \in H(\text{div}; S)\) with pointwise constraints on the surface; see (15) below. We will show that both problems are equivalent and that the primal solution can be recovered from the predual solution. We wish to point out that the constraints \(|p|_2 \leq \beta\) arising in our setting are nonlinear. This is in contrast with [35, eq. (2.1)], where simple bounds \(-\beta 1 \leq p \leq \beta 1\) were obtained due to a slightly different definition of the TV-seminorm, which is, however, not invariant under changes of the parametrization.

Solving the predual problem has a number of advantages compared to solving the primal problem directly. First, we do not have to deal with the discretization of the nonsmooth term \(\int_S |\nabla u|\) in the finite element context, nor employ an optimization algorithm for the nonsmooth problem (1); we mention however that such a program was carried out in a different context in [4]. Second, as was pointed out in [5], the finite element solution of minimization problems in \(BV\) spaces may suffer from low convergence rates. Finally, as was observed previously in [14, 15, 19, 35], we mention that the predual variable \(p\) serves as an edge detector in the image.

Let us recall some preliminary results from convex analysis; see for instance [57, Ch. 2.8]. Given two locally convex Hausdorff spaces \(X, Y\), two proper convex functions \(F : X \to \mathbb{R} \cup \{\infty\}\), \(G : Y \to \mathbb{R} \cup \{\infty\}\) as well as a bounded linear map \(A : X \to Y\) we have, due to the Fenchel-Young inequality, the relation of weak duality

\[
\inf_{x \in X} \{ F(x) + G(Ax) \} \geq \sup_{y^* \in Y^*} \{ -F^*(A^*y^*) - G^*(-y^*) \} .
\]

Here \(F^* : X^* \to \mathbb{R}\) and \(G^* : Y^* \to \mathbb{R}\) are the Fenchel conjugates of \(F\) and \(G\), defined by

\[
F^*(x^*) = \sup_{x \in X} \{ (x, x^*) - F(x) \} \quad \text{and} \quad G^*(y^*) = \sup_{y \in Y} \{ (y, y^*) - G(y) \}
\]

and \(X^*\) and \(Y^*\) are the topological dual spaces of \(X\) and \(Y\). Moreover \(A^* : Y^* \to X^*\) stands for the adjoint operator of \(A\). Under the assumption

\[
A(\text{dom} F) \cap \{ y \in Y : G \text{ is continuous in } y \} \neq \emptyset
\]

This manuscript is for review purposes only.
strong Fenchel duality holds, i.e.,
\[
\inf_{x \in X} \{ F(x) + G(Ax) \} = \max_{y^* \in Y^*} \{ - F^*(A^*y^*) - G^*(-y^*) \}.
\]

We now apply this to our specific setting. As in [35] we define the operator \( B \) as
\[
B := \alpha \text{id} + K^*K \in L(L^2(S)),
\]
where id is the identity mapping. Furthermore, we define
\[
\|w\|_{B^{-1}} = \langle w, B^{-1}w \rangle_{L^2(S)} = (w, w)_{B^{-1}}
\]
for any \( w \in L^2(S) \). Notice that in view of our standing assumptions (\( \alpha > 0 \) or \( K^*K \) coercive), \( \|w\|_{B^{-1}} \) is a norm equivalent to the standard norm of \( L^2(S) \).

\[\text{Theorem 8.} \quad \text{The Fenchel dual problem of (15) is equivalent to the optimization problem (1). In other words, (15) can be seen as the predual of the primal problem (1).}\]

\[\text{Proof.} \quad \text{The proof proceeds along the lines of [35, Th. 2.2]. We invoke the Fenchel duality theory in the setting } X = H(\text{div}; S), Y = L^2(S) \text{ and } A = - \text{div} : X \to Y. \text{ It is convenient to identify } Y \text{ with } Y^* \text{ so that } A^* = - \text{div}^* : Y \to X^*. \text{ Define the functions } F : H(\text{div}; S) \to \mathbb{R} \text{ and } G : L^2(S) \to \mathbb{R} \text{ as}\]
\[
F(p) := \begin{cases} 0 & \text{if } |p|_2 \leq \beta \, \text{ a.e. on } S, \\ \infty & \text{otherwise,} \end{cases}
\]

\[
G(v) := \frac{1}{2} \| v - K^*f \|_{B^{-1}}^2.
\]

From [35] we have
\[
G^*(v^*) = \sup_{v \in L^2(S)} \{ (v, v^*)_{L^2(S)} - G(v) \} = \frac{1}{2} \| K v^* + f \|_{L^2(S)}^2 + \frac{\alpha}{2} \| v^* \|_{L^2(S)}^2 - \frac{1}{2} \| f \|_{L^2(S)}^2
\]
for \( v^* \in L^2(S) \). With regard to \( F^* : H(\text{div}; S)^* \to \mathbb{R} \) it is clear that
\[
F^*(p^*) = \sup_{p \in \mathcal{B}_0} \langle p, p^* \rangle_{H(\text{div}; S), H(\text{div}; S)^*}
\]
holds, where
\[
\mathcal{B}_0 := \{ p \in H(\text{div}; S) : |p|_2 \leq \beta \, \text{ a.e. on } S \}.
\]
Next we argue that the set
\[ B_1 := \{ p \in C^\infty(S; T(S)) : |p|_2^2 \leq \beta \text{ a.e. on } S \} \]
is dense in \( B_0 \) in the topology of \( H(\text{div}; S) \). In fact, in the case \( S = \mathbb{R}^n \) and in the absence of bounds \( |p|_2^2 \leq \beta \), the result is classical; see for instance [30, Theorem I.2.4]. The proof can be done by mollification. Given \( p \in H(\text{div}; \mathbb{R}^n) \), define \( p_\epsilon := p \ast \eta_\epsilon \), where \( \{ \eta_\epsilon \} \) is a family of Friedrichs mollifiers and \( \ast \) denotes convolution. Since mollification preserves pointwise bounds, the convergence \( p_\epsilon \to p \) in \( H(\text{div}; \mathbb{R}^n) \) remains true under the constraint \( |p|_2^2 \leq \beta \). In case \( S \) is a compact surface, consider a finite atlas \( \{ x_\ell \}_{\ell=1}^N \) of parametrizations \( x_\ell : U_\ell \to V_\ell \). Moreover, let \( \{ \theta_\ell \}_{\ell=1}^N \) denote a partition of unity subordinate to \( \{ U_\ell \} \). Consider \( p \in B_0 \) and write \( p = \sum_{\ell=1}^N p_\theta_\ell \). Then \( [(p \theta_\ell) \circ x_\ell]_\epsilon \to [(p \theta_\ell) \circ x_\ell] \) in \( L^2(\mathbb{R}^2) \), which is enough to confirm the convergence \( p_\epsilon = \sum_{\ell=1}^N [(p \theta_\ell) \circ x_\ell]_\epsilon \circ x_\ell^{-1} \to \sum_{\ell=1}^N [(p \theta_\ell) \circ x_\ell] \circ x_\ell^{-1} = p \) in \( L^2(S) \) due to the boundedness of \( \det(G(x_\ell)) \) for all \( \ell = 1, \ldots, N \); see (4). By applying an analogous argument, \( \text{div} p_\epsilon \) can likewise be shown to converge to \( \text{div} p \) in \( L^2(S) \), which proves the density of \( B_1 \) in \( B_0 \).

Consequently, it is enough to take the supremum over \( p \in B_1 \) in (\ast\ast). Hence, for every
\[ u \in L^2(S) \]
we obtain
\[ F^*((-\text{div})^*u) = \sup \{ \langle p, (-\text{div})^*u \rangle_{H(\text{div}; S), H(\text{div}; S)^*} : p \in B_1 \} \]
\[ = \sup \{ \langle u, -\text{div} p \rangle_{L^2(S)} : p \in B_1 \} \]
\[ = \beta \sup \{ \langle u, -\text{div} p \rangle_{L^2(S)} : p \in C^\infty(S; T(S)) : |p|_2^2 \leq 1 \text{ a.e. on } S \}. \]

According to Definition 3 we get
\[ F^*((-\text{div})^*u) = \begin{cases} \beta \int_S |\nabla u| & \text{if } u \in BV(S), \\ \infty & \text{otherwise.} \end{cases} \]

Thus, since \( F \) and \( G \) are proper and convex and condition (12) is fulfilled for them, there is no duality gap between the optimal values of (1) and (15), i.e., (10) becomes an equality and
\[ \inf_{p \in H(\text{div}; S)} \{ F(p) + G(-\text{div} p) \} \]
\[ = \sup_{u \in L^2(S)} \{ -F^*((-\text{div})^*u) - G^*(-u) \} \]
\[ = \sup_{u \in BV(S)} \{ -F^*((-\text{div})^*u) - G^*(-u) \} \]
\[ = \sup_{u \in BV(S)} \left\{ -\frac{1}{2} \| Ku - f \|_{L^2(S)}^2 - \frac{\alpha}{2} \| u \|_{L^2(S)}^2 - \beta \int_S |\nabla u| \right\} + \frac{1}{2} \| f \|_{L^2(S)}^2. \]

Finally, it is immediate to check that (19) is in turn equivalent to (1). \hfill \blacksquare

**Corollary 9.** Problem (1) and its predual (15) are solvable.
Proof. The existence of a solution to (15) follows from standard arguments using the direct method of the calculus of variations and the embedding $BV(S) \hookrightarrow L^2(S)$ proved in Proposition 7. We refer the reader to the proof of Proposition 11 for details, where the same arguments are applied to a variation of (15). Regarding the solvability of (1), since condition (12) is fulfilled for $F$ and $G$ defined in the proof of Theorem 8, we conclude that the supremum in the RHS of (19) is attained, so (13) holds and the optimization problem (1) is solvable. □

The following theorem shows how the optimal solutions to (1) and (15) are related to each other.

**Theorem 10.** Suppose that $\overline{p}$ is an optimal solution to (15) and $\overline{u}$ is optimal to (1). Then

$$B \overline{u} = \text{div} \overline{p} + K^*f.$$  \hspace{1cm} (20)

Proof. Suppose that $\overline{p}$ and $\overline{u}$ are optimal to (15) and (1), respectively. Then the following conditions are fulfilled, see for instance [27, Prop. 4.1],

$$(-\text{div})^*\overline{u} \in \partial F(\overline{p}) \quad \text{in} \quad H(\text{div}; S)^*,$$

$$-\text{div} \overline{p} \in \partial G^*(-\overline{u}) \quad \text{in} \quad L^2(S),$$  \hspace{1cm} (21)

where $\partial F$ stands for the standard representation of the subdifferential of the convex function $F : H(\text{div}; S) \rightarrow \mathbb{R}$, and $\partial G^*$ is defined analogously. The second condition in (21) is equivalent to

$$G^*(-\overline{u}) + G(-\text{div} \overline{p}) - (-\text{div} \overline{p}, -\overline{u})_{L^2(S)} = 0.$$  \hspace{1cm} (22)

Using the expressions (16) and (17) for $G$ and $G^*$, this becomes

$$\frac{1}{2} \|K(-\overline{u}) + f\|^2_{L^2(S)} + \frac{\alpha}{2} \|\overline{u}\|^2_{L^2(S)} - \frac{1}{2} \|f\|^2_{L^2(S)} + \frac{1}{2} \|\text{div} \overline{p} + K^*f\|^2_{B^{-1}} = (\text{div} \overline{p}, \overline{u})_{L^2(S)},$$  \hspace{1cm} (23)

or equivalently,

$$\frac{1}{2} \|K\overline{u}\|^2_{L^2(S)} + \frac{\alpha}{2} \|\overline{u}\|^2_{L^2(S)} + \frac{1}{2} \|\text{div} \overline{p} + K^*f\|^2_{B^{-1}} = (\text{div} \overline{p} + K^*f, \overline{u})_{L^2(S)},$$  \hspace{1cm} (24)

Applying the definition of $B$, see (14), we obtain

$$\frac{1}{2} \|u - B^{-1}(\text{div} \overline{p} + K^*f)\|^2_B = 0$$  \hspace{1cm} (25)

and (20) follows. □

**4. Algorithmic Approach and Finite Element Discretization.** In this section we describe a novel approach for solving (1) via its predual (15). Once again, recall that the pointwise constraints $|p|_2 \leq \beta$ are nonlinear. This is in contrast with $|p|_\infty \leq \beta$ obtained in [35] due to a slightly different definition of the TV-seminorm. The nonlinearity of the constraint would render the analysis and application of a primal-dual active set method more challenging although this has been successfully pursued, for instance, in [34, 55] in different contexts.
Our solution approach is based on a logarithmic barrier method to deal with the inequality constraints. Consequently, we consider the following family of convex problems for a decreasing sequence of barrier parameters $\mu \searrow 0$:

\[
\begin{cases}
\text{Minimize} & \frac{1}{2} \| \text{div} \, p + K^* f \|_{B^{-1}}^2 - \mu \int_S \ln \left( \beta^2 - |p|^2 \right) \, ds \\
\text{subject to} & |p|_2 \leq \beta \text{ a.e. on } S.
\end{cases}
\]

(22)

Notice that the constraint $|p|_2 \leq \beta$ is explicitly kept in (22) for mathematical convenience and it avoids the need to define the logarithmic barrier term for negative arguments. For any fixed barrier parameter $\mu > 0$, problem (22) will be solved using Newton’s method. The constraint $|p|_2 \leq \beta$ is not enforced explicitly but its satisfaction will be monitored throughout the Newton iterations. More details concerning the implementation are given in section 5.

For convenience, we use the abbreviations

\[
H(p) := \frac{1}{2} \| \text{div} \, p + K^* f \|_{B^{-1}}^2 \quad \text{and} \quad b(p) := -\mu \int_S \ln \left( \beta^2 - |p|^2 \right) \, ds
\]

in the sequel.

### 4.1. Existence and Uniqueness for the Preludal Barrier Problem

The analysis of interior point methods in $L^p$ spaces including a convergence analysis of the central path has been addressed in [46, 54] in the context of optimal control problems. Notice that the presence of the logarithmic barrier term helps to overcome the lack of strict convexity of the objective in (15). We therefore obtain the following result.

**Proposition 11.** For every $\mu > 0$, problem (22) possesses a unique solution $p \in H(\text{div}; S)$.

**Proof.** It is easy to check that the objective $H(p) + b(p)$ is bounded below by $b(0) = -\mu (\text{area } S) \ln(\beta^2)$ but it may attain the value $\infty$. Let us consider a minimizing sequence $\{p_n\}$. Owing to the boundedness of both terms in the objective as well as $|p_n|_2 \leq \beta$, $\{p_n\}$ is bounded in $H(\text{div}; S)$. Hence there exists a subsequence (again denoted by $\{p_n\}$) such that $p_n \rightharpoonup \bar{p}$ in $L^2(S; T(S))$ holds with $|\bar{p}|_2 \leq \beta$ a.e. on $S$, as well as $\text{div} \, p_n \rightharpoonup \text{div} \, p$ in $L^2(S)$.

By weak sequential lower semicontinuity of $H$,

\[
H(\bar{p}) = \frac{1}{2} \| \text{div} \, \bar{p} + K^* f \|_{B^{-1}}^2 \leq \liminf_{n \to \infty} \frac{1}{2} \| \text{div} \, p_n + K^* f \|_{B^{-1}}^2
\]

holds. Let us argue that $b$ is also weakly sequentially lower semicontinuous w.r.t. $L^2(S; T(S))$.

To this end, it suffices to show that $b$ is sequentially lower semicontinuous w.r.t. the strong topology of $L^2(S; T(S))$ on

\[
\mathcal{B} := \{ q \in L^2(S; T(S)) : |q|_2 \leq \beta \text{ a.e. on } S \},
\]

since $\mathcal{B}$ is closed and convex and $b$ is convex in $\mathcal{B}$. Arguing similarly as in the proof of [54, Proposition 2], suppose that $q_n \to q$ in $L^2(S; T(S))$ holds, where $q_n \in \mathcal{B}$, and thus $q \in \mathcal{B}$.
holds as well. We have to show $b(q) \leq \liminf_{n \to \infty} b(q_n)$, which is clear if the right hand side is $\infty$. In case $\liminf_{n \to \infty} b(q_n) < \infty$, we can select a subsequence, denoted by $\{q_j\}$, such that

$$\lim_{j \to \infty} b(q_j) = \liminf_{n \to \infty} b(q_n) \quad \text{and} \quad q_j \xrightarrow{j \to \infty} q \text{ a.e. on } S.$$  

In particular, $b(q_j) \leq C$ holds for all $j$.

Let us define

$$g_n := -\mu \ln \max\{\beta^2 - |q_n|^2, 1\}, \quad h_n := -\mu \ln \min\{\beta^2 - |q_n|^2, 1\},$$

$$g := -\mu \ln \max\{\beta^2 - |q|^2, 1\}, \quad h := -\mu \ln \min\{\beta^2 - |q|^2, 1\}.$$  

Since $x \mapsto \ln x$ is Lipschitz continuous with Lipschitz constant 1 on $[1, \infty)$, we have

$$|g_n - g| \leq \mu \max\{\beta^2 - |q_n|^2, 1\} - \max\{\beta^2 - |q|^2, 1\} \leq \mu |q_n|^2 - |q|^2 \leq \mu \beta^2$$

holds. Notice that $\lim_{j \to \infty} q_j = q$ a.e. on $S$ implies $\lim_{j \to \infty} g_j = g$ and $\lim_{j \to \infty} h_j = h$ a.e. on $S$. Hence Lebesgue's dominated convergence theorem implies

$$\lim_{j \to \infty} \int_S g_j \, ds = \int_S g \, ds.$$  

Consequently,

$$\lim_{j \to \infty} \int_S h_j \, ds = \lim_{j \to \infty} b(q_j) - \lim_{j \to \infty} \int_S g_j \, ds$$

exists as well.

Using $h_n \geq 0$ and $|g_n| \leq \mu |q_n|^2 - \beta^2 + 1|$, we obtain

$$0 \leq \int_S h_j \, ds \leq C - \int_S g_j \, ds \leq C + \mu \int_S |q_j|^2 - \beta^2 + 1 \, ds \leq C + C'$$

for all $j$. By Fatou’s lemma, we thus conclude

$$0 \leq \int_S h \, ds = \int_S \lim_{j \to \infty} h_j \, ds \leq \lim_{j \to \infty} \int_S h_j \, ds \leq C + C'.$$

This implies

$$\liminf_{n \to \infty} \int_S (g_n + h_n) \, ds = \liminf_{n \to \infty} b(q_n) = \lim_{j \to \infty} b(q_j) \geq \int_S (g + h) \, ds = b(q).$$

Consequently, both summands $H$ and $b$ in the objective are weakly sequentially lower semi-continuous, which implies that $\overline{\mathbf{p}}$ is a (global) minimizer of (22).

To show its uniqueness, we verify that the second part of the objective $b$ is strictly convex where it is finite. To this end, let $p_1$ and $p_2$ be two elements of $B$ where $b(p_1), b(p_2) < \infty$ and
\textbf{TV IMAGE RECONSTRUCTION PROBLEMS ON SMOOTH SURFACES} 15

\begin{align*}
\mathbf{p}_1 \text{ not equal to } \mathbf{p}_2 \text{ a.e. on } S. \text{ Then by classical arguments there exists a set } E \subset S \text{ of positive surface measure and } \varepsilon > 0 \text{ such that } |\mathbf{p}_1 - \mathbf{p}_2|_2 \geq \varepsilon \text{ a.e. on } E. \\
\text{Let us define } g(\mathbf{p}) := |\mathbf{p}|^2_2 \text{ and } h(z) := -\mu \ln(\beta^2 - z), \text{ whence } b(\mathbf{p}) = \int_S h(g(\mathbf{p})) \, ds \text{ holds.}\\
\text{On the set } E, \text{ we have the following pointwise estimate due to the strong convexity of } g, \\
(\star) \quad g(\lambda \mathbf{p}_1 + (1 - \lambda) \mathbf{p}_2) - \lambda g(\mathbf{p}_1) - (1 - \lambda) g(\mathbf{p}_2) = -\lambda (1 - \lambda) |\mathbf{p}_1 - \mathbf{p}_2|^2_2 \leq -\lambda (1 - \lambda) \varepsilon^2 \\
\text{for all } \lambda \in [0, 1]. \text{ Next we use that } h \text{ is convex and strictly increasing on } [0, \beta^2]. \text{ Its minimal slope is attained at } z = 0 \text{ so we have } h'(z) \geq h'(0) = \mu / \beta^2 \text{ for all } z \in [0, \beta^2]. \text{ Consequently,} \\
\text{we have } h(r) \geq h(\ell) + h'(\ell)(r - \ell) \geq h(\ell) + h'(0)(r - \ell) \text{ for all } 0 \leq \ell \leq r < \beta^2. \text{ Applying this} \\
estimate with } \ell = g(\lambda \mathbf{p}_1 + (1 - \lambda) \mathbf{p}_2) \text{ and } r = \lambda g(\mathbf{p}_1) + (1 - \lambda) g(\mathbf{p}_2) \text{ and using } (\star), \text{ we obtain} \\
h\bigg(g(\lambda \mathbf{p}_1 + (1 - \lambda) \mathbf{p}_2)\bigg) \leq h\bigg(\lambda g(\mathbf{p}_1) + (1 - \lambda) g(\mathbf{p}_2)\bigg) - \frac{\mu}{\beta^2} \lambda (1 - \lambda) \varepsilon^2. \\
\text{Using the convexity of } h \text{ we can estimate further} \\
h\bigg(g(\lambda \mathbf{p}_1 + (1 - \lambda) \mathbf{p}_2)\bigg) \leq \lambda h(g(\mathbf{p}_1)) + (1 - \lambda) h(g(\mathbf{p}_2)) - \frac{\mu}{\beta^2} \lambda (1 - \lambda) \varepsilon^2, \\
\text{which holds a.e. on } E. \text{ Similarly, we obtain the same estimate without the last term on } S \setminus E. \\
\text{Integrating these inequalities over } S, \text{ we finally obtain the estimate} \\
b(\lambda \mathbf{p}_1 + (1 - \lambda) \mathbf{p}_2) \leq \lambda b(\mathbf{p}_1) + (1 - \lambda) b(\mathbf{p}_2) - \frac{\mu}{\beta^2} \lambda (1 - \lambda) \varepsilon^2 \text{ (area } E), \\
\text{which confirms the strict convexity of } b \text{ on its domain.}
\end{align*}

Next we address the first-order necessary and sufficient optimality conditions for (22). The main difficulty compared to finite dimensional barrier methods is that one cannot a-priori exclude that the minimizer approaches the bound $|\mathbf{p}|_2 \leq \beta$ on parts of the surface, which complicates the discussion of differentiability of the barrier term. The proof uses techniques introduced in [52], where optimal control problems with pointwise simple bounds on the control and also the state were discussed. Although the present problem is generally simpler due to the absence of state constraints, the nonlinearity of the constraint $|\mathbf{p}|_2 \leq \beta$ requires modifications. We therefore do provide the proof of the following theorem but postpone it to the appendix to streamline the presentation.

\textbf{Theorem 12.} The vector field $\mathbf{p} \in H(\text{div}; S)$ is the unique solution for (22) if and only if $|\mathbf{p}|_2 \leq \beta$ holds a.e. on $S$ and

\begin{equation}
(\text{div } \mathbf{p} + K^* f, \text{ div } \delta \mathbf{p})_{B^{1,1}} + \mu \int_S 2 \frac{(\mathbf{p}, \delta \mathbf{p})_2}{\beta^2 - |\mathbf{p}|_2^2} \, ds = 0
\end{equation}

for all $\delta \mathbf{p} \in H(\text{div}; S).$
4.2. Implementation. All numerical studies are based on two different geometries obtained by scanning physical objects with the Artec Eva 3D scanner. The scanner software provides Wavefront .obj files, which contain a description of the geometry via vertices and triangles. In both examples the surface of the scanned object is closed, i.e., without boundary, in accordance with our analysis. The surface texture is provided by the scanner software as a 2D flat bitmap file (see Figure 1, left), together with a mapping of each physical surface triangle into said bitmap. Thus, originally the textured object is described by a varying number of pixels glued onto each surface triangle. Due to the impossibility of continuously mapping a closed surface onto the flat bitmap, there are necessarily discontinuities in the bitmap and there may also be regions which do not appear on the physical surface. Essentially, two adjacent triangles on the surface can be part of discontinuous regions in the texture file. This data is shown in Figure 1 for our first test case.

Figure 1. Left: Texture bitmap as delivered by the scanner software. Right: Texture mapped onto the geometry.

In order to apply our novel solution scheme, the above mentioned Wavefront object including the texture needs to be made available to the finite element library which is used to discretize the predual barrier problems (22). One way of achieving this is to provide the texture data $f$ at each quadrature point. However, for ease of implementation and processing within the finite element framework FEniCS [40], we instead converted the textured object into the finite element setting by interpolation. To account for both natural discontinuities in the texture as well as the discontinuity of the surface-to-texture mapping, we chose a discontinuous Lagrange (DG) finite element representation of the texture data $f$. To be more precise, let $P_r$ define the space of polynomials of maximum degree $r$, then the texture $f$ and the final output $u$ of our scheme are to fulfill $f|_K, u|_K \in P_r$ for all triangles $K$ of the scanned.
surface. Thus, \( u \) and \( f \) are elements of the \( \mathcal{DG}_{r} \) finite element space on the surface. The image data \( f \) is always scaled to the interval \([0, 1]\).

To carry out the texture preprocessing, we compute the spatial location for each degree of freedom of the surface DG function \( f \) within the texture bitmap and use the respective gray value at the nearest pixel. For color textures, this is realized via a vector valued DG function on the surfaces with values in the RGB color space. In the original Wavefront object each surface triangle usually obtains data from multiple texture pixels. Thus, in order to maintain an appropriate quality of the texture in the DG setting, higher order finite element spaces are needed, depending on the quality of the scan. Although in the original Wavefront object the number of pixels per triangle may vary significantly, we use a constant finite element order \( r = 2 \) or \( r = 3 \) in our examples.

### 4.3. Discretization of the Predual Variable by Raviart-Thomas Surface Finite Elements

Before recovering the image \( u \) we determine the predual (edge detector) vector field \( p \in H(\text{div}; S) \) via a sequence of barrier problems (22). For the latter we employ a conforming discretization by surface Raviart–Thomas (RT) finite elements. Although this choice of discretization is natural from the analytical point of view, RT elements seem to be rarely used in the context of image processing; see, however, \([23, 5]\). We therefore provide details in this section, focusing first on flat domains and later on general orientable surfaces. Notice that a discretization by continuous Lagrangian elements as in \([39]\) would also be conforming but it does not exhaust the space \( H(\text{div}; S) \) under mesh refinement. By contrast, only functions in the closed subspace of \( H^{1} \) vector fields can be approximated.

The Raviart–Thomas element space \( \mathcal{RT}_{r+1} \) \((r \geq 0)\) on a triangle \( T \) is designed to be the smallest vector valued polynomial space with \([P_{r}]^{2} \subset \mathcal{RT}_{r+1} \subset [P_{r+1}]^{2}\) such that the divergence maps onto \( P_{r} \); see \([47], [29, \text{Chapter 1.4.7}]\) or \([40, \text{Ch. 3.4.1}]\). In more explicit terms, the polynomial space for the \( \mathcal{RT}_{r+1} \) FE over a flat triangle \( T \) is given by \([P_{r}]^{2} + xP_{r}\), where \( x \in \mathbb{R}^{2} \) denotes the spatial coordinate. The dimension of this space is \((r+1)(r+3) = 3, 8, 15, 24, \ldots\)

In order to obtain an \( H(\text{div}) \)-conforming approximation, continuity of the normal component across all inter-element edges \( E \) must be ensured. Let us consider a triangulated flat domain \( \Omega \subset \mathbb{R}^{2} \) endowed with a conforming simplicial triangulation, i.e., any two intersecting triangles intersect either in a common vertex or a common edge. Then the global \( \mathcal{RT}_{r+1} \) space on \( \Omega \) is defined as

\[
\{ p \in [L^{1}(\Omega)]^{2} : p_{|T} \in \mathcal{RT}_{r+1}|_{T} \text{ for all triangles } T, \quad \langle p \cdot n \rangle_{E} = 0 \text{ for all interior edges } E \},
\]

Here \( \langle p \cdot n \rangle_{E} = p_{|T_{1}} \cdot n_{1} + p_{|T_{2}} \cdot n_{2} \) denotes the jump of \( p \cdot n \) across the edge \( E \) between elements \( T_{1} \) and \( T_{2} \) with edge normals \( n_{1} \) and \( n_{2} \), respectively. Continuity of the normal components, i.e., \( \langle p \cdot n \rangle_{E} = 0 \), is conveniently achieved by observing that the restriction \( (p \cdot n)|_{E} \) of any \( p \in \mathcal{RT}_{r+1}|_{T} \) to an edge \( E \) is a scalar polynomial of degree \( r \) in a single variable. \( (p \cdot n)|_{E} \) is therefore determined by \( r + 1 \) point values along the edge, and the continuity of \( p \cdot n \) across \( E \) amounts to a coincidence of the degrees of freedom (up to sign) on neighboring triangles. The remaining \((r+1)(r+3) - 3(r+1) = r(r+1)\) degrees of freedom on each triangle are defined

This manuscript is for review purposes only.
as evaluations of \( p \) at points inside the triangle\(^2\). In practical computations, each element \( T \subset \mathbb{R}^2 \) is obtained as an affine copy of a reference cell \( T_0 \subset \mathbb{R}^2 \) via the affine map \( F : T_0 \to T \).

Local basis functions must be mapped via the associated contravariant Piola transformation in order to preserve normal traces; see [49] for details.

The transition from the \( \mathcal{RT}_{r+1} \) space over a flat domain \( \Omega \subset \mathbb{R}^2 \) to one over a two-dimensional surface \( S \) is conceptionally straightforward by allowing \( F \) to be an affine map of rank 2 from \( T_0 \) to a surface triangle \( T \subset S \subset \mathbb{R}^3 \). We refer the reader to [48] for details.

A similar construction has been described in [39] for the case of linear continuous Lagrange elements, which do not require the Piola transform but which are not dense in \( H(\text{div}; S) \).

For the purpose of illustration, Figure 2 depicts a number of typical basis functions from the lowest-order space \( \mathcal{RT}_1 (r = 0) \) on a flat and spherical mesh.

![Figure 2](image)

**Figure 2.** Left: global basis function \( p \) from \( \mathcal{RT}_r \) associated with the degree of freedom located on the edge \( E \) adjacent to both triangles in \( \mathbb{R}^2 \). One clearly sees that \( (p \cdot n)|_E \) is constant, i.e., of degree \( r = 0 \), and continuous, i.e., \( \llbracket p \cdot n \rrbracket _E = 0 \) holds. On all other edges we have \( p \cdot n = 0 \). Middle: the same situation on a spherical mesh of topological dimension 2 in \( \mathbb{R}^3 \). Right: detail.

**Recovery of the Image.** The last step is to recover the image \( u \) from \( p \). As proved in Theorem 10 we have the relation

\[
(25) \quad u = B^{-1}(\text{div} \ p + K^* f).
\]

In our examples, which demonstrate denoising and inpainting, \( K, K^* \) and therefore \( B = \alpha \text{id} + K^* K \) are all pointwise operations, which do not involve differentiation. We therefore choose matching polynomial degrees, i.e., \( u \in \mathcal{DG}_r, \ p \in \mathcal{RT}_{r+1} \) and \( f \in \mathcal{DG}_r \). In terms of finite element functions, (25) is realized by solving an orthogonal projection problem in \( L^2(S) \), which is represented by a block-diagonal mass matrix in \( \mathcal{DG}_r \) and therefore inexpensive to solve.

**5. Numerical Results.**

\(^2\)Integral moments of \( p \) over the edges and triangles are sometimes used as degrees of freedom, in place of point evaluations. This does not change the space \( \mathcal{RT}_{r+1} \) but only its representation in terms of global basis functions.
5.1. Gray-Scale Denoising. In this section we consider the classical denoising problem with $K = \text{id}$. The initial test case is the scanned terracotta duck from Figure 1 but with the texture data converted to a gray scale. Recall that our image data is scaled to a range $[0,1]$. The geometry consists of 354 330 triangles and 177 167 vertices. The surface texture is mostly uniform, however there are some details around the eye and a second order DG function ($r = 2$) manages to resolve these quite well. Also there are sharp interfaces between body, beak and feet. As such, this object provides an excellent first test case and we expect that these interfaces are preserved by the total variation approach.

We added artificial noise based on a normal distribution with standard deviation $\sigma = 0.1$ and zero mean to each entry in the coefficient vector representing the image data $f$. For this
and the subsequent color denoising problem described in the following subsection, the value
\( \alpha = 0.0 \) is used and we consider only variations in \( \beta \). Notice that \( B \) in (14) is boundedly
invertible even for \( \alpha = 0 \) since \( K = \text{id} \) holds.

The denoising results shown in Figure 3 were obtained by a rather basic yet effective
interior point approach, in which a sequence of barrier approximations (22) to the predual
problem (15) are solved. Each instance of (22) is solved by applying Newton’s method to
the optimality system (24); see Algorithm 1. Based on numerical experience and the progress
observed in the image \( u \) recovered from (25), we used \( \mu_{\text{start}} = 1.0, \mu_{\text{end}} = 0.02 \) and \( N_{\text{max}} = 8 \)
in our study. The adjustment of the barrier parameter \( \mu \) is based on a simple backtracking
strategy, depending on the number of Newton iterations necessary to reach the termination
criterion \( \| \delta p \|_{L^2(S; T(S))} \leq 10^{-5} \). Here \( \delta p \) denotes the update calculated in each Newton step.

**Algorithm 1** Basic interior point method

\[
\begin{align*}
\mu & \leftarrow \mu_{\text{start}}, \ d_{\mu} \leftarrow 0.6, \ p \leftarrow 0 \\
\text{while } \mu > \mu_{\text{end}} \text{ do} & \\
\text{repeat} & \\
\text{Perform at most } N_{\text{max}} \text{ steps of Newton’s method to solve (24) for } p_{\text{new}}, \text{ given the current value of } \mu, \text{ starting from initial guess } p & \\
\text{s } & \leftarrow \text{ number of Newton iterations} \\
\mu & \leftarrow 1.2 \mu, \ d_{\mu} \leftarrow \min\{0.9, 1.2 d_{\mu}\} \\
\text{until Newton’s method did converge in } N_{\text{max}} \text{ steps;} & \\
\text{p } & \leftarrow p_{\text{new}} \text{ (accept new approximation)} \\
\text{if } 4 < s \text{ then} & \\
\text{\quad } d_{\mu} & \leftarrow 1.1 d_{\mu} \\
\text{else} & \\
\text{\quad } d_{\mu} & \leftarrow 0.9 d_{\mu} \\
\text{end} & \\
\mu & \leftarrow d_{\mu} \mu \text{ (propose } \mu \text{ for next instance of (24))} \\
\text{end} & \\
\end{align*}
\]

For both \( \beta = 0.1 \) and \( \beta = 0.3 \) the terminal value for \( \mu \) was reached after 4 reductions,
necessitating the solution of 5 instances of problem (22) with a Newton scheme. As shown
in Figure 4 each of these required between 3 and 5 Newton steps and this behavior is typical
for primal interior point methods. The total wall-clock time on four non-hyper threaded cores
of an Intel i5-4690 CPU running at 3.50 Ghz was slightly less than 40 minutes. Comparing
the results shown in Figure 3 one can see that—as expected—with increasing values of \( \beta \), the
noise is reduced more effectively and although the object looks progressively smoother due
to a reduction in contrast, sharp corners are preserved. The convergence plots for different
values of \( \beta \) are also shown in Figure 4. We expect that the efficiency can be improved and the
number of Newton steps reduced by employing a primal-dual interior point method. This is
left to future research.
Figure 4. Convergence of the inner (Newton) and outer iterations of the interior point method (Algorithm 1) for the duck denoising example. Different colors denote outer iterations and their decreasing barrier parameter $\mu$ with a stopping criterion of $\|\delta p\|_{L^2(S; T(S))} \leq 10^{-5}$ for Newton’s method. Different subplots for respective denoising parameters $\beta = 0.1, 0.2, 0.3, 0.5$.

5.2. Color Denoising. The second test case consists of a scanned shoe, whose data is provided by the Artec Group Inc.$^3$ under the Creative Commons Attribution 3.0 Unported License. The shoe consists of exactly 100 000 triangles and 50 002 vertices. It provides an excellent second test case because of discontinuous color changes given by the stripes, while at the same time there are also very fine features on the sole and a leathery texture on the outside. Noise is added in the same way to each of the RGB channels as described for the gray-scale test case above. In this example we chose to represent the color texture in terms of a vector valued discontinuous Galerkin function of order $r = 3$. This amounts to problems with 1.8 million degrees of freedom for the predual variable $p$ associated with a single color

$^3$https://www.artec3d.com

This manuscript is for review purposes only.
The denoising procedure was conducted individually per RGB channel. Initial, noisy and denoised objects are shown in Figure 5. The sharp edges between the stripes are preserved for different values of $\beta$. Details of the leather’s structure, most prominently visible in the yellow stripes in the noise-free image, start reappearing after the bulk of the noise is removed for $\beta = 0.5$. Notice however that some of these features are part of the geometric resolution and not just the texture. On the other hand, the dotted texture in the interior and part of the stitchings seem less discernible due to the reduced contrast for $\beta = 0.5$. As was noted earlier, the predual vector field $p$ can be interpreted as an edge detector, which is shown in Figure 6 for each RGB channel.

We also conducted experiments using the joint BV-norm, cf. [10]

$$\int_S |\nabla u| = \sup \left\{ \int_S \sum_{j=1}^3 u^j \text{div} \eta^j \, ds : \eta \in W \right\}$$

of the vector-valued unknown $u = (u^1, u^2, u^3) \in [BV(S)]^3$. In contrast to the scalar case the test space is now defined as

$$W := \left\{ (\eta^1, \eta^2, \eta^3) \in [C^\infty(S; T(S))]^3 : \sum_{j=1}^3 |\eta^j(p)|^2 \leq 1 \text{ for all } p \in S \right\},$$

compare [12, 26]. It can be expected that this modification better suppresses color fringes (similar to chromatic aberration), which occur when the value of two or more color channels have jump discontinuities at neighboring pixels. We refer the reader to [11, Chapter 6.3.4] for a discussion of alternative definitions of vector-valued BV norms in the context of color image restoration.

Use of the joint BV norm leads to the following modified predual problem compared to (15),

$$\begin{align*}
\text{Minimize} & \quad \frac{1}{2} \left\| \begin{pmatrix} \text{div} p^1 \\ \text{div} p^2 \\ \text{div} p^3 \end{pmatrix} \right\|_{B^{-1}}^2 + K^* f \\
\text{over} & \quad (p^1, p^2, p^3) \in [H(\text{div}; S)]^3 \\
\text{subject to} & \quad \sum_{j=1}^3 |p^j|^2 \leq \beta^2 \quad \text{a.e. on } S,
\end{align*}$$

see also [12]. Notice that the variables $p^j$ in this problem are coupled through the inequality constraints even if — as is the case in our examples — $K$ and $K^*$ act on each color component separately. This leads to an increased complexity of the problem. The conversion of the inequality constraints into a barrier term as in (22) is straightforward. In our numerical experiments, we did not experience significantly improved results using this model and therefore do not pursue this further here.
Figure 5. Shoe test case: noise free and noisy originals (top row) and denoising results for $\beta = 0.2$ and $\beta = 0.5$ (bottom row).
5.3. Color Inpainting. The problem of not being able to scan an object completely is quite common, as there might be areas the scanner cannot look into due to its size and the non-convexity and curvature of the object. The inside of the tip of a shoe might be such an example. Data corruption can be another reason for lack or loss of data. Although these issues concern both geometry and texture, the focus of this subsection is on the reconstruction of missing texture information alone.

We simulate the loss of texture data during the scan process on the outside of an object by setting to zero all degrees of freedom in the image data which belong to cells with indices in the range $30,000$ to $33,000$. This corresponds to a data loss of $3\%$. We denote this erased image region by $S_0 \subset S$. Due to the apparently layered scan process the index range chosen corresponds to bands or “stripes” as shown in Figure 7, which are better visible than unreachable areas inside the tip.

As usual for TV inpainting problems the mapping $K$ is now chosen so as to ignore the corrupted data. This leads to

$$ (Ku)(p) := \chi_{S \setminus S_0}(p) u(p), $$

where $\chi$ is an indicator function with value $1$ in the uncorrupted area $S \setminus S_0$. Since $K$ is self-adjoint and idempotent,

$$ K^*(Ku) = K(Ku) = Ku. $$

$(Ku)(p) - f(p) = 0$ holds for all $p \in S_0$ and the (corrupted) value of $f|_{S_0}$ does not increase the data fidelity part of the objective in (1).

Contrary to the denoising situation, $K^*K$ is no longer invertible and $\alpha > 0$ is required.
Using the definition of $K$ one easily deduces the formula

\[(B \, u)(p) = \begin{cases} \alpha \, u(p) & \text{for } p \in S_0 \\ (\alpha + 1) \, u(p) & \text{for } p \in S \setminus S_0. \end{cases}\]

The results of the inpainting test case are shown in Figure 7 for different values of $\beta$ and with $\alpha = 0.1$ constant for each case.

6. **Comparison with Split Bregman Method.** In this section we compare our approach with the split Bregman method [31], a widely used approach for the solution of (1) and related problems. This method was originally proposed for flat images defined on Cartesian grids. The
discretization of the image $u$ by finite elements, which is more natural than finite differences in the case of images on surfaces, necessitates minor modifications to the discrete realization of the algorithm, which we briefly describe in the sequel. In order to minimize the amount of technical details, we restrict the discussion to polynomial degree $r = 0$ for both algorithms, i.e., $u$ is piecewise constant. The higher order case will be discussed in a future work.

The split Bregman method addresses the primal problem (1) and introduces a second variable $d$ for the gradient, which is considered an independent unknown with equality $d = \nabla u$ holding only in the limit. Since $u \in D_{G_0}$, the gradient $\nabla u$ is a collection of line measures concentrated on the edges of the triangulation. Indeed, $\nabla u|_E = [u]_E n_E$ holds, where $E$ denotes an edge, $[u]_E$ is the scalar jump of $u$ across $E$, and $n_E$ is the edge normal. The sign of the jump and the direction of the normal depend on the choice of an arbitrary but fixed orientation. The TV-seminorm of $u$ is given by

$$|u|_{TV(S)} = \sum_E |[u]_E| \ell_E,$$

where $\ell_E$ is the length of the edge $E$.

In order to represent $\nabla u$ by a second variable $d$ exactly, one scalar quantity $d_E = [u]_E$ per edge suffices. Therefore the discrete function $d$ can be thought of as a member of a non-standard finite element space, consisting of discontinuous constants on the skeleton of the mesh. It is natural to choose the same discretization for the auxiliary quantity $b$ and represent it by collection of edge values $b_E$; see [31, Sect. 4]. It can be shown that $b$ represents the predual (edge detector) variable $p$ and the two quantities are related by

$$p|_E = \lambda b_E n_E.$$

The split Bregman method for piecewise constant surface images $u$ is based on the following discrete functional,

$$(27) \quad \beta \sum_E |d_E| \ell_E + \frac{\alpha}{2} [u]_{L^2(S)}^2 + \frac{1}{2} \|Ku - f\|_{L^2(S)}^2 + \frac{\lambda}{2} \sum_E |d_E - [u]_E - b_E|^2 \ell_E,$$

where $\lambda > 0$ is a given penalty parameter. The minimization of (27) w.r.t. $u$ in the denoising case ($K = \text{id}$) amounts to solving a linear system with a sparse matrix $A$, whose entries are

$$(A)_{ij} = \begin{cases} (\alpha + 1) |T_i| + \lambda |\partial T_i| & \text{if } i = j \\ -\lambda \ell_E & \text{if } i \neq j \text{ and } T_i \cap T_j = E \\ 0 & \text{otherwise}. \end{cases}$$

Here $|T_i|$ denotes the area of triangle $T_i$ and $|\partial T_i|$ is the length of its perimeter. In the case of an inpainting problem, $\alpha + 1$ is replaced by $\alpha$ on triangles in the inpainting region $S_0$.

This manuscript is for review purposes only.
Owing to the structure of (27), the minimization w.r.t. the variable $d$ is given explicitly by the coefficient-wise formula

$$d_E = \text{shrink} \left( \|u\|_E + b_E, \frac{\beta}{\lambda} \right).$$

Finally, the update step for $b$ is given coefficient-wise by $b_E := b_E + \|u\|_E - d_E$.

It is worth noting that in the implementation of the split Bregman algorithm in FEniCS we can take advantage of the availability of a finite element space consisting of piecewise constants on the skeleton of the mesh. The variables $b$ and $d$ are discretized in this space. Therefore the linear system representing the optimality conditions for the minimization of (27) w.r.t. $u$ can be assembled using the UFL form language. The minimization of (27) w.r.t. $d$ is carried out by applying the shrinkage operator to the coefficient vector of $d$. Similarly, the update of $b$ is also done coefficient-wise.

To achieve a fair comparison, we consider an appropriate discrete formulation of the barrier term in (22). For $u \in \mathcal{DG}_0$ it is easy to see that

$$|u|_{TV}(S) = \sup \left\{ \int_S u \, \text{div} \, p \, ds : p \in \mathcal{RT}_1, \, |p \cdot n_E| \leq 1 \text{ for all edges } E \right\}$$

holds. Recall that $p \cdot n_E$ are exactly the degrees of freedom in the Raviart–Thomas space $\mathcal{RT}_1$, where $n_E$ is the edge normal with an arbitrary orientation. Consequently, we discretize the logarithmic barrier term in (22) by

$$-\mu \sum_E \left[ \ln (\beta + p \cdot n_E) + \ln (\beta - p \cdot n_E) \right] \ell_E.$$

Figure 8 shows a comparison of our interior point approach with the split Bregman iteration, applied to a gray-scale version of the shoe denoising example above with $\alpha = 0$ and $\beta = 0.1$. This problem has 100,000 degrees of freedom for the unknown $u$ and 150,000 for each of $d$ and $b$. For the split Bregman method we used the hand-tuned penalty parameter $\lambda = 1$. Although normally unnecessary, we calculated the image $u$ via (25) in each iteration of the interior point approach so that we can compare the values of the primal objective (1) achieved by both methods in each iteration. In this example with lowest order discretization, we found that we were able to reduce the barrier parameter $\mu$ more dynamically than proposed in Algorithm 1. We used $\mu_{\text{start}} = 1.0$, a fixed reduction factor of $d_\mu = 0.05$ and a final value $\mu_{\text{end}} = 6.25 \times 10^{-6}$. The number of Newton steps per value of $\mu$ are in the range 4 to 14.

As expected, one Newton step in our interior point iteration requires more time than one step of split Bregman. This is due to the larger size of the linear system as well as the assembly of the terms pertaining to the logarithmic barrier term and its derivatives. Nevertheless, we find that the proposed interior point method shows a similar performance as split Bregman in this large-scale example, both w.r.t. time-to-solution and objective value achieved. Our method converges after 5 outer iterations or 156 s to an objective value of 349.80, whereas the split Bregman method achieves the same objective value after 71 iterations or 109 s.
7. Conclusion and Outlook. We considered an analog of the TV–$L^2$ image reconstruction approach for images on smooth surfaces. Complementary to [39], we proved the well-posedness of the model and its predual, and rigorously established strong duality with the predual in function space. The predual problem is a quadratic optimization problem for the vector field $p \in H(\text{div}; S)$ with pointwise nonlinear inequality constraints on the surface. As in the flat case, $p$ serves as an edge detector. We proposed and analyzed a function space interior point method for the predual problem. Based on the finding that the latter is posed in $H(\text{div}; S)$, we are led to choose a conforming finite element discretization by the surface analog of first- or higher-order Raviart–Thomas finite element spaces. In contrast to linear Lagrangian elements employed in [39], our discretization exhausts the space when the surface mesh is refined. Numerical examples, which comprise denoising and inpainting problems, show the viability of the approach for real-world geometries consisting of more than $350\,000$ and $175\,000$ vertices. Our method can be easily adapted to surfaces with boundary, by replacing $H(\text{div}; S)$ with $H_0(\text{div}; S)$. This amounts to imposing the boundary condition $p \cdot n = 0$ along the boundary $\partial S$, where $n$ is the outer unit normal vector in the tangent plane. The analysis presented carries over with minor changes.

There is room for improvement in various directions. For instance, the polynomial order $r$ of the finite element space $DG_r$ for the image data $f$ could be adjusted locally to reflect the level of detail present in each surface cell. This would then naturally lead to discretizations of $p$ and $u$ with varying polynomial degree as well. Moreover, we have so far been solving the predual problem with a basic primal interior point approach, running Newton’s method to convergence for each value of the barrier parameter $\mu$. A more sophisticated primal-dual interior-point method with inexact system solves might help reduce the computational cost for high-dimensional problems. While we are exploiting the MPI-based parallelism of the FEniCS

This manuscript is for review purposes only.
library for system assembly and direct system solves already, more efficiency might be gained by preconditioned iterative solvers with tailored preconditioners. This appears particularly promising in order to treat also problems with non-local operators \( K \) efficiently. This is left for future research.

**Appendix A. Proof of Theorem 12 and Auxiliary Results.** Let us denote by

\[
\ell(p) := \begin{cases} 
-\mu \ln (\beta^2 - |p|_2^2) & \text{if } |p|_2 < \beta \\
\infty & \text{otherwise}
\end{cases}
\]

and \( \nabla \ell(p) = 2 \mu \frac{p}{\beta^2 - |p|_2^2} \) if \( |p|_2 < \beta \) the pointwise barrier term and its gradient\(^4\), defined for \( p \in \mathbb{R}^3 \) and in particular for \( p \) in the tangent space \( T_p(S) \) of the surface \( S \) at some point. Moreover, let

\[
b(p) := \int_S \ell(p) \, ds \quad \text{and} \quad \langle b'(p), \delta p \rangle := \int_S \langle \nabla \ell(p), \delta p \rangle_2 \, ds
\]

denote the integrated barrier term (cf. (23)) and its formal derivative for vector fields \( p, \delta p \in L^2(S; T(S)) \). Let us recall from the proof of Proposition 11 that \( b \) is convex and it can take values in \( \mathbb{R} \cup \{\infty\} \). We denote by \( \partial b(p) \subset L^2(S; T(S))^* \) the convex subdifferential of \( b \) at \( p \).

Notice that \( L^2(S; T(S))^* \) can be identified with \( L^2(S; T(S))^* \) and also with \( L^2(S; T(S)) \).

Before stating the proof of Theorem 12 we require some preliminary results. The following lemma parallels [52, Lemma 4.4] and its proof is therefore omitted.

**Lemma 13.** Consider \( p, \delta p \in L^2(S; T(S)) \) such that all of \( b(p), b(p + \delta p) \) and \( \langle b'(p), \delta p \rangle \) are finite. Then \( b \) is directionally differentiable at \( p \) in the direction \( \delta p \), and its directional derivative satisfies

\[
\langle b'(p), \delta p \rangle = \langle b'(p), \delta p \rangle \geq \int_S \langle m, \delta p \rangle_2 \, ds \quad \text{for all } m \in \partial b(p),
\]

where the subdifferential is considered a subset of \( L^2(S; T(S)) \).

The next result is equal to [52, Prop. 4.5] but the proof requires a number of modifications.

**Proposition 14.** Let \( p \in L^2(S; T(S)) \) be given. Then we have:

(i) If \( \nabla \ell(p) \) belongs to \( L^2(S; T(S)) \), then \( \partial b(p) = \{ \nabla \ell(p) \} \).

(ii) If \( \nabla \ell(p) \) does not belong to \( L^2(S; T(S)) \), then \( \partial b(p) = \emptyset \).

**Proof.** The proof is split into three parts, which combine to yield the result.

Part A: We begin by considering the case \( b(p) < \infty \), which implies \( |p|_2 < \beta \) a.e. on \( S \). By convexity of \( \ell \), we obtain \( \langle \nabla \ell(p), \delta p \rangle_2 \leq \ell(p + \delta p) - \ell(p) \) a.e. and therefore

\[
\int_S \langle \nabla \ell(p), \delta p \rangle_2 \, ds = \langle b'(p), \delta p \rangle \leq b(p + \delta p) - b(p)
\]

\(^4\)This should not be confused with the gradient of a scalar function on \( S \) in Definition 4. In the present context the gradient \( \nabla \ell(p) \) is the transpose of the derivative of the function \( \ell : \mathbb{R}^3 \to \mathbb{R} \).
for all \( \delta p \in L^2(S; T(S)) \), provided that \( \nabla \ell(p) \in L^2(S; T(S)) \) holds. This shows \( \nabla \ell(p) \in \partial b(p) \) in this case.

Part B: Now suppose that \( m \in \partial b(p) \) holds and let \( M \subset S \) be an arbitrary measurable subset and \( v : S \to T(S) \) be a vector field of class \( C^0 \). Due to the compactness of \( S \), \( \|v\|_{L^\infty(S; T(S))} \) is finite. We are going to show that necessarily

\[
\int_M (\nabla \ell(p), v)_2 \, ds = \int_M (m, v)_2 \, ds
\]

holds, which then implies \( m = \nabla \ell(p) \) and \( \nabla \ell(p) \in L^2(S; T(S)) \). To this end, define \( M_\delta := \{ p \in E : \beta - |p|_2 > \delta \} \) for \( \delta > 0 \). Next we define \( v_k := \chi_{M_{1/k}} v \) and \( \varepsilon_k := (2k \|v\|_{L^\infty(S; T(S))})^{-1} \) for \( k \in \mathbb{N} \). Then, since \( \nabla \ell(p) = 2 \mu \frac{p}{\beta^2 - |p|_2^2} \) belongs to \( L^\infty(M_\delta; T(S)) \) for any \( \delta > 0 \), we have

\[
|\langle b'(p), \pm \varepsilon_k v_k \rangle| = \pm 2 \mu \varepsilon_k \int_{M_{1/k}} \frac{(p, v_k)_2}{\beta^2 - |p|_2^2} \, ds < \infty.
\]

Moreover,

\[
b(p \pm \varepsilon_k v_k) = -\mu \int_{S \setminus M_{1/k}} \ln(\beta^2 - |p|_2^2) \, ds - \mu \int_{M_{1/k}} \ln(\beta^2 - |p \pm \varepsilon_k v_k|_2^2) \, ds.
\]

The first integral is finite since \( b(p) \) is. For the second integral, we use that

\[
\beta - |p \pm \varepsilon_k v_k|_2 \geq \beta - |p|_2 - \varepsilon_k \|v\|_{L^\infty(S; T(S))} \geq \frac{1}{k} - \frac{1}{2k} = \frac{1}{2k}
\]

holds a.e. on \( M_{1/k} \). Hence by multiplication with \( \beta + |p \pm \varepsilon_k v_k|_2 \geq \beta \) we conclude

\[
\beta^2 - |p \pm \varepsilon_k v_k|_2 \geq \frac{\beta}{2k} \quad \text{a.e. on } M_{1/k}
\]

and thus the second integral in (**) is finite as well. So we have shown that for \( \delta p = \pm \varepsilon_k v_k \), the terms \( b(p), b(p + \delta p) \) and \( \langle b'(p), \delta p \rangle \) are all finite. Hence Lemma 13 yields

\[
\langle b'(p), \pm \varepsilon_k v_k \rangle \geq \pm \varepsilon_k \int_S (m, v_k)_2 \, ds \quad \text{for all } m \in \partial b(p).
\]

This implies

\[
\int_S (\nabla \ell(p), v_k)_2 \, ds = \langle b'(p), v_k \rangle = \int_S (m, v_k)_2 \, ds \quad \text{for all } m \in \partial b(p), k \in \mathbb{N}.
\]

It remains to pass to the limit in (***) to show (*). Let us begin with the second term in (***), and observe that \( \chi_M v = \lim_{k \to \infty} v_k \) holds a.e. on \( S \) since the set where \( |p|_2 = \beta \) holds has zero measure. Moreover, the integrand is dominated pointwise by \( |(m, v_k)_2| \leq |m|_2 \|v\|_{L^\infty(S; T(S))} \in L^2(S) \). Thus by Lebesgue’s dominated convergence theorem we obtain

\[
\lim_{k \to \infty} \int_S (m, v_k)_2 \, ds = \int_M (m, v)_2 \, ds.
\]
We now address the first term in \((***)\). Since it is not clear whether or not \(\nabla \ell(p)\) belongs to \(L^2(S; T(S))\), we cannot argue by dominated convergence. Instead, let us define

\[
S^+ := \{ p \in S : (p, v)_2 \geq 0 \} \quad \text{and} \quad S^- := S \setminus S^+.
\]

Then

\[
\chi_{S^+}(\nabla \ell(p), v)_2 = \begin{cases} 2 \mu \frac{(p, v)_2}{\beta^2 - |p|^2} \geq 0 & \text{on } S^+ \cap M_{1/k} \\ 0 & \text{elsewhere} \end{cases}
\]

and therefore \(\{\chi_{S^+}(\nabla \ell(p), v)_k\}_k\) is non-negative and monotone increasing on \(S^+\) with pointwise limit \(\chi_{S^+ \cap M}(\nabla \ell(p), v)_2\). By the monotone convergence theorem,

\[
\lim_{k \to \infty} \int_{S^+} (\nabla \ell(p), v)_2 \, ds = \int_{S^+ \cap M} (\nabla \ell(p), v)_2 \, ds
\]

holds. Similarly, this result can be shown with \(S^−\) as well. We can therefore pass to the limit in \((***)\) and conclude (*), which in turn proves \(m = \nabla \ell(p)\) as well as \(\nabla \ell(p) \in L^2(S; T(S))\).

Part C: If \(b(p) = \infty\), then by definition \(\partial b(p) = \emptyset\) holds.

The result now follows easily by combining Parts A–C.

So far we have considered the subdifferential of the barrier term \(b\) w.r.t. the \(L^2(S; T(S))\) topology. This is however not sufficient since problem (22) is posed in \(H(\text{div}; S)\) and further modifications of the arguments in [52] are required. Let us define by \(\tilde{b}\) the restriction of \(b\) to \(H(\text{div}; S)\), and let

\[
(29) \quad \partial \tilde{b}(p) := \left\{ \tilde{m} \in H(\text{div}; S) : \tilde{b}(q) \geq \tilde{b}(p) + (\tilde{m}, q - p)_{H(\text{div}; S)} \quad \text{for all } q \in H(\text{div}; S) \right\}
\]

denote the subdifferential of \(\tilde{b}\) at \(p \in H(\text{div}; S)\). Finally, \(\Pi : L^2(S; T(S)) \to H(\text{div}; S)\) denotes the \(H(\text{div}; S)\)-orthogonal projector, defined by

\[
\tilde{m} = \Pi \ m \iff (\tilde{m}, z)_{H(\text{div}; S)} = (m, z)_{L^2(S; T(S))} \quad \text{for all } z \in H(\text{div}; S).
\]

**Corollary 15.** Let \(p \in H(\text{div}; S)\) be given. Then we have \(\partial \tilde{b}(p) = \Pi \partial b(p)\) and consequently:

\((i)\) If \(\nabla \ell(p)\) belongs to \(L^2(S; T(S))\), then \(\partial \tilde{b}(p) = \{\Pi \nabla \ell(p)\}\).

\((ii)\) If \(\nabla \ell(p)\) does not belong to \(L^2(S; T(S))\), then \(\partial \tilde{b}(p) = \emptyset\).

**Proof.** Let \(\Lambda : H(\text{div}; S) \to L^2(S; T(S))\) denote the continuous embedding, and let \(\Lambda^* : L^2(S; T(S))^* \to H(\text{div}; S)^*\) denote its adjoint. Since by definition \(\tilde{b}(p) = b(\Lambda p)\) and \(\Lambda p = p\) holds for all \(p \in H(\text{div}; S)\), we conclude from the chain rule that

\[
\mathcal{R}_{H(\text{div}; S)} \partial \tilde{b}(p) = \Lambda^* \mathcal{R}_{L^2(S; T(S))} \partial b(\Lambda p) = \Lambda^* \mathcal{R}_{L^2(S; T(S))} \partial b(p)
\]
holds; see for instance [27, Prop. I.5.7]. Notice that the Riesz maps $\mathcal{R}_{H(\text{div}; S)} : H(\text{div}; S) \to H(\text{div}; S)^*$ and $\mathcal{R}_{L^2(S; T(S))} : L^2(S; T(S)) \to L^2(S; T(S))^*$ are present here since we identify the subdifferential in both Hilbert spaces with elements from the Hilbert space itself; cf. (29).

We have thus shown that

$$\tilde{\partial} b(p) = \mathcal{R}_{H(\text{div}; S)}^{-1} \Lambda^* \mathcal{R}_{L^2(S; T(S))} \partial b(p)$$

holds for all $p \in H(\text{div}; S)$. It is now an easy exercise to verify that $\mathcal{R}_{H(\text{div}; S)}^{-1} \Lambda^* \mathcal{R}_{L^2(S; T(S))}$ is equal to $\Pi$.

**Proof of Theorem 12.** Recall from (23) the definition of $H$ and $b$ and let us denote, as above, by $\tilde{b}$ the restriction of $b$ to $H(\text{div}; S)$. The (unique) minimizer of (22) is characterized by

$$0 \in \partial (H(p) + \tilde{b}(p)).$$

The function $H$ is convex and continuous on all of $H(\text{div}; S)$. Moreover, recall from the proof of Proposition 11 that $b$ is convex and weakly sequentially lower semicontinuous w.r.t. $L^2(S; T(S))$, and thus $\tilde{b}$ has the same property w.r.t. $H(\text{div}; S)$. In addition, $\tilde{b}$ is finite e.g., at $p \equiv 0$. Therefore (*') is equivalent to

$$0 \in \partial H(p) + \tilde{b}(p)$$

by the sum rule of subdifferentials; see for instance [27, Prop. I.5.6]. Notice that this also implies $|p|_2 \leq \beta$ a.e. on $S$ since otherwise $\tilde{b}(p) = \infty$ and the subdifferential is empty. Having characterized $\partial \tilde{b}(p)$ in Corollary 15 and using the obvious Fréchet differentiability of $H$, we can write equivalently (using the notation from Corollary 15)

$$0 = \mathcal{R}_{H(\text{div}; S)}^{-1} H'(p) + \Pi \nabla \ell(p)$$

$$\iff 0 = \mathcal{R}_{H(\text{div}; S)}^{-1} H'(p) + \mathcal{R}_{H(\text{div}; S)}^{-1} \Lambda^* \mathcal{R}_{L^2(S; T(S))} \nabla \ell(p)$$

$$\iff 0 = H'(p) \delta p + (\Lambda^* \mathcal{R}_{L^2(S; T(S))} \nabla \ell(p), \delta p)_{H(\text{div}; S), H(\text{div}; S)^*} \quad \text{for all } \delta p \in H(\text{div}; S)$$

$$\iff 0 = H'(p) \delta p + (\mathcal{R}_{L^2(S; T(S))} \nabla \ell(p), \Lambda \delta p)_{L^2(S; T(S)), L^2(S; T(S))} \quad \text{for all } \delta p \in H(\text{div}; S)$$

$$\iff 0 = H'(p) \delta p + (\nabla \ell(p), \Lambda \delta p)_{L^2(S; T(S))} \quad \text{for all } \delta p \in H(\text{div}; S).$$

This is precisely (24).

**Acknowledgments.** The authors would like to thank the three anonymous reviewers for their constructive comments, which helped improve the presentation of the paper. Parts of this paper were written while the second author was visiting the University of British Columbia, Vancouver. He would like to thank the Department of Computer Science for their hospitality.

---

**REFERENCES**


This manuscript is for review purposes only.


WEAK AND STRONG FORM SHAPE Hessians AND THEIR AUTOMATIC GENERATION

STEPHAN SCHMIDT∗§

Abstract. By analyzing variational problems formulated in the Unified Form Language, a structure-aware differentiation tool is presented, which can automatically generate both the classical boundary representation and the weak or “volume” formulation of first and second order shape derivatives. Where applicable, the tool can either automatically apply the divergence theorem in tangent spaces for the strong form or calculate discrete material derivatives for the weak form. Furthermore, additional assumptions and simplifications can also be automatically applied, such that a repeated application leads to symmetric shape Hessians. The resulting expression can then be processed by the FEniCS environment, resulting in the semi-automatic creation of shape optimization chains from a user-supplied Lagrangian only. The methodology is tested by conducting shape Newton optimization using examples from geometry and CFD. The respective software is released as open source at https://bitbucket.org/Epoxid/femorph.

Key words. Shape Optimization, Shape Derivatives, Shape Hessians, Code Generation

AMS subject classifications. 68T35, 68Q42, 68U05, 68U15, 68U20, 65D18, 49M15

1. Introduction. Scientific computing plays an ever increasing role in understanding complex physical processes and computer simulations are nowadays an indispensable tool in many fields and applications, including engineering, physics, medicine, climate studies and many more. Recent trends with respect to creating and maintaining high fidelity simulation codes for partial differential equations (PDEs) governing such complex problems are high level languages and automatic code generation. Being able to postulate a problem in a programming paradigm close to the mathematical problem naturally makes understanding, developing and maintaining complex codes much easier. In this setting, any modeling change of the mathematical problem ideally carries over into the simulation code immediately. Also, generating the simulation software automatically from a near mathematical source can potentially lead to independence of the underlying hardware from the user’s perspective, especially with regard to generating scalable solvers for distributed and heterogeneously parallel CPU/GPU/FPGA systems. Examples of these developments are the “Distributed and Unified Numerics Environment (DUNE)” [5] and especially the FEniCS project [20] and Firedrake [25]. These frameworks strike out in particular, because they can process the Unified Form Language (UFL) [3] as an input1, which means they generate simulation software based on sources that are particularly close to the variational problem. Due to its closeness to UFL and its code generation capabilities, the FEniCS project forms an excellent basis for the work presented herein, that is the making the transition from generating simulation software to automatically creating higher order optimization software for problems in computational geometry and PDE constrained shape optimization based on both the classical Hadamard-form as well as the weak or “volume” form of first and second order shape derivatives.

In general, there are two strategies to facilitate this step: optimize-then-discretize and discretize-then-optimize. With respect to shape optimization, the gap between the two is particularly large as discussed in [4, 21]. With respect to the first approach,
the problem is analytically differentiated and afterwards the optimization problem is discretized and solved. The advantages of this approach are mathematical insights of the problem at hand and mesh independent convergence. Also, a comparatively large degree of structure exploitation is possible when creating the optimization code from scratch. The downsides are a discrepancy between the treatment of an infinite dimensional problem versus the finite dimensional problem post discretization. Typical examples where this becomes apparent are hyperbolic conservation laws, where the continuous formulation often does not consider the numerical stabilization terms by e.g. upwindfluxes and limiters. Another example would be iterative solvers for steady state problems and then disregarding the iterative solution process when deriving the adjoint model. With respect to shape optimization, this means that given sufficient problem regularity, the divergence theorem can be used to arrive at a boundary representation of the derivative. However, typical Galerkin-type discretizations usually do not offer the required regularity [4, 21]. Nevertheless, this approach has been successfully applied to many large scale problems [27, 30].

The latter approach, \textit{discretize-then-optimize}, typically means using automatic or algorithmic differentiation (AD) to create a solver based on the finite dimensional computer program [14, 16]. Mesh independent convergence can be achieved if the continuous model is taken into account. However, the main advantage of this approach is guaranteed correctness of the derivative information for the discretized model. Thus, this methodology is successfully applied even in cases where the continuous differentiability of the forward problem is not given, a situation often encountered in modeling of turbulent flows [7]. Furthermore, the discrete adjoint will inherit the contractive properties of the primal code [1, 8]. However, to create an efficient adjoint solver, several transformations are necessary in addition to execution reversal. Some of these are checkpointing strategies [15, 19, 35] and “reverse accumulation” [8], which can be interpreted as an early example of a hybridization strategy between the discrete and continuous approach by re-introducing the continuous assumption of having solved the PDE exactly without any iterative scheme. These steps almost always require human understanding of the sources, because AD tools typically process code on a “per instruction” level, without having insights into the problem being solved or on the meaning and purpose of variables or loops. Due to this reason, generating shape derivatives automatically has been a challenging task so far, requiring considerable human intervention. In particular the mesh deformation procedure must be highly integrated into the simulation code, because treating solution procedure, objective function evaluation and mesh deformation separately requires storing very large dense Jacobian matrices as intermediate values [13].

1.1. Purpose of the S-AD Software. With the advent of high level domain specific programming languages such as UFL, it nowadays becomes feasible to create computer programs, where the differentiation tool has detailed information available about the mathematical problem being solved. A forerunner of the advantages such a hybrid approach offers can for example be seen in the dolfin-adjoint framework for transient problems [11]. The purpose of the present work is to demonstrate the effectiveness, applicability and user-friendliness of hybridization strategies by developing a Shape-AD Tool (S-AD), which automatically produces both classical and weak shape derivatives of first and higher order.

The Iso-Perimeter Problem or “Dido’s Problem” to find the object $\Omega$ of a given
volume $\text{Vol}_0$ with minimum surface area will serve as a running example

$$\min_{\Omega} J(\Omega) := \int_{\partial\Omega} 1 \, ds$$

subject to

$$\int_{\Omega} 1 \, dx = \text{Vol}_0.$$  

For $\lambda \in \mathbb{R}$, the Lagrangian of this problem is given by

$$(2) \quad \mathcal{L}(\Omega, \lambda) := \int_{\partial\Omega} 1 \, ds + \lambda \left( \int_{\Omega} 1 \, dx - \text{Vol}_0 \right).$$

The necessary optimality condition is that the derivative of the Lagrangian vanishes with respect to arbitrary perturbations $(V, \tilde{\lambda})$ of the domain $\Omega$ and the adjoint variable $\lambda$. Through a variety of transformation steps discussed in Section 2, these partial derivatives are given by

$$(3) \quad d\mathcal{L}(\Omega, \lambda)[V] = \int_{\partial\Omega} (V, n) (\kappa + \lambda) \, ds$$

$$(4) \quad \langle \nabla_\lambda \mathcal{L}(\Omega, \lambda), \tilde{\lambda} \rangle := \left. \frac{d}{d\varepsilon} \right|_{\varepsilon=0} \mathcal{L}(\Omega, \lambda + \varepsilon \tilde{\lambda}) = \tilde{\lambda} \left( \int_{\Omega} 1 \, dx - \text{Vol}_0 \right),$$

where $n$ is the outer normal and $\kappa$ is the mean curvature of $\partial\Omega$. The shape derivative (3) vanishes for arbitrary $V$ if the relation $\kappa(s) = -\lambda \forall s$ holds. Because $\lambda$ is a constant, curvature must not change over $\partial\Omega$. Furthermore, if the volume constraint is satisfied, then (4) also vanishes. These properties are fulfilled by a circle of volume $\text{Vol}_0$, which is thus a critical point of this optimization problem.

It is the intention of this work to discuss a computer program that can automatically conduct the shape differentiation procedure from (2) to (3) for a wide class of problems, including PDE constraints. The program accepts inputs and outputs of equations in the UFL language, which then seamlessly interfaces with the FEniCS code generation capabilities to generate the actual optimization software. To achieve these transformations, the differentiation tool has to exhibit considerable knowledge of the primal problem, as certain variables such as normal and curvature require special treatments and the boundary representation is only achievable when the divergence theorem, or even its counterpart in tangent spaces, is applied. Generating the weak form requires distinguishing between spatial, Gâteaux and material derivatives. Because material derivatives and spatial derivatives do not commute, the automatic generation of material derivative adjoints would be quite difficult for classical AD software, as spatial derivatives would have to be recognized from per-instruction analysis of, e.g., differentiation stencils. To the best of the author’s knowledge, this is the first instance where any AD-like software automatically applies such integral and derivative transformations. Consequently, any code generation framework that accepts UFL input, such as FEniCS, DUNE or Firedrake, is automatically transformed into a framework for solving problems in computational geometry, PDE constrained shape optimization and image processing with an exceptional user-friendliness and
human readable derivative output. One downside of hybridizing towards the continuous problem is of course the re-introduction of consistency and differentiability questions, but can be mitigated by generating the weak or “volume” shape derivative instead. It is also worth noting that the repeated application of strong form shape differentiation results in non-symmetric Hessian expressions, for which one remedy is the computation of certain symmetric Riesz-representations \cite{32} or assuming certain Lie-bracket criteria are fulfilled \cite{9}, the latter will be built into codes based on the strong form automatically if the user indicates doing so. With respect to weak form shape derivatives, the symmetry question is related to commuting the differentiation directions of material derivatives.

This work is structured as follows: Chapter 2 offers a brief summary on continuous shape optimization and extends those concepts to weak and strong second order derivatives, especially for boundary objectives. Next, Chapter 3 discusses the Unified Form Language and the unique opportunities for structure exploitation while constructing the respective differentiation software as part of this work. Then, the actual S-AD procedure is presented in Chapter 4 with special emphasis on expression simplification for shape Hessians via repeated application. Finally, Chapter 5 demonstrates the applicability of the methodology by automatically generating solvers for first- and second order optimization schemes in computational geometry and fluid dynamics.

2. Shape Optimization.

2.1. First Order Calculus. In this section, the results for shape differentiation are summarized from \cite{9, 33}. Let $\Omega$ be the unknown domain to be optimized and let $\mathcal{D}$ be the hold-all domain, a domain to contain all of the iterates of the shape optimization problem, meaning $\Omega \subset \mathcal{D} \subset \mathbb{R}^d$. We follow the notion of “perturbation of identity”, meaning a deformed domain $\Omega(\epsilon)[V]$ is defined by

$$\Omega(\epsilon)[V] := \{x + \epsilon V(x) : x \in \Omega\},$$

where the vector field $V \in C^0_0(\mathcal{D}, \mathbb{R}^d)$. For sufficiently smooth families of functions $f : \mathcal{D} \times [0, \delta] \to \mathbb{R}, (x, \epsilon) \mapsto f(x, \epsilon) := f_\epsilon(x)$, we seek to differentiate both volume and surface integrals

$$J_1(\Omega(\epsilon)) := \int_{\Omega(\epsilon)[V]} f(x, \epsilon) \, dx, \quad J_2(\Omega(\epsilon)) := \int_{\Gamma(\epsilon)[V]} f(s, \epsilon) \, ds.$$

Here, $\Gamma(\epsilon) \subset \partial \Omega(\epsilon)$ denotes parts of the boundary of $\Omega(\epsilon)$. In the following, we use $\Gamma$ synonymously with $\partial \Omega$, because $V$ can be considered constant zero on fixed parts of $\partial \Omega$.

The explicit dependency of $f$ on the deformation parameter $\epsilon$ arises quite naturally. For example, the normal $n$ or the curvature $\kappa$ depend on both the evaluation point $x \in \mathcal{D}$ and the current shape $\Omega(\epsilon)$. Also, $f$ can be the solution of a partial differential equation and its numerical value again depends on the evaluation point $x$ and the actual shape $\epsilon$ of the domain the PDE is defined in. To process these two effects in detail, two different derivative concepts are usually introduced: On the one hand, the material derivative, given by

$$\dot{f}(x)[V] := \lim_{\epsilon \to 0^+} \frac{f(x + \epsilon V(x), \epsilon) - f(x, 0)}{\epsilon} \quad (5)$$

can be seen as the “total derivative” of all dependencies on $\epsilon$, meaning this derivative concept takes into account both the shift in the evaluation point and a change due to
a direct dependency on the geometry as discussed above. On the other hand, the local or shape derivative is used to process only the change of \( f \) due to its direct dependency on the geometry while the evaluation point is kept constant during the deformation of the domain. A direct definition of this object can be somewhat problematic, as the evaluation point \( x \) can be outside of \( \Omega(\epsilon) \) for \( \epsilon > 0 \). For this reason, the shape derivative is usually defined by

\[
f'(x)[V] := f[V](x) - Df(x,0) \cdot V(x),
\]

which means the inner part stemming from the chain rule and the motion of the evaluation point is removed. For this reason, the local derivative can also be interpreted as a partial derivative with respect to the direct geometric dependency only. Here, \( D \) and \( \nabla \) is used to denote the Jacobian or gradient with respect to the spatial variable.

For brevity, the notation \( f(x) := f(x,0) \) will be used in the following, likewise for \( \Omega(0) \) and \( \Gamma(0) \). A problem is called shape-differentiable at \( \Omega \) in direction \( V \), if the limit

\[
dJ(\Omega)[V] = \lim_{\epsilon \to 0^+} \frac{J(\Omega(\epsilon)[V]) - J(\Omega)}{\epsilon}
\]

exists and is linear in \( V \). The quantity \( dJ(\Omega)[V] \) is then called the shape derivative of \( J \) in direction \( V \) at \( \Omega \) or the Eulerian semi-derivative. Using a straightforward approach of change of variables, the volume case \( J_1 \) can very conveniently be expressed by

\[
dJ_1(\Omega)[V] = \int_{\Omega} \text{div}(f(x)V(x)) \, dx + \int_{\Omega} f'(x)[V] \, dx
\]

\[
= \int_{\Omega} f(x) \cdot \text{div} V(x) + f'[x][V] \, dx
\]

\[
= \int_{\Gamma} \langle V, n \rangle f \, ds + \int_{\Omega} f'[V] \, dx,
\]

where \( \langle .., \rangle \) denotes the inner product. Formula (7) is referred to as the weak form of the shape derivative as the existence of the material derivative \( f'[V] \) is necessary only, whereas the strong or boundary formulation of formula (8) requires both the existence of the local derivative \( f'[V] \) and the boundary trace of \( f \). The latter is found after application of the divergence theorem. Thus, on a formal level, the difference of the strong and weak form of the shape derivative is just a different grouping of the terms in the div-expression and the local derivative \( f'[V] \), paired with an application of the divergence theorem. Their true differences only become apparent post finite element discretization: The weak form requires the numerical evaluation of the material derivative and the material derivative of finite element functions remain in the same finite element space. Contrarily, the strong form requires the computation of the local derivative and the local derivative of a finite element function typically is not in the same finite element space. For more details see [4].

For boundary integrals \( J_2 \), the process is similar but considerably more delicate.
Following the same domain deformation strategy as above, one arrives at the expressions

\[ dJ_2(\Omega)[V] = \int_\Gamma \langle \nabla f, V \rangle + f \text{div}_\Gamma V \, ds + \int_\Gamma f' [V] \, ds \]

(9)

\[ = \int_\Gamma f \text{div}_\Gamma V + f' [V] \, ds, \]

(10)

where formula (10) is again referred to as the weak form of the shape derivative. Furthermore, \( \text{div}_\Gamma \) denotes the tangential divergence, for which the extrinsic definition

\[ \text{div}_\Gamma f := \text{div} f - \langle Df \cdot n, n \rangle \]

can be used here. Application of the divergence theorem in tangent spaces to formula (9) leads to

\[ dJ_2(\Omega)[V] = \int_\Gamma \langle V, n \rangle \left[ (\nabla f, n') + \kappa f \right] \, ds + \int_\Gamma f'(s) [V] \, ds \]

(11)

\[ = \int_\Gamma \langle V, n \rangle \left[ \frac{\partial f}{\partial n} + \kappa f \right] \, ds + \int_\Gamma f'(s) [V] \, ds, \]

(12)

where \( \kappa := \text{div}_\Gamma n \) is the curvature of \( \Gamma \). Seeing as this requires the normal derivative of \( f \), formula (12) is again denoted the strong form of the shape derivative. Being dependent on a form of the divergence theorem, the transformation from equation (6) to (8) and from equation (9) to (12) is the first application of a divergence-like theorem for any AD tool, to the best of our knowledge.

2.2. Return to the Dido Problem. We can now briefly discuss how the two partial derivatives, equations (3) and (4), are created from the Lagrangian of the Dido Problem, equation (2). We can identify \( f \equiv 1 \) in the boundary integral and \( f \equiv \lambda \) in the volume integral. Both are interpreted as spatially constant functions independent of the deformation \( \epsilon \), so the gradient terms and the local shape derivatives in formulas (8) and (12) vanish. Also, the constant \( \text{Vol}_0 \) does not depend on \( \epsilon \) and vanishes, immediately leading to expression (3). This example also shows that including objectives depending on normal \( n \) and curvature \( \kappa \) suggests itself, as these naturally appear when higher order derivatives are desired. For this reason, we consider such geometric dependencies next.

2.3. Dependencies on the Geometry. With the intention of constructing an AD-like tool, general objective functions need to be able to be processed, in particular those, where \( f \) depends on the geometry itself. To this end, we restrict ourselves to two cases: normal and curvature dependency. Because \( f \) is required to exist in the hold-all \( D \), both \( n \) and \( \kappa \) are thought of as (normalized) extensions of the normal and curvature into a tubular neighborhood of \( \Gamma \). For simplicity, we consider these extensions to be constant in normal direction, that is \( Dn \cdot n = 0 \) and \( D\kappa \cdot n = 0 \). With respect to the normal dependency, application of the chain rule to a function results in

\[ f(s, \epsilon) := g(s, n_\epsilon(s_\epsilon)) \Rightarrow f'(s)[V] = \langle \nabla_2 g(s, n(s)), \dot{n}(s)[V] \rangle, \]

(13)

where \( \nabla_2 \) denotes the partial derivative with respect to the second argument and \( \dot{n}[V] \) is the material derivative of the normal, which is given by

\[ \dot{n}[V] = -(D_\Gamma V)^T \cdot n, \]

(14)
where \( D_V := DV - DV_{nn}^T \) is the tangential Jacobian of \( V \). Because

\[
1 = \langle n_\epsilon(x_\epsilon), n_\epsilon(x_\epsilon) \rangle \quad \Rightarrow \quad 0 = \langle \hat{n}[V], n \rangle,
\]

it is easy to see that if \( V \) is a perturbation in the normal direction only, the above equation (14) can be transformed to

\[
\dot{n}[V] = -\nabla_\Gamma \langle V, n \rangle,
\]

where \( \nabla_\Gamma g := \nabla g - \langle \nabla g, n \rangle n \) is the tangential gradient. The tangential Stokes formula states for a vector \( v \), a scalar quantity \( a \) and a closed surface \( \partial \Omega \) all of sufficient regularity

\[
\int_{\partial \Omega} \langle v, \nabla_\Gamma a \rangle \, ds = \begin{cases} 0, & \text{if } v \text{ is normal} \\ \int_{\partial \Omega} -a \cdot (\text{div}_\Gamma v - \kappa \langle v, n \rangle) \, ds, & \text{otherwise}. \end{cases}
\]

Thus, in the situation with normal dependency, equation (13), the tangential Stokes formula can be applied and one arrives at

\[
dJ_2(\Omega)[V] = \int_{\Gamma} \langle V, n \rangle \left[ (\nabla g, n) + \kappa g \right] \, ds + \int_{\Gamma} \langle \nabla_2 g, \dot{n}[V] \rangle \, ds
\]

\[
= \int_{\Gamma} \langle V, n \rangle \left[ \frac{\partial g}{\partial n} + \kappa (g - \langle \nabla_2 g, n \rangle) + \text{div}_\Gamma \partial_2 g \right] \, ds.
\]

If the normal \( n \) enters \( g \) linearly in equation (13), then the above can be simplified to the directional shape derivative for expressions with linear dependency on the normal

\[
dJ_2(\Omega)[V] = \int_{\Gamma} \langle V, n \rangle \, \text{div}_2 g \, ds.
\]

More details can be found in [26, 30, 34]. Thus, it becomes clear that the resulting code to be generated can be simplified considerably in case equation (14) is relevant or, alternatively, one of the cases of equation (19) or (20) can be used.

With respect to curvature, it is easy to see that the identity

\[
\dot{k}[V] = -\Delta_\Gamma \langle V, n \rangle
\]

holds [10, 17]. Hence, a curvature dependency in accordance to

\[
f(s, \epsilon) := h(s, \kappa_\epsilon(s_\epsilon)) \Rightarrow f'[V] = \partial_2 h \cdot \dot{k}[V]
\]

will modify the shape derivative of equation (12) to

\[
dJ_2(\Omega) = \int_{\Gamma} \langle V, n \rangle \left[ (\nabla h, n) + \kappa h \right] \, ds + \int_{\Gamma} \partial_2 h \cdot \dot{k}[V] \, ds.
\]

Using the tangential Stokes formula (17) again, one arrives at the identity

\[
\int_{\Gamma} \partial_2 h \cdot \dot{k}[V] \, ds = \int_{\Gamma} -\partial_2 h \cdot \Delta_\Gamma \langle V, n \rangle \, ds = \int_{\Gamma} \langle \nabla_\Gamma \partial_2 h, \nabla_\Gamma \langle V, n \rangle \rangle \, ds.
\]

This transformation is necessary to achieve symmetry in the context of repeated differentiation. Naturally, combinations of equations (13) and (22) can occur and the S-AD software is able to treat those by combining each of the rules above.
2.4. Weak and Strong Repeated Differentiation Shape Hessians. When constructing differentiation software, one of the benefits of an algorithmic approach is that once first order directional derivatives can be created, any higher order directional derivatives can also be created by re-applying the transformation routines. In the context here, this means a natural concept for second order directional shape derivatives is the concept of repeated differentiation, i.e., the transformation schemes from Chapter 2 will be applied twice. For strong form second order shape derivatives, this does not lead to symmetric expressions and a related problem with respect to commuting directions in material derivatives arises when considering the weak form.

The normal enters equation (8) linearly. Either applying the differentiation rules of equation (20) to (8) or applying deformation (6) onto itself leads to a strong form second order directional derivative of the structure

\[ d^2 J_1(\Omega)[V,W] = \int_{\Gamma} \langle W, n \rangle \mathrm{div} \langle f V \rangle \, ds + \int_{\Gamma} \langle W, n \rangle f' \langle V \rangle \, ds + \int_{\Omega} f''[V,W] \, dx, \]

in which the two directions \( V \) and \( W \) do not commute. If one assumes that \( V \) and \( W \) are normal, i.e., \( V = \langle V, n \rangle n \), then one arrives at

\[ \langle W, n \rangle \mathrm{div} \langle f V \rangle = \langle W, n \rangle \langle \nabla f, n \rangle + f \langle (D V)^T W, n \rangle + f \langle W, n \rangle \langle V, n \rangle \kappa, \]

where one has also used that due to normalization, the identity \((D n)^T n = 0\) holds by the same reasoning as in equation (15). Thus, symmetry can be achieved if \( V \) and \( W \) are thought of as being independent of each other by assuming

\[ (DV)^T W = 0, \]

which is often called the Lie-Bracket condition in the literature [9]. Other approaches are the Riemannian view of [32]. Thus, one arrives at a symmetric strong form second order derivative

\[ d^2 J_1(\Omega)[V,W] = \int_{\partial \Omega} \langle W, n \rangle \langle V, n \rangle \left( \langle \nabla f, n \rangle + f \kappa \right) \, ds \]

\[ + \int_{\partial \Omega} \langle W, n \rangle f' \langle V \rangle + \langle V, n \rangle f' \langle W \rangle \, ds + \int_{\Omega} f''[V,W] \, dx. \]

With respect to the weak form of the shape derivative, the resulting higher order expression is more convenient. In particular, for autonomous vector fields \( V, W \) with \( V[W] = 0 \), that is again \((DV)^T W = 0\), a repeated application of formula (7) leads to

\[ d^2 J_1(\Omega)[V,W] = \int_{\Omega} f \mathrm{div} V \mathrm{div} W + f[V] \mathrm{div} W + f[W] \mathrm{div} V \]

\[ - f \mathrm{tr}(DV DW) + \tilde{f}[V,W] \, dx, \]

where the symmetric trace term arises due to material and spatial derivatives not commuting. In particular, one has for a differentiable field \( u \)

\[ (Du)[V] = D(\dot{u}[V]) - (DuDV) \]

\[ (\mathrm{div} u)[V] = (\mathrm{tr}(Du))[V] = \mathrm{div} (\dot{u}[V]) - \mathrm{tr}(DuDV). \]
Thus, the question of symmetry is completely related to \( f[V,W] \) as in all other terms \( V \) and \( W \) commute naturally. This second order material derivative will be discussed in more detail in Section 5.2.

Literature on Hessians of boundary objectives is very scarce. Indeed, higher order directional shape derivatives based on repeated differentiation of boundary objectives can quickly become very inconvenient expressions, for example when multiple non-linear dependencies on normals and curvature expressions are present. In this case, the objective cannot be easily transformed back into a volume integral via the divergence theorem and expressions like equations (19) and (23) with their dependency on tangential derivatives have to be processed again. Thus, we will focus on the most accessible case of equation (11) without explicit normal or curvature dependency in \( f \).

The following transformation and simplification rules will then enable processing the arbitrary case for the strong form automatically by the Shape-AD software.

With \( V \) normal, one arrives at

\[
\begin{align*}
\frac{d^2 J_2[\Omega]}{ds^2}[V,W] &= \int_{\Gamma} (W,n) \langle \nabla [(V,n) (\langle \nabla f, n \rangle + \kappa f)] , n \rangle \ ds + \int_{\Gamma} (W,n) \kappa (V,n) (\langle \nabla f, n \rangle + \kappa f) \ ds \\
&\quad + \int_{\Gamma} (V,n) \langle \nabla f, n \rangle + \kappa f \rangle \ ds + \int_{\Gamma} (V,n) (\langle \nabla f, \dot{n}[W] \rangle + \kappa [W]f) \ ds \\
&\quad + \int_{\Gamma} (V,n) \langle \nabla f'[W], n \rangle + \kappa f'[W] \rangle \ ds + \int_{\Gamma} (V,n) (\langle \nabla f'[V], n \rangle + \kappa f'[V]) \ ds \\
&\quad + \int_{\Gamma} f''[V,W] \ ds.
\end{align*}
\]

(29)

For the normal variation term \( \langle V,n \rangle \langle \nabla f, \dot{n}[W] \rangle \), one has

\[
\begin{align*}
\int_{\Gamma} (\langle V,n \rangle \nabla f, -\nabla \Gamma (W,n)) \ ds &= \int_{\Gamma} (W,n) \left[ \text{div} \Gamma (\langle V,n \rangle \nabla f) - \kappa \langle V,n \rangle \nabla f, n \rangle \right] \ ds \\
&= \int_{\Gamma} (W,n) \langle V,n \rangle \text{div} \Gamma f + (W,n) \langle \nabla f, \nabla \Gamma (V,n) \rangle - \kappa (W,n) \langle V,n \rangle \nabla f, n \rangle \ ds.
\end{align*}
\]

(30)

Due to equation (16), it is easy to see that for \( V \) and \( W \) normal, one has

\[
\langle V, \dot{n}[W] \rangle = \langle V,n \rangle \langle n, \dot{n}[W] \rangle = 0
\]

(31)

and in particular

\[
\langle W,n \rangle \langle \nabla \Gamma (V,n) , \nabla f \rangle = \langle W,n \rangle \langle (DV)^T n, \nabla f \rangle = \langle (DV)^T W, \nabla f \rangle.
\]

(32)

The above expression becomes symmetric assuming independency between \( V \) and \( W \) as in (24). The transformation of equation (30) can be made more convenient by removing the normal component of \( \nabla f \), which leads to an expression involving the tangential Laplacian and no curvature, however, we rather use the present structure as this is the transformation the S-AD tool will be conducting natively.
For the curvature term, combine equation (23) with (16) and thereby reduce the asymmetry to the same situation as with the variation of the normal in equation (30), namely

\[(33)\]
\[
\int_{\Gamma} (V, n) \hat{\kappa}[W] f \, ds
\]
\[
= \int_{\Gamma} f(\nabla_{\Gamma} (V, n), \nabla_{\Gamma} (W, n)) - \langle V, n \rangle \langle \nabla_{\Gamma} f, n \rangle ds
\]
\[
= \int_{\Gamma} f(\nabla_{\Gamma} (V, n), \nabla_{\Gamma} (W, n)) + \langle V, n \rangle \langle W, n \rangle \Delta_{\Gamma} f + \langle W, n \rangle \langle \nabla_{\Gamma} (V, n), \nabla_{\Gamma} f \rangle \, ds
\]
\[
= \int_{\Gamma} f(\nabla_{\Gamma} (V, n), \nabla_{\Gamma} (W, n)) - \langle \nabla_{\Gamma}((V, n), \nabla_{\Gamma} f) + \langle W, n \rangle \langle \nabla_{\Gamma} (V, n), \nabla_{\Gamma} f \rangle \, ds.
\]

Hence, all remaining terms in equation (30) become symmetric using \((DV)^T W = 0\) again.

Finally, if the weak form of the surface shape derivative, formula (10), is applied repeatedly with the usual requirement of \((DV)^T W = 0\), the expression

\[(34)\]
\[
d^2 J_2[V, W] = \int_{\Gamma} f(\nabla_{\Gamma} V) \text{div}_{\Gamma} W + f[V] \text{div}_{\Gamma} W + f[W] \text{div}_{\Gamma} V
\]
\[
- f \text{tr}(DV DW) + \hat{f}[V, W]
\]
\[
+ f \left( (DV)^T n, DWn \right) + \langle DVn, (DW)^T n \rangle
\]
\[
+ f \left( (DW)^T n, (DV)^T n \right) - 2 \langle DVn, n \rangle \langle DWn, n \rangle \right) ds
\]

is created, which is immediately symmetric except for the \(\hat{f}[V, W]\) material derivative, which will be discussed later in Section 5.2.

Summarizing the above, one can see that the expressions arising in higher order shape derivatives can be very technical and having a computer aid at hand can be of great benefit. Furthermore, because any of these transformations requires detailed knowledge of the actual meaning of any variable in the code, the UFL programming language was chosen as an ideal basis for this endeavor.

2.5. Hessian for the Dido Problem. We can now use the results from Section 2.4 to determine the symmetrized strong form shape Hessian of the Dido Lagrangian, equation (2). For the volume part, we arrive due to (25) at

\[
d^2 J_1(\Omega)[V, W] = \int_{\partial\Omega} (W, n) \langle V, n \rangle \lambda \kappa \, ds
\]

For the surface part, we utilize (29) and arrive at

\[
d^2 J_2(\Omega)[V, W] = \int_{\Gamma} (W, n) \langle \nabla (\langle V, n \rangle \kappa), n \rangle \, ds + \int_{\Gamma} (W, n) \langle V, n \rangle \kappa^2 \, ds
\]
\[
+ \int_{\Gamma} (V, n \hat{\kappa}[W]) \kappa \, ds + \int_{\Gamma} (V, n \hat{\kappa}[W]) ds
\]
The first term is asymmetric and vanishes with assumption (24) and $0 = Dn \cdot n$, the
term involving the variation of the normal vanishes due to (31) and the curvature
variation reduces to the tangential Laplacian due to (33). Thus, using all simplifications
and symmetrizations, the second derivatives of the Dido problem are then given by

$$d^2 L(\Omega, \lambda)[V, W] = \int_{\partial \Omega} \langle W, n \rangle \langle V, n \rangle (\lambda \kappa + \kappa^2) + \langle \nabla_\Gamma \langle V, n \rangle, \nabla_\Gamma \langle W, n \rangle \rangle \, ds \quad (35)$$

$$\langle \nabla_\lambda dL(\Omega, \lambda)[W], \dot{\lambda} \rangle = \int_\Gamma \langle W, n \rangle \dot{\lambda} \, ds \quad (36)$$

$$d(\nabla_\lambda L(\Omega, \lambda), \dot{\lambda})[V] = \int_\Gamma \langle V, n \rangle \dot{\lambda} \, ds \quad (37)$$

As a remark, these expressions are given to verify the output of the S-AD tool
symbolically. However, the way the Hessian is derived here does not reflect the strategy
of the tool: While we have used tailor made Hessian expressions from Section 2.4
directly, these are not implemented in the tool. Rather, it will utilize first order
differentiation formulas only, but arrive at the same Hessian by utilizing them twice
and applying necessary symmetrizations and simplifications afterwards.

2.6. The Core Idea. A three step approach to solve the problem of automatically
transforming general expressions into their shape differentiated counterparts suggests
itself. As a first step, the input has to be transformed into one of the basic expressions
for integrals, such as equations (6), (8), (11). Then, as the second step, dependencies
on geometric quantities have to be detected and the geometric derivatives, i.e., (16)
and (21) have to be created. In both cases, these transformations can necessitate
using the tangential Stokes formula. Finally, as a third step, a simplification stage
will incorporate additional knowledge such as $(Dn)^T n = 0$ or $(DV)^T W = 0$, thereby
achieving symmetry in case of repeated differentiation. Ideally, this creates expressions
as convenient as (20) automatically. The above mentioned steps will form the three
core elements of the S-AD tool and they are presented algorithmically in Figure 5.
The majority of Section 4 will be used to discuss how the link between general UFL
input and the respective transformation formulas can be established.

3. The Unified Form Language.

3.1. UFL as a Means for Shape Differentiation. UFL is a very high level
domain-specific embedded language in Python [2, 3]. The design goal is to postulate
variational forms of partial differential equations in a near-mathematical language,
which can then be processed by a form compiler to produce actual low level code.

Consider for example the Poisson problem to find $u$ such that

$$-\Delta u = 2 \text{ in } \Omega, \quad u = 0 \text{ on } \partial \Omega$$

In variational form, this means to find $u \in U_0$ such that

$$\int_\Omega \langle \nabla u, \nabla v \rangle \, dx = \int_\Omega 2v \, dx \quad \forall v \in V_0.$$  

Here, $U_0$ is called the space of trial functions and $V_0$ the space of test functions. This
variational problem can be expressed very conveniently in Python using DOLFIN resp.
UFL:
from dolfin import *
MyMesh = UnitSquareMesh(10,10)

U = FunctionSpace(MyMesh, "CG", 1)

v = TestFunction(U)

v = TrialFunction(U)

a = (inner(grad(u), grad(v)))*dx
l = Constant(2.0)*v*dx
bc = DirichletBC(U, Constant(0.0), "on_boundary")

u = Function(U)
solve(a==l, u, bc)

For the purpose of this work, it is important to note that the assignments \(a = \ldots\) and \(l = \ldots\) in the example above can be seen as if a mathematical integral is assigned to program variables. Furthermore, the integrand of these expressions include finite element representations of mathematical functions. In this sense, the UFL language is conveying the actual mathematical meaning of variables and operators. The internal representation of these variables \(a\) and \(l\) is a tree and the S-AD tool is supposed to transform these tree representations into their shape differentiated counterparts.

With the original intention to describe finite element problems, there are minor shortcomings when trying to postulate general integrals within UFL. Considering for example the following integrals with trial-function \(u\) and test-function \(v\)

\[
a_1(\varepsilon, u, v) := \int_\Omega \varepsilon^2 (\nabla v, \nabla u) + v \cdot u \, dx
\]

\[
a_2(u, v) := \int_\Omega v \, dx \cdot \int_\Omega u \, dx,
\]

integral \(a_1\) is directly representable in UFL because it contains a test- and a trial-function and constitutes a classical bilinear expression. But expression \(a_2\) is not representable in UFL because the second integral contains a trial-function but no test-function and does not stem from a classical variational form of a partial differential equation. In the UFL tree representation, the integral is the root node for form \(a_1\), but for form \(a_2\) the root node would be the product operator. The S-AD tool presented here will only be able to process inputs where the root node is a functional, linear or bilinear form, as other integral expressions cannot be postulated in UFL right now.

One of the few instances where this matters is the barycenter/centroid

\[
C_i := \frac{1}{\text{Vol}(\Omega)} \int_\Omega x_i \, dx,
\]

which cannot be automatically shape differentiated by this approach, because it cannot even be postulated in UFL while keeping the domain dependency of the volume factor. For this special instance, a manual implementation of the shape derivative is provided within the S-AD framework. However, in most applications, the centroid constraint is postulated in addition to a volume constraint, in which case the volume factor in \(C_i\) can be treated as constant with respect to the domain \(\Omega\), making automatic treatment possible again.
3.2. Differentiation Capabilities of UFL. Another benefit of using UFL for automating shape derivatives is its differentiation capabilities. As an example, the computation of the normal variation term (13) for a repeated shape differentiation of the final volume form \( g(s,V,n) := \langle V(s), n(s) \rangle f(s) \) from (8) is considered. This expression constitutes the last term in (18) and it is our intention of enabling the S-AD software to automatically detect a linear dependency on the normal and thus create the directionald shape derivative for expressions with linear dependency on the normal, i.e., create (20) instead of (19).

The internal UFL tree representation of the primal boundary integral with integrand \( \langle V,n \rangle f \) is shown in Figure 1. In UFL, the notion Argument is used to refer to test- and trial-functions, which takes the role of \( V \) in this instance. Creating the inner variation of the normal in accordance to equation (13) can be achieved by invoking the Gâteaux derivative possibilities of UFL by specifying \( dn \) as the direction for a differentiation with respect to the component where \( n \) is entering the UFL form. In its present state, UFL distinguishes between operations with scalar or generalized tensor inputs. Thus, the scalar or “real” Product operation node is strictly different from the vector nodes Inner, Outer and Dot. Furthermore, the derivative functionality only features transformation rules for scalars. Consequently, the Gâteaux differentiation of the example input \( \langle V,n \rangle f \) is first transformed into component operations using the Indexed node, which extracts components of a given MultiIndex. Afterwards, the tree is processed in “coefficient-to-integral”, i.e., “bottom-to-top” order, where each relevant coefficient in the now scalar component-wise expression is replaced by zeros if there is no dependency on the respective component of the differentiation variable, or the respective derivative direction, otherwise. The tree of the component-wise form is then transformed in accordance to the standard rules of differentiation. The resulting new differentiated scalar UFL tree representation is shown in Figure 2.

A natural approach would be to now have the S-AD tool place equations (14) or (16) for \( \bar{n} [V] \) into a differentiated tree representation for the \( dn \) sub-branch. However, with the transformation into scalar operations on indexed quantities, the result is neither human readable nor can transformations (19) or (20) be easily applied. Furthermore, the scalar tree is also too low-level to build in additional knowledge and symmetry assumptions for repeated differentiation shape Hessians or to include the structure of the extensions \( n \) and \( \kappa \) into the tubular neighborhood. The following section introduces
new transformations for the tree with vector-nodes overcoming these difficulties.

4. Implementation of the S-AD Scheme. We will now discuss the automatic creation of boundary representation shape derivatives within UFL. Implementation of the two basic formulas (8) and (12) is very straightforward by changing the root node to an exterior facet integral and by applying gradient operators to the integrand where needed. However, either the local shape derivative term involving \( f'[V] \) or the material derivative \( \dot{f}[V] \) must be processed further to allow for a repeated application of the S-AD procedure with symmetric results.

4.1. Derivative Nodes.

4.1.1. Local and Spatial Derivatives. To achieve symmetry via repeated differentiation, a new layer of differentiation rules is introduced to UFL with the intention of omitting reversal to component expressions, both for Gâteaux and spatial derivatives. For tensors \( a, b \) of appropriate dimensionality, the new Gâteaux rules include

\[
\begin{align*}
\text{Inner}(a, b) & \rightarrow \text{Inner}(a, b') + \text{Inner}(b, a') \quad \text{(if dimensionality is equal)} \\
\text{Inner}(a, b) & \rightarrow \text{Dot}(a, b') + \text{Dot}(b, a') \quad \text{(if } a, b \text{ are vectors)} \\
\text{Dot}(a, b) & \rightarrow \text{Dot}(a, b') + \text{Dot}(a', b) \\
\text{Grad}(a) & \rightarrow \text{Grad}(a') \\
\text{Div}(a) & \rightarrow \text{Div}(a')
\end{align*}
\]

(38)

It is worth noting that within UFL, for a matrix \( A \) and a vector \( v \), the relationship \( \text{Dot}(v, A) = A^T v \) and \( \text{Dot}(A, v) = A v \) holds, and no new rules for transposes are needed.
This is particularly important when defining the spatial gradient for non-terminal expressions such as Inner$(a, b)$. Here, the rule

$$\text{Grad(Inner}(a, b)) \leadsto \text{Dot}(b, \text{Grad}(a)) + \text{Dot}(a, \text{Grad}(b))$$

is implemented. Thus, the transformation $\nabla(V, n) = (DV)^T n + (Dn)^T V$ is applied correctly.

Finally, UFL does not distinguish between row and column vectors, so for an expression $D(av) = aDv + vDa$, with $a$ scalar and $v$ a vector, we also include the transformation $\text{Grad}(av) = a\text{Grad}(v) + \text{Outer}(v, \text{Grad}(a))$, and for the outer product, we introduce $(a_1 \otimes b_1) \cdot b_2 = (a_1, b_1) \cdot (b_1, b_2)$. With this set of rules in place, the UFL tree when S-AD processing a linear expression such as the first part of (8) stays exactly the same as for the primal problem presented in Figure 1 and one of the sub-branches stemming from (14) or (16) can be appended over the Coefficient $n$, resp. $dn$ node.

It is worth noting that within UFL, the Grad operator will always apply spatial derivatives with respect to all spatial dependencies. In particular, this means that a naïve application of the UFL Grad to $f$ in equation (11) will generate unwanted spatial derivatives of the normal in cases such as equation (13), where there is an additional geometric or spatial dependency

$$\langle \nabla^{\text{tot}} f(s, n(s)), n(s) \rangle = \langle \nabla_1 f(s, n(s)), n(s) \rangle + \langle Dn(s)^T \nabla_2 f(s, n(s)), n(s) \rangle$$

$$= \langle \nabla_1 f(s, n(s)), n(s) \rangle + \langle \nabla_2 f(s, n(s)), Dn(s)n(s) \rangle$$

and the latter extra term will be removed from the expression once additional knowledge about the extension of the normal field is incorporated automatically. A similar situation arises with curvature.

4.1.2. Material Derivatives. To achieve convenient weak form shape Hessian expressions as in equations (26) and (34), material and spatial derivatives need to commute as seen in equation (27). It is worth noting that when $f$ stems from the variational formulation of a PDE, more arbitrary spatial derivatives such as $\text{div}$ or $\text{curl}$ need to be dealt with. To this end, it is very convenient that spatial derivatives form their own nodes in the UFL tree and we likewise implement node transformation rules for each of those. This is another situation where the high level programming paradigm really shines, as recognizing spatial derivative from elementary code would be very difficult.

4.2. Maximally Expanded UFL Trees. To make the transition from equation (16) and (18) to equation (19), complex transformations are needed. In particular, one tangential differentiation operator in (16) needs to be swapped, which requires the application of the tangential divergence theorem, equation (17), to an arbitrary UFL form. Furthermore, we need to solve pattern recognition problems for finding simplifiable expressions such as $Dn n = 0$ or $(Dn)^T n = 0$ within the tree.

4.2.1. The Divergence Theorem. The above intention poses several challenges. On the one hand, the vector expression $v$ in equation (17) need not be a single UFL coefficient, but can be created by almost any arbitrary form, e.g.,

$$\epsilon (v_1 + v_2, \nabla_\Gamma a) = \langle \epsilon v_1 + \epsilon v_2, \nabla_\Gamma a \rangle = \langle v_1 + v_2, \epsilon \nabla_\Gamma a \rangle = \langle v_1 + v_2, \nabla_\Gamma (\epsilon a) \rangle$$

for $\epsilon$ scalar and constant. All of these representations have radically different UFL trees. On the other hand, there is no $\nabla_\Gamma$-operator in UFL and a simple substitute via
∇Γa = ∇a − ∂a∂nn increases the difficulty of recognizing pattern (39) in the UFL tree even more, as the subtraction can happen in different branches, making this pattern detection a non-local operation, potentially involving the tree as a whole.

The latter can be overcome by introducing the ∇Γ operator as a new node in the UFL tree or, more quickly, by first using a single vector coefficient function ˜v := ∇Γa as a marker and a substitution afterwards. To overcome the pattern recognition problem,

\[
\begin{align*}
\text{exterior facet integral} & \quad \text{Sum} \\
\text{Inner/Dot} & \quad \text{Sum} \\
\hat{v} & \quad \text{vector } v \\
\text{Product} & \quad \text{Sum} \\
\text{Scalar } c & \quad \text{Inner/Dot} \\
\text{Scalar } c & \quad \text{Scalar } c \\
\hat{v} & \quad \text{vector } v \\
\end{align*}
\]

(a standard representation, i.e., “maximally expanded”, representation of the UFL tree is first created by re-sorting the tree. The desired sorting is shown in Figure 3 for the expression

\[ \int_{\Gamma} \langle \hat{v}, (1 + c)v \rangle + c... ds. \]

All Sum operations are sorted closest to the integral node and all “scalar”, e.g., Product, and all “vector-to-scalar” nodes, Inner and Dot, are attached to those sums. Furthermore, all constants are pulled out of gradient nodes and the rules of expanding derivatives such as ∇ and div are applied. This sorting is achieved by having the enhanced differentiation transformation (38) apply the distributive laws in addition to the rules of expanding scalar and vector differentiation. If re-interpreted as an equation, this means (39) is always brought into the form

\[ (c \cdot v_1, \hat{v}) + c(v_2, \hat{v}). \]

Hence, there is still an ambiguity how to deal with scalar sub-branches c. Because UFL does not have a native node for scaling a vector, the two summands in (40) create
radically different tree representations as shown in Figure 4a and 4b. Depending on whether we want to apply the divergence theorem or symmetry assumptions, either of the two representations is preferred.

Fig. 4: UFL tree Sum sub-branch transformation and identification for application of the divergence theorem and vector-product elimination.

After achieving the maximally expanded form, the linearity of the integral means any Dot/Inner branch can be replaced separately by the tree representation of (17). To achieve this, we sort each of the Sum inputs in Figure 3 to match the situation of Figure 4a with the Inner and \( \tilde{v} \)-node the top-most node to any of the sums. That is, scalars are “pulled-in” to the non-marker vector-node \( v \). Afterwards, the gradient operator can be removed from the marker \( \tilde{v} \) or the branch is eliminated in accordance to (17). To re-establish human readability, the resulting expression post divergence theorem is then transformed back into the shape of Figure 4b. Naturally, this methodology and sorting process is applicable in general and can also swap the \( \text{div}_\Gamma \) operator, which is needed to process curvature dependencies as in equation (23).

4.2.2. Including Simplifications and Symmetry. As discussed in Section 2.4, symmetry in second derivatives is generally reintroduced by the assumptions \( V = \langle V, n \rangle n, W = \langle W, n \rangle n, Dn n = 0 \) and \( (DV)^T W = 0 \) and incorporating these into an arbitrary shape-differentiated UFL tree can pose challenges in pattern recognition. To this end, we assume the UFL tree is given in the maximally expanded form with scalars “pulled-out”, as in Figure 3 and 4b. In maximally expanded form, the above terms manifest as

\[
0 = (DV)^T W \Rightarrow \langle W, n \rangle (DV)^T n = 0 \\
V = \langle V, n \rangle n \Rightarrow (Dn)^T V = (V, n)(Dn)^T n = 0
\]
and each of the expressions \((Dn)^T n = 0\), \(Dn n = 0\), \((DV)^T W = 0\), \((DV)^T n = 0\) and \((Dn)^T V = 0\) cannot be expanded further and will thus appear in the maximally expanded tree where they can be found and the respective branches can then be removed.

![Flowchart Diagram](image)

Fig. 5: Rough overview of the flowchart of the S-AD program. Core Step 2 shows the differentiation of the normal only, curvature is handled analogously. “Weak sd” and “strong sd” refers to whether the weak or strong form of the shape derivative is desired.
Because most of the above pattern can also present themselves in adjoined form, that is \( \langle v_1, A \tilde{v} \rangle = \langle A^T v_1, \tilde{v} \rangle \), the pattern recognition sweep is performed twice, where in any sequence of Inner / Dot to Dot nodes the adjoint structure is also checked.

Each of the high-level transformations to create the shape derivative in UFL thus requires either tree 4a or 4b. To this end, we establish the "sum first" representation of Figure 3 in general, and then produce either of the two representations of Figure 4 on demand within each subroutine of the derivator. All of the above elements are then combined into the actual differentiation routine as shown in Figure 5.

5. S-AD Applications.

5.1. A Purely Geometric Problem. We return to Dido’s Problem (1) as a first test case. The Lagrangian (2) is implemented and automatically discretized using the following DOLFIN code, where the real number \( \text{Vol}_0 \) will be added to the respective component after assembly into an array because it does not involve a test-function and would violate the UFL restrictions as discussed in Subsection 3.1.

```python
from dolfin import *
from SAD_geometry import VolumeNormal, ComputeDivNVolume
from ShapeOpt import *

mesh = UnitSquareMesh(10, 10)
FV = VectorElement("CG", mesh.ufl_cell(), 1)
FR = FiniteElement("R", mesh.ufl_cell(), 0)
TotalSpace = FunctionSpace(mesh, FV*FR)
Q0 = Function(TotalSpace)
(V0, lmb0) = Q0.split()
(V, lmb) = TestFunctions(TotalSpace)
(W, dlmb) = TrialFunctions(TotalSpace)
OneV = Expression("1.0", domain=mesh)
L = OneV*dx(domain=mesh) + lmb0*(OneV*dx(domain=mesh))

N = VolumeNormal(mesh)
kappa = ComputeDivNVolume(mesh)
sd = ShapeDerivative(L, mesh, V, n=N, State=lmb0, StateDirection=lmb,
    kappa=kappa, Is_Normal=V, GenerateSurface=True)
sh = ShapeDerivative(sd, mesh, W, n=N, State=lmb0, StateDirection=dlmb,
    kappa=kappa, Is_Normal=V,W, SymmetryDirection=V, GenerateSurface=True)
```

The S-AD derivator then automatically resolves the respective `ShapeDerivative` calls for the first order derivative `sd` to

\[
\begin{align*}
\text{sd} & = \{v_0[2]*One\}*dx + \{lmb0*One*(((v_0[0], v_0[1])) : (n))\}*ds \\
& + \{One*Curvature*(((v_0[0], v_0[1])) : (n))\}*ds
\end{align*}
\]

where \( v_0[2] \) denotes the test-function corresponding to the adjoint variable \( lmb0 \) and \( v_0[0], v_0[1] \) denotes the test-function of the perturbation field \( V \) in 2D. Thus, the first line is the same as equation (4) without the \( \text{Vol}_0 \) constant. Furthermore, the second and third line above equal the maximally expanded form of (3).

The second order derivative \( \text{sh} \) is being derived as
Automatic Generation Weak and Strong Shape Hessians

\[ \text{sh} = (v_0[2] \cdot \text{One}((v_1[0], v_1[1]):(n)) \cdot \text{ds} \]

\[ + (\text{Curvature} \cdot \text{lm}0 \cdot \text{One}((v_0[0], v_0[1]):(n)) \cdot (v_1[0], v_1[1])) \cdot \text{ds} \]

\[ + (\text{Curvature} \cdot \text{One} \cdot \text{Curvature}((v_0[0], v_0[1]):(n)) \cdot (v_1[0], v_1[1])) \cdot \text{ds} \]

\[ - \{\text{One}((n):(\text{grad}(v_0[0], v_0[1])) \cdot (n)) \cdot (v_1[0], v_1[1])) \cdot \text{ds} \]

\[ + \{\text{One}((n):(\text{grad}(v_0[0], v_0[1])) \cdot (n). \text{grad}(v_1[0], v_1[1])) \} \cdot \text{ds} \]

where \( v_{-1} \)-quantities denote the respective trial-functions. The first line corresponds to the off-diagonal term (36) and the third line corresponds to the off-diagonal term (37). The second line corresponds to the first part of (35) and the last two lines form the tangential Laplacian in (35). In summary, a rather convenient expression of the strong form second order derivative is found automatically and all simplifications have been applied.

Newton’s method is used to solve this optimization problem. Thus, we compute the next iterate via

\[ \Omega_{k+1} := \{x_k + W(x_k) : x_k \in \Omega_k\}, \quad \lambda_{k+1} := \lambda_k + \lambda, \]

where the updates \((W, \lambda)\) are such that

\[ \text{KKT}(V, W, \tilde{\lambda}, \lambda) = -\text{sd}(V, \tilde{\lambda}) \]

for all \( V \) in the CG1 finite element space and \( \tilde{\lambda} \) in \( \mathbb{R} \). In an unmodified Newton method, the KKT-matrix would be given by the shape Hessian \( \text{sh} \). But assembling \( \text{sh} \) into a matrix here will result in a rank deficit unless one uses test- and trial functions on the boundary only. Also, we are interested in a mesh movement everywhere and \( \text{sh} \), like the mathematical expressions (35) (36) and (37) from the shape derivative of the Lagrangian, only consists of boundary integrals. Thus, we fix the rank deficit of the KKT-system by including a mesh deformation procedure for the volume nodes by using

\[ \text{KKT}(V, W, \tilde{\lambda}, \lambda) := \text{sh}(V, W, \tilde{\lambda}, \lambda) + \text{md}(V, W), \]

where the additional term \( \text{md} \) is given by

\[ \text{md}(V, W) := \int_{\Omega} \langle V, W \rangle + 0.1 \cdot \langle \nabla V, \nabla W \rangle \, dx. \]

A physical interpretation of this procedure is that the resulting deformation of the boundary is diffusing into the interior. The mass-matrix in \( \text{md} \) ensures solvability without specifying boundary conditions and the diffusivity of 0.1 was found empirically. Another interpretation of this procedure is projecting the boundary update into \( H^1(\Omega) \), using a scaled norm. Summarizing the above, the mesh deformation to adapt the interior to the new boundary can be seen as “built-in” into the Newton direction.

The resulting optimal shape is shown in Figure 6, where also a 3D example is presented. As of version 2016.1, FEniCS does not provide a vertex normal nor any curvature information. To this end, we construct a vertex-normal as a CG-1 FEniCS vector function in the volume by first averaging edge-normals on the surface into vertices and then extending this function into the volume. A curvature extension is then approximated by applying the tangential divergence to this vertex-normal function with some minor Laplacian-type smoothing.
5.2. CFD and Navier–Stokes. One of the most natural applications involving shape optimization problems is drag minimization in fluid dynamics. Although most aerodynamic applications typically seek to minimize the force applied to a flow obstacle \[18, 27, 34\], the problem of minimizing energy dissipation in a viscous, incompressible channel flow around an obstacle is better understood mathematically. Indeed, thanks to the works \[23, 24\], the optimal profile in a Stokes flow is known analytically and studies on robust mesh deformation procedures and shape Hessians can be found in \[28, 31\]. Furthermore, the problem of minimizing shear in a channel flow around an obstacle also serves as a model problem for a variety of approaches to shape optimization \[12\] and is given by

\[
\min_{\Gamma_0} J(u, p, \Omega) := \int_{\Omega} \eta (\nabla u, \nabla u) \, dx
\]

subject to

\[
-\eta \Delta u + \rho Du \cdot u + \nabla p = 0 \text{ in } \Omega
\]
\[
\text{div } u = 0 \text{ in } \Omega
\]

\[
\int_{\Omega} 1 \, dx = \text{Vol}_0
\]

\[
\frac{1}{\text{Vol}(\Omega)} \int_{\Omega} x_i \, dx = bc_i, \ i = 1, 2,
\]
where the last constraint means the barycenter must not change. Here, \( \Omega \) describes a flow channel with fixed outer walls featuring an obstacle \( \Gamma_0 \) as a hole inside. The geometry is also shown in parts in Figure 7. Problem (44) is thought to have additional Dirichlet-conditions for the respective walls and inflow boundaries. For simplicity, those Dirichlet-conditions are considered being embedded into the respective function space for the velocity \( u \). Furthermore, \( p \) denotes the pressure and the viscosity \( \eta \) is a positive constant. Similarly, \( \rho \) denotes the constant density. The shape derivative in strong form, that is the shape derivative of the objective and the PDE constraints but not including the volume and barycenter, can for example be found in [22, 26, 28, 29].

The Lagrangian in UFL is given by

\[
L = \eta \cdot \text{inner}(\nabla u_0, \nabla u_0) \cdot dx + (\eta \cdot \text{inner}(\nabla u_0, \nabla l u_0) + lp_0 \cdot \text{div}(u_0) - p_0 \cdot \text{div}(lu_0)) \cdot dx
\]

The S-AD software is instructed to generate the UFL form of the gradient and Hessian of the Lagrangian in all variables via the command

\[
dL = \text{ShapeDerivative}(L, \text{mesh}, V, n=N, \text{State}=q, \text{StateDirection}=\text{TestFunction(TotalSpace)}, \kappa=kappa, \text{boundary_parts}=\text{boundary_parts}, \text{Is_Normal}=[V], \text{GenerateSurface}=\text{False})
\]

\[
\text{KKT} = \text{ShapeDerivative}(dL, \text{mesh}, W, n=W, \text{State}=q, \text{StateDirection}=\text{TrialFunction(TotalSpace)}, \kappa=kappa, \text{boundary_parts}=\text{boundary_parts}, \text{Is_Normal}=[V, W], \text{SymmetryDirection}=V, \text{GenerateSurface}=\text{False})
\]

Here, supplying the \text{State} = q \text{ command paired with GenerateSurface = False instructs the software to check for inner material derivative dependencies as in equation (7). In mathematical notation, the weak form shape derivative is found to be

\[
dJ(u, p, \Omega)[V] = \int_\Omega \text{div } V \left(\eta \|
abla u\|^2 + \eta \langle \nabla \lambda_u, \nabla u \rangle + \rho \langle \lambda_u, Du \cdot u \rangle - p \text{div } \lambda_u + \lambda_p \text{div } u \right) \, dx
\]

\[
+ \int_\Omega 2\eta \langle (DV)^T \nabla u, \nabla u \rangle - \eta \langle (DV)^T \nabla u, \nabla \lambda_u \rangle - \eta \langle \nabla u, (DV)^T \nabla \lambda_u \rangle \, dx
\]

\[
+ \int_\Omega -\rho \langle \lambda_u, DuDVu \rangle + p \text{tr}(D\lambda_u DV) - \lambda_p \text{tr}(DuDV) \, dx,
\]

where \((\lambda_u, \lambda_p)\) solves the adjoint Navier–Stokes problem

\[
-\eta \Delta \lambda_u + \rho \left( (Du)^T \cdot \lambda_u - D\lambda_u \cdot u \right) - \nabla \lambda_p = 2\eta \Delta u
\]

\[
div \lambda_u = 0
\]

in \( \Omega \) with Dirichlet zero for \( \lambda_u \) on all walls and at the inflow boundary. The Hessian is too lengthy to discuss here. Utilizing the assumption of the adjoint variables solving (47) exactly, the above derivative expression does not contain any material derivatives \((\dot{u}[V], \dot{p}[V])\), meaning the \( \int[V,W] \) quantity in (26) does not pose a problem. A rigorous theoretical study of the problem can for example be found in [6].
We employ the same Newton optimization scheme as in (41) and (42) except we now operate on the PDE state, the adjoint for the PDE and the adjoints for the geometric constraints. The mesh deformation is also embedded into the Newton step similar to (43). The resulting sparse linear systems are solved using UMFPACK, which avoids difficulties with the saddle point structure of the respective matrices. With the focus on the development of the S-AD tool here, we kindly refer to a follow-up publication for details on a shape Newton optimization scheme based on the weak form. Initial and optimal shape are shown in Figure 7.

![Fig. 7: Initial and optimized obstacle in a Navier–Stokes flow.](image)

6. Conclusion and Outlook. The main purpose of this work is discussing how modern, high level programming languages based on domain specific languages can be used to incorporate a new level of structure exploitation when mitigating simulation code to solvers for optimization problems. This is done by creating a Shape-AD (S-AD) toolbox, which can automatically generate first- and higher order shape derivatives in both the classical “strong” or Hadamard boundary representation and the “weak” or volume representation for both boundary and volume integrals. Additional dependencies on normals and curvatures can also be handled. In particular, deriving the Hadamard-form marks the first application of the divergence theorem for an AD-like tool. Likewise, generating the weak form marks the first time where a code transformation software distinguishes between Gâteaux, material and spatial derivatives. The subsequent creation of shape Hessians via repeated differentiation poses problems in pattern recognition to achieve a symmetric expression.

To this end, we focus on UFL, which is a possible input to many modern PDE-frameworks and the above transformation problems are tackled via multiple re-sorting strategies of the tree representation of the mathematical problem. The resulting methodology is tested on Dido’s problem as a purely geometric test-case in strong form and the energy dissipation problem in a Navier–Stokes flow in weak form.

The author wishes to thank Simon Funke, Nicolas Gauger and Andrea Walther for their support with respect to UFL and algorithmic differentiation.

REFERENCES

Automatic Generation Weak and Strong Shape Hessians


Three-Dimensional Large-Scale Aerodynamic Shape Optimization Based on Shape Calculus

Stephan Schmidt∗ Imperial College, London, England SW7 2AZ, United Kingdom
Caslav Ilic† DLR, German Aerospace Center, 38108 Braunschweig, Germany
Volker Schulz‡ University of Trier, 54296 Trier, Germany
and
Nicolas R. Gauger§ RWTH Aachen University, 52062 Aachen, Germany

DOI: 10.2514/1.J052245

Large-scale three-dimensional aerodynamic shape optimization based on the compressible Euler equations is considered. Shape calculus is used to derive an exact surface formulation of the gradients, enabling the computation of shape gradient information for each surface mesh node without having to calculate further mesh sensitivities. Special attention is paid to the applicability to large-scale three dimensional problems like the optimization of an Onera M6 wing or a complete blended-wing–body aircraft. The actual optimization is conducted in a one-shot fashion, in which the tangential Laplace operator is used as a Hessian approximation, thereby also preserving the regularity of the shape.

Nomenclature

\[ a = \text{rotation of the coordinate system} \]
\[ A_i = \text{Euler flux Jacobian matrices} \]
\[ B = \text{reduced Hessian} \]
\[ c = \text{vector constraints, e.g., flow solver residual} \]
\[ C_D = \text{drag coefficient} \]
\[ C_L = \text{lift coefficient} \]
\[ C_P = \text{pressure coefficient} \]
\[ D_z z = \text{Jacobian of quantity } z \text{ with respect to } x \]
\[ D_x z = \text{reduced Jacobian of quantity } z \text{ with respect to } x \]
\[ dA = \text{general integration measure for volume integrals} \]
\[ ds = \text{general integration measure for surface integrals} \]
\[ dz[V] = \text{material derivative of quantity } z \text{ in direction } V \]
\[ E = \text{total energy} \]
\[ f = \text{objective function, standard optimization problem} \]
\[ g = \text{shape gradient} \]
\[ H = \text{enthalpy} \]
\[ H_{z_1 z_2} = \text{second partial derivative of the objective with respect to } z_1 \text{ and } z_2 \]
\[ I = \text{identity matrix} \]
\[ J = \text{objective function, shape optimization problem} \]
\[ \mathcal{L} = \text{Lagrangian} \]
\[ \ell = \text{scalar constraints, e.g., lift or structural constraints} \]
\[ n = \text{outer normal} \]
\[ p = \text{pressure} \]
\[ q = \text{design vector} \]
\[ t = \text{degree of deformation applied by } T_p \text{, baseline design corresponds to } t = 0 \]
\[ T = \text{bijective family of mappings applying the shape deformation} \]
\[ U = \text{vector of conserved variables} \]
\[ u = \text{velocity vector} \]
\[ u_0 = \text{conserved variables with enthalpy as last component} \]
\[ V = \text{smooth vector field prescribing the deformation direction} \]
\[ x_i = \text{coordinate axis, chord direction} \]
\[ x_i = \text{coordinate axis, span direction} \]
\[ x_i = \text{coordinate axis, thickness direction} \]
\[ y = \text{state vector} \]
\[ z[V] = \text{shape derivative of quantity } z \text{ in direction } V \]
\[ \alpha = \text{angle of attack} \]
\[ \Gamma = \text{unknown boundary to be optimized} \]
\[ \gamma = \text{adiabatic exponent} \]
\[ T_0 = \text{Euler slip wall, i.e., the aircraft surface} \]
\[ \delta_{ik} = \text{Kronecker symbol} \]
\[ \epsilon = \text{smoothing parameter} \]
\[ \kappa = \text{additive mean curvature} \]
\[ \lambda = \text{adjoint variable} \]
\[ \mu = \text{adjoint variable for vector constraints} \]
\[ \nu = \text{adjoint variable for scalar constraints} \]
\[ \rho = \text{density} \]
\[ \tau = \text{one-shot dampening factor} \]
\[ \Omega = \text{domain occupied by the fluid} \]

I. Introduction

A DJOIN-T-BASED aerodynamic shape optimization, especially for industry-sized problems, has in the past almost always followed a parametric approach, meaning that parts of the aircraft like the wing cross sections are deformed by adding smooth ansatz functions, such as the popular Hicks–Henne functions [1], to the geometry. Other approaches frequently encountered for CAD-free fully three-dimensional (3-D) parameterizations are, for example, perturbing the control points of spline surfaces or free-form deformations. All of these approaches have in common that the actual optimization problem is considered postparameterization, meaning...
The gradient is computed according to the standard Lagrange formula:
\[
\frac{\partial J}{\partial q} = \frac{\partial J}{\partial q} - \lambda^T \frac{\partial c}{\partial q}
\]
(1)
where \(\lambda\) is the corresponding adjoint variable solving
\[
\frac{\partial c}{\partial q} \lambda = \frac{\partial J}{\partial y}
\]
(2)
The adjoint flow solver is therefore independent of the shape optimization nature of the problem as only the derivatives with respect to the flow states are needed. However, in order to construct the gradient out of the primal and adjoint states, the parameterization of the shape must be considered. Especially the term \(\frac{\partial c}{\partial q}\) requires knowledge of the sensitivity of the flow solver residual with respect to mesh nodes positions effected by the parameterization \(q\). Although this approach is known to be applicable and well working, one is often forced into finite differencing for these terms [2,3]. This often makes very fine parameterizations, such as using the mesh node positions itself, rather impractical if not prohibitive as the time to compute the adjoint flow solution is indeed independent of the number of design parameters, but the gradient computation actually is not. Although it is possible to counter this problem by introducing another adjoint for the mesh deformation, using, e.g., algorithmic differentiation on the mesh deformation tools in reverse mode [4] or a continuous approach [5], special care must be taken not to run into memory limitations by considering the entire design chain at once. These problems arise in part due to the fact that in order to maintain mesh quality, usually the whole volume mesh, or at least significant parts of it, need to be moved and adapted, even for small changes such as moving a single surface node. Thereby, any locality that might otherwise be present due to the local finite element or finite volume stencils is destroyed. Furthermore, most flow solvers, although they have an adjoint capability, cannot efficiently compute the derivative of the flow residual with respect to the input mesh. Finally, the mapping of any other set of design parameters, e.g., CAD parameters, to the actual mesh nodes is frequently hidden within closed source software environments, and no derivative is available either.

The alternative is to treat the problem in a nonparametric fashion, while still staying within the framework of the continuous adjoint approach. This has the additional advantage that some of the standard infrastructure necessary for adjoint-based optimization can be reused easily. In the past, nonparametric approaches have been used to derive optimal shapes for certain flow situations on a theoretical level. For example, in [6,7], a rugby ball–like shape is shown to be optimal for creeping Stokes flows, whereas in [8], optimality of the so-called Sear–Haack body for inviscid compressible flow is shown. Nonparametric approaches can also be found in [9,10], but they are hardly used for any actual computations. The idea considered in the present work is to use shape calculus to differentiate the aerodynamic forces directly with respect to the input geometry, thereby arriving at a form of Eq. (1), which is specific for shape optimization problems and does not need explicit knowledge of the problematic partial derivatives. Shape calculus or shape sensitivity analysis describes the mathematical topic when the shape of a domain is the unknown. The methodology can be used to arrive at exact surface formulations of the gradients for shape optimization problems, which is often termed the Hadamard form [11,12]:
\[
\Gamma^{k+1} := \{x + \tau g(x)n(x) : x \in \Gamma^k\}
\]
where \(\tau\) is the step length of the algorithm. Therefore, using the surface mesh node positions is a natural choice, and furthermore, the deformation of the volume mesh is completely removed from the derivative chain. Although the volume mesh nodes must of course still be somehow adapted to the new surface geometry, the derivative of the mesh deformation and the variation of the flow residual with respect to the design are not required for an exact gradient evaluation because they are included in \(g\) on an analytic level. Thus, an advantage of the approach discussed here is that knowledge of the volume mesh deformation scheme is not required, one can use any volume mesh generator while conducting the actual optimization. This is due to the analytic nature of the formulation, and so a one-to-one correspondence of the surface tessellation and the volume mesh is not necessary throughout the optimization process, hence the name nonparametric optimization. Using the surface mesh node positions as design parameters encompasses any other form of parameterization, as any parameterization can be interpreted as a restriction of the design space, i.e., a projection of the mesh node movement into a coarser space. Although there might be frequent occasions in which engineering restrictions necessitate a restriction of the design space, being able to efficiently optimize all surface node positions is a strong indication that any other parameterization is also going to work well, unless the design space becomes too restrictive. There are previous works in aerodynamic shape optimization that use all surface mesh node positions [13], but usually, the considerable overhead in computing the gradient based on Eq. (1) has made this approach very inefficient [14].

To summarize the preceding, we present the applicability of a novel adjoint-based shape optimization algorithm to very-large-scale 3-D problems. The algorithm is very fast because it exploits the mathematical problem structure and operates on surfaces alone. Furthermore, it does not require any partial derivatives of the mesh deformation. A possible downside could be discrepancies between the continuous and discrete models of the flow solver and applicability to shapes of a low regularity class.

Here, the shape gradient \(g\) is used in a one-shot optimization strategy similar to [15,16]. Being a reduced Sequential Quadratic Programming method, one-shot depends on a proper approximation of the reduced Hessian, for which the surface or tangential Laplace operator is used. Pseudodifferential operator symbol calculus conducted in [17,18] suggests that using a Hessian approximation based on an anisotropic operator with anisotropies in the chord and span directions would be best, but we found isotropic diffusion to be working very well also. Sometimes called gradient smoothing or the Sobolev gradient method, similar techniques have been used in [19,20] as a means to preserve the regularity of the aircraft shape.

Although the applicability to two-dimensional (2-D) airfoil optimizations using the compressible Euler equations has been previously considered in [21,22], the aim of this paper is to study the applicability to large-scale 3-D problems. To this end, the optimization of both the Onera M6 wing as well as the optimization of a complete blended-wing–body aircraft is shown. Special emphasis also lies on the correct computation of the respective surface quantities needed for evaluating the shape derivative on triangulated unstructured surface meshes. We found the combination of the one-shot approach paired with surface gradients based on shape calculus extremely efficient, such that a blended-wing–body aircraft could be optimized conveniently using up to 460,517 individual mesh node positions as design unknowns. Further considerations for the incompressible Navier–Stokes equations can be found in [21,23,24]. Potential flow inverse design is considered in [25].

II. Shape Calculus

A. Problem Introduction: Aerodynamic Forces

A very brief review of shape calculus is given next. More details on shape sensitivity analysis in general can be found in [11,12]. The inviscid fluid forces acting on the aircraft surface \(\Gamma_0\) are given by
\[
J(U, \Omega) = \int_{\Gamma_0} (p \cdot a) n \, dS
\]
(4)
where \(U := (\rho, p, u)\) are the conserved Euler state variables with \(\rho\) being the density, \(u = (u_1, u_2, u_3)^T\) is the velocity vector, and \(E\) is the...
total energy of the fluid. Furthermore, the pressure \( p \) is linked to the conserved state variables by the perfect gas law:

\[
p = (\gamma - 1)p\left(E - \frac{1}{2}\|u\|^2\right)
\]

with \( \gamma \approx 1.4 \) being the adiabatic exponent of air. The normal to the aircraft surface is denoted by \( n \), and \( \alpha \) is the rotation of the coordinate system, meaning for an angle of attack \( \alpha \) choosing

\[
a_g := (\cos \alpha, 0, \sin \alpha)^T
\]

leads to \( J \) being the aerodynamic inviscid pressure drag force. Similarly, choosing \( a \) as

\[
a_L := (-\sin \alpha, 0, \cos \alpha)^T
\]

will result in \( J \) being the lift force. In the following, it is thus sufficient to consider surface functionals only. Also, the lift and drag forces do not need to be treated separately.

### B. Shape Calculus for Surface Functionals

A finite deformation of the aircraft surface is thought to be given by

\[
\Gamma_0 := \Gamma(t_0) = \{\Gamma(x) : x \in \Gamma_0\}
\]

where \( \Gamma_t \) is a family of bijective mappings usually given by either the perturbation of identity

\[
\Gamma_t(x) = x + tV(x)
\]

or the speed method

\[
\frac{dx}{dt} = V(t, x), \quad x(0) = x_0 \in \Gamma_0
\]

Thus, the actual perturbation direction is given by the vector field \( V \), which is supposed to be Lipschitz continuous. Sometimes, \( V \) is also called the “design velocity.” For first-order calculus, both the perturbation of identity and the speed method are known to be equivalent [11,12]. Assuming enough regularity such that the chain rule holds, the preliminary shape derivative of Eq. (4) is given by

\[
dJ(U)[V] = \left[ \frac{d}{dt} \right]_{t=0} \int_{\Gamma_0} \langle p \cdot a, n \rangle \, dS + \int_{\Gamma_0} \langle p \cdot a, \nabla n \rangle \, dS
\]

Using standard shape differentiation techniques and tangential calculus [11,12,21,22], one arrives at

\[
\frac{d}{dt} \left. \int_{\Gamma_0} \langle p \cdot a, n \rangle \, dS \right|_{t=0} = \int_{\Gamma_0} \langle V, n \rangle \left[ \frac{\partial p}{\partial n} \cdot a, n \right] + \kappa \langle p \cdot a, n \rangle \, dS
\]

for the first term in Eq. (8). Using the same techniques, one can also arrive at

\[
\int_{\Gamma_0} \langle p \cdot a, \nabla n \rangle \, dS = \int_{\Gamma_0} \langle V, n \rangle [\text{div}_V(p \cdot a) - \kappa(p \cdot a, n)] \, dS
\]

where \( \text{div}_V \) is the surface or tangential divergence operator.

### C. Shape Calculus for the Local Shape Derivative of the State Equation

Adjoint calculus must now be used to remove the local shape derivative of the pressure \( p \) \( \{V\) in Eq. (8), which will be conducted analogously to [26,27]. Let the local shape derivatives of the conserved variables be given by

\[
U'[V] = (p'[V], (\rho u)'[V], (\rho E)'[V])^T
\]

They satisfy the linearized Euler equations given by

\[
\frac{\partial}{\partial x_1} (A_1 U'[V]) + \frac{\partial}{\partial x_2} (A_2 U'[V]) + \frac{\partial}{\partial x_3} (A_3 U'[V]) = 0
\]

inside the flow domain. Letting \( \lambda \) solve the adjoint compressible Euler equations,

\[
-A_1^T \frac{\partial}{\partial x_1} \lambda - A_2^T \frac{\partial}{\partial x_2} \lambda - A_3^T \frac{\partial}{\partial x_3} \lambda = 0 \quad \text{in } \Omega
\]

integration by parts in Eq. (11) shows that

\[
0 = \int_{\Omega} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{\partial}{\partial x_1} \lambda \, dV \, dS
\]

As discussed in [26,27] and given proper far-field adjoint boundary conditions, the relation

\[
0 = \int_{\Gamma_0} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{\partial}{\partial x_1} \lambda \, dV \, dS
\]

holds, where \( U_H \) is given by

\[
U_H := (\rho, \rho u_1, \rho u_2, \rho u_3, \rho H)^T
\]

Because of the fluid velocity satisfying the Euler slip boundary condition

\[
\langle u', n \rangle = 0
\]

on the aircraft surface, the local shape derivatives of the velocities are then correspondingly given by

\[
\langle u'[V], n \rangle = -\langle V, n \rangle \left[ \frac{\partial u}{\partial n} \right] - \langle u, \nabla n \rangle
\]

For more details, see [21,22]. Inserting this into Eq. (13), one arrives at

\[
0 = \int_{\Gamma_0} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{\partial}{\partial x_1} \lambda \, dV \, dS
\]

Adding the preceding to the preliminary gradient Eq. (8) and using Eq. (9) but not yet Eq. (10), one can see that

\[
\frac{dJ(U)[V]}{dt} = \int_{\Gamma_0} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{\partial}{\partial x_1} \lambda \, dV \, dS
\]

where \( \text{div}_V \) is the surface or tangential divergence operator.

If the adjoint boundary condition

\[
\langle \lambda_2, \lambda_3, \lambda_4 \rangle n + \langle a \rangle = 0
\]

is satisfied on the wing, the gradient will further simplify to

\[
\frac{dJ(U)[V]}{dt} = \int_{\Gamma_0} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{\partial}{\partial x_1} \lambda \, dV \, dS
\]

This especially means that existing adjoint flow solvers need not be modified to be useable for the computation of nonparametric shape.
derivatives, because both the adjoint field equation and the boundary conditions stay the same as in the classical approach. Using now also Eq. (10), one arrives at
\[ dJ(U)[V] = \int_{\Gamma_0} \left( V \cdot n \frac{\partial}{\partial n}(a, n) - \lambda U_H \frac{\partial}{\partial n} n \right) \] 
\[ + \text{div}_r(pa - \lambda U_H u) \] 
\[ dS \] 
(16)

which is the final form of the gradient.

Comparing Eq. (15) with Eq. (16), one can see that the final Hadamard form of the gradient requires the evaluation of the tangential divergence operator on the unstructured surface mesh of the aircraft, but this is equivalent to the computation of the mean curvature \( \kappa \) and the variation of the normal \( d\eta[V] \). This transformation is based on the tangential Stokes theorem, which states for a real valued differentiable function \( g \) and a vector valued function \( v \) on \( \Gamma_0 \)
\[ \int_{\Gamma_0} g \text{div}_r v + (\nabla_T g, v) dS = \int_{\Gamma_0} \kappa g(v, n) dS \]
For more information on the required tangential calculus, please see [11,21]. Because of more literature being available concerning mesh curvature of unstructured triangulated surfaces and the normal variation \( d\eta[V] \), it is quite easily computable on a discrete level, the latter approach was chosen.

The following methodology is used for evaluating Eq. (15) within the discretized framework of flow and adjoint solver. We assume that \( V \) is given by a linear hat function on the curved surface mesh, that is,
\[ V_k(x_k) = n(x_k) \delta_{ik} \]
with linear interpolation in between. Thus, index \( k \) here denotes the \( k \)-th component of the discretized gradient vector and at the same time also the node number of the \( k \)-th surface node. Index \( i \) corresponds to all nodes adjacent to node \( k \). At a point \( x \) that is on the surface but not part of the mesh, we assume a linear interpolation in between. When computing the \( k \)-th gradient component, which is determined by \( V_k \), the discretized integral (15) has therefore a compact support on the surface, and the integration reduces to the patch of surface triangles of which node \( k \) is the center. Finally, this integration is conducted using a standard quadrature rule for triangles. The process is visualized in Fig. 1. As discussed in [21], the variation \( d\eta[V],[x_k] \) of the face normal \( n_\tau \) is given by
\[ d\eta[V],[x_k] = \frac{n_\tau \times (x_k - x_{k+1})}{|T|} \]

where \( T \) is the surface triangle patch centered around node \( x_k \) with vertices \( x_l \). The mean curvature \( \kappa \) of the surface mesh is computed as described in [28] and the necessary normal derivatives are computed automatically within the flow solver software environment. Although not the main focus of this work, we also found the preceding methodology well applicable in less smooth situations. For example, in [22], we see the automatic formation of sharp leading edges in supersonic flow. Similarly, the automatic formation of sharp-nosed bodies of shiplike shapes in incompressible viscous fluids can be seen in [23,24].

III. One-Shot Optimization and Hessian Approximation

A. Overview of the One-Shot Method

To motivate the one-shot method, a standard minimization problem is considered:
\[ \min_{(y, q)} f(y, q) \]
subject to
\[ c(y, q) = 0 \quad \ell(y, q) = 0 \]
where \( c(y, q) = 0 \) refers to the flow solution residual being zero and \( \ell(y, q) = 0 \) means that additional constraints such as lift, volume, or bending stiffness are kept. The Lagrangian of the preceding is given by
\[ L(y, q; \mu, \nu) = f(y, q) - (\mu, c) - (\nu, \ell) \]
and using Newton’s method to solve the necessary optimality conditions of the preceding problem, the system
\[ \begin{bmatrix} H_{c y} & H_{c q} & (D, c)^T & (D, \ell)^T \\ H_{q y} & H_{q q} & (D, c)^T & (D, \ell)^T \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta q \\ (\Delta y, \Delta q, \Delta \mu, \Delta \nu)^T \end{bmatrix} = \begin{bmatrix} -V_L C \\ -V_q C \end{bmatrix} \]
needs to be solved with the actual design update given by
\[ (y_{k+1}, q_{k+1}, \mu_{k+1}, \nu_{k+1}) = (y_k, q_k, \mu_k, \nu_k) + (\Delta y, \Delta q, \Delta \mu, \Delta \nu)^T \]

Here, \( L \) denotes the Lagrangian. Assuming there exists an approximation of the Hessian of the form
\[ \begin{bmatrix} H_{c y} & H_{c q} \\ H_{q y} & H_{q q} \end{bmatrix} \approx \begin{bmatrix} 0 & 0 \\ 0 & B \end{bmatrix} \]
and further assuming \((D, c)^{-1}\) exists, a block Gauss elimination and replacing \( \Delta \nu \) with \( \nu_{k+1} = \nu_k + \Delta \nu \) results in the system
\[ \begin{bmatrix} B & \tilde{D}_f \\ (D, \ell)^T & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ \nu_{k+1} \end{bmatrix} = \begin{bmatrix} -\tilde{D}_f \\ -\ell + \lambda_c c \end{bmatrix} \]
(17)
where \( \lambda_c = (D, c)(D, c)^{-1} \), which is given by the adjoint flow solver. More details can be found in [15,21]. In the context of a standard minimization problem, the reduced gradient \( \tilde{D}_f \) of the objective function is given by
\[ \tilde{D}_f = \nabla_y f - (D, c)^T (D, c)^{-T} \nabla_y f \]
with an analogous definition of the reduced gradient \( \tilde{D}_f \) of the scalar constraints. Here, however, the respective shape derivatives will be used directly for \( \tilde{D} \) resulting in a shape one-shot method.

An additional aspect of the one-shot method, not directly visible in Eq. (17), is the fact that the state and adjoint flow variables are usually computed by an iterative flow solver. This usually results in any optimization procedures essentially becoming a two-loop approach: an outer optimization loop with several inner loops for the primal and...
respective adjoint iterative flow solvers. For the problems considered here, this two-loop approach is broken up by only performing a limited number of solver and adjoint iterations to compute the derivatives needed in Eq. (17). Thus, optimality of the design and feasibility of the flow state is computed simultaneously, thereby greatly reducing the wall-clock runtime. Please note that a discussion of the convergence properties of the one-shot method as well as a comparison of the discrete and continuous adjoint approaches is outside the scope of this paper. However, the method presented here can be both employed within the continuous and discrete adjoint approaches and also without a one-shot optimization scheme. More details on the convergence properties of one shot can be found in [29–31].

Assuming an exact Hessian approximation, B, an update based on Newton’s method as in Eq. (17) requires a step length restriction or line search only for convergence globalization with respect to the initial guess q0. However, this does no longer hold when using inexact quantities within the one-shot context. Therefore, a damping factor r is introduced in Eq. (17) that acts on the direction of optimality only but not feasibility. This leads to an update of the design being computed according to

\[ q_{k+1} = q_k - r B^{-1} \tilde{D}_f - B^{-1} \tilde{D}_f V k_{k+1} \]  

(18)

where r was chosen manually for each of the applications. More details can be found in [15].

B. Summary of the Algorithm

The following flow chart is a summary of the optimization algorithm. In the following, y is used to denote all physical unknowns of the flow solution, whereas c is the residual of the flow solver.

1) Initialization: k = 0, start with initial guess \( y^0 \), \( \Gamma_0^0 \), \( \rho_0^0 \), \( \rho_0^1 \). Set V as family of admissible deformations. Initialize damping r and Hessian approximation c. 
2) repeat
3) Perform \( n_a \) adjoint iterations to approximate \( k^{t+1} \) the solution to Eqs. (12) and (14).
4) Compute the shape derivative according to Eq. (15)
\[ \tilde{D}_f = (V, n) \left( \left( \frac{d}{dx}(a, n) + \kappa p(a, n) - \lambda U n \frac{d}{dn} \right) + \left( \frac{p a - 2 \lambda U n u, dn(V)}{B} \right) \right) \]
5) Perform \( n_f \) adjoint iterations for all additional (scalar) constraints \( \ell \). Obtain \( \ell^{k+1} \).
6) Use \( \ell^{k+1} \) to compute the Jacobian \( \tilde{D}_\ell \) of all additional (scalar) constraints.
7) Discretize \( B = (\ell \Delta + I) \) as an approximation of the reduced Hessian.
8) Solve Eq. (17) for \( k^{k+1} \)
\[ \tilde{D}_f B^{-1} \tilde{D}_f k_{k+1} = \ell - \ell^{k+1} \epsilon - \tilde{D}_f B^{-1} \tilde{D}_f \]
9) Solve Eq. (18) (and update boundary
\[ \Gamma_0^{k+1} = \{ x - \ell B^{-1} \tilde{D}_f = B^{-1} B \tilde{D}_f k_{k+1} | x \in \Gamma_0^k \} \]
10) Adapt volume mesh or remesh by any preferable means.
11) Update state \( y^k \) by \( n_f \) steps in the forward solver. Obtain \( y^{k+1} \).
12) until “convergence”

Note that, in our special case here, we set the family of admissible deformations \( V \) to contain a movement of all nodes except for those that define the planform. The stopping criteria of the algorithm is adapted from one of the stopping criteria of the flow solver and can be seen as a form of a Cauchy criterion: the optimization is considered to have terminated when there is no longer any progress within the objective.

C. Hessian Approximation

Crucial for the performance of one shot is having a good approximation of the reduced Hessian operator B. A natural choice would be the shape Hessian of the problem. However, shape Hessians are fairly complex objects even for moderate problems. Although they have been successfully used in solving shape optimization problems numerically \([32,33]\), it is often much more convenient to use a suitable approximation, especially in the cases in which the Hessian is not positive definite away from the optimum.

An analysis of the operator symbol of the Hessian for the Euler shape optimization problem conducted in \([17,18]\) suggests it is best to approximate the Hessian by an anisotropic operator in the chord and span directions, in which chordwise, the Hessian closely resembles a diffusive operator like the Laplacian. Because of this fact and the previous successes of gradient smoothing techniques \([19,20]\), we approximate the Hessian according to

\[ B \approx -\epsilon \Delta + I \]  

(19)

where \( \Delta \) is the tangential Laplace operator on the curved 2-D aircraft surface mesh and I is the identity. Further studies of shape Hessians for a variety of other fluid dynamics problems can be found in \([23,25]\]. During computation, the tangential Laplacian is computed as described in \([34]\). The effects of this preconditioning on the drag gradient of the Onera M6 wing can be seen in Fig. 2. The following test cases were chosen to demonstrate the applicability of the methodology discussed in the preceding sections to large-scale complex 3-D problems. Because a direct comparison to more classical approaches in such a high-dimensional search space is quite cumbersome, we would like to refer to \([22]\) for more details on how this methodology compares to Broyden–Fletcher–Goldfarb–Shanno (BFGS) update formulas and classical Hicks–Henne ansatz functions. Likewise, the approach discussed here is also very well applicable in robust or multipoint design problems, and more details can be found in \([35]\).

IV. Onera M6

A. Overview of the Test Case

The first problem under consideration is the shape optimization of an Onera M6 wing at a cruise condition of Mach 0.83 and a 3.01 deg angle of attack, which are being held constant during the optimization. In this configuration, the computed lift coefficient is \( C_L = 0.2762 \), which is to be kept constant. The initial drag coefficient is computed as \( C_D = 0.01058 \). The primal and adjoint

![Fig. 2 Effects of the Laplace–Beltrami preconditioner (19) on the drag gradient for \epsilon = 0, 10^{-2}, 10^{-1}, 10^{0}\) on the Onera M6 wing.](image-url)
flow states are computed using vertex-centered finite volumes with the DLR flow solver TAU. More details and validations of TAU can be found in [36,37]. The mesh consists of 18,285 surface mesh nodes. Because the volume mesh is perturbed using the algebraic mesh deformation tool that is part of the TAU suite, the planform had to be fixed as otherwise the deformation tool was very often unable to make volume meshes of satisfying quality. Because of this reason, the surface mesh nodes were also moved in the $x_3$ direction only, meaning the gradient was evaluated according to Eq. (15) for a movement of each node in direction of the normal in the current optimization iteration at that node. However, before any actual mesh deformation is applied, there is a projection of this gradient with respect to a movement in the $x_3$ direction only. We therefore expect a better performance of Eq. (15) when more sophisticated mesh deformation tools are available. Fixing the planform reduces the effective number of unknowns for the shape to 16,792. Because most inviscid meshes feature a numerically sharp trailing edge with potentially infinite curvature, any possible problems stemming from this point are therefore also circumvented. Counting the field nodes also, there are 541,980 unknowns for the Euler fluid state.

**B. Gradient Validation**

Before the actual nonparametric wing optimization is conducted, we first confirm the accuracy of the surface gradient by the means of a comparison with the classical gradient Eq. (1) evaluated using TAU adjoint and finite differences for the respective partial derivatives. Because all 16,792 nonplanform relevant nodes need to be perturbed in the $x_3$ axis direction, an equal number of deformed meshes and flow solver residual calculations must be made. The whole process required 2 days and 7 h on a 3 GHz Pentium IV processor. To ease the computational burden, one-sided finite differences were used with a perturbation of $\epsilon = 10^{-3}$, multiplied with an approximation of inscribed circle diameter of the associated dual volume face.

Cuts of the drag gradient for different spanwise positions are shown in Fig. 3, which uses the same convention as the $C_P$ plots do, meaning the gradients for the upper and lower sides are plotted within the same figure. We found the results obtained using the surface shape derivative to be surprisingly accurate, especially given the fact that the derivation and benefits of the surface shape derivatives heavily rely on the exploitation of the nature of shape optimization problems in the continuous setting. Although these analytic considerations are completely independent of the discrete mesh deformation chain, there are next to no discrepancies between the surface gradient formulation and the finite differences for all areas of the wing, except maybe for the singularity of the leading-edge stagnation point in this inviscid flow.

**C. Results and Optimized Wing**

Although the lift constraint itself already implies a certain internal volume, the total volume is to be kept constant as a very rough
Fig. 4 Initial and optimized Onera M6 wings.

Fig. 5 Pressure distributions across airfoil cuts for the initial and optimized Onera M6 wings.
measure of preserving structural strength in case the shock is strong enough that the optimizer attempts thinning the wing excessively. The volume constraint is treated explicitly by the optimizer, and its gradient has been derived exactly on the discrete level. For the Laplace–Beltrami Hessian approximation, a value of $\epsilon = 0.0136$ is used.

The planform pressure distribution on the upper surfaces of the initial and optimized wings is shown in Fig. 4. Respective CP distributions are shown in Fig. 5 and the initial and optimized airfoil cuts are shown in Fig. 6. The optimized Onera M6 wing has a drag coefficient of $C_D = 0.0070$, which corresponds to an improvement of 28.47%. Also, the optimized wing has a lift coefficient of $C_L = 0.2723$, which means the lift was preserved up to 1.41%. The optimal solutions were found after 70 one-shot iterations with 10 inner iterations in each adjoint flow solver and 20 iterations in the primal. The actual design update was conducted as given by Eq. (18), which also contains the typical one-shot step dampening. Looking at the pressure distributions in Fig. 5, one can see that the optimized wing is indeed shock free over the complete span. Because the cross-sectional thickness of each airfoil was not fixed but only the total volume of the wing, one can see that the optimized wing has become thinner at the root and thicker towards the tip, which is less than prefect from a structural point of view. However, there was no mathematical constraint to account for structural requirements, making this acceptable for the purpose of the present work.

The convergence of the drag objective during the optimization can be seen in Fig. 7.

V. Very Efficient Large Aircraft Blended-Wing–Body

A. Standard Mesh

The second test is the optimization of the “Very Efficient Large Aircraft (VELA),” a blended-wing–body concept [38]. A traditional optimization of this aircraft can also be found in [39]. The tetrahedral mesh consists of 115,673 surface mesh points, of which 113,956 remain as design unknowns after fixing the planform. The mesh has a total number of 1,061,433 nodes in the field. The flow and both the

![Fig. 6 Airfoil cuts for the initial and optimized Onera M6 wings.](image)

![Fig. 7 Convergence of the objective function for the M6 test case.](image)
adjoints for lift and drag are again computed using the DLR flow solver TAU. As in the Onera M6 case, the gradient is again computed according to Eq. (15). After one update of the aircraft surface, the new volume mesh is created by deforming the mesh from the previous iteration using the algebraic mesh deformation tool that is part of the TAU software. Because of this tool having difficulties in deforming the volume mesh for perturbations of the type \( V_{i,k}(x) = n(x) \delta_{i,k} \) with linear interpolation in between points, the gradient is again

Fig. 8 Initial and optimized VELA aircraft.

Fig. 9 Pressure distributions across airfoil cuts for the initial and optimized VELA aircraft.
projected for a movement in the $x_3$ direction only. For the smoothing preconditioner, $\epsilon = 10$ is used.

The initial and optimized aircraft are shown in Fig. 8. Some $C_P$ plots are shown in Fig. 9, whereas the respective airfoil cross sections are presented in Fig. 10. At a 1.8 deg angle of attack and a cruise condition of Mach 0.85, the initial configuration has a drag coefficient of $C_D = 0.004770$ and a lift coefficient of $C_L = 0.1787$ (these values are unusually low due to the high reference area of the wing–body planform). Both the angle of attack and the Mach number are being held constant during the optimization. The optimal solution is found after 151 one-shot iteration steps with 20 inner iterations for each of the two adjoint solvers and 40 inner iterations for the primal flow solver. The optimized design has a drag coefficient of $C_D = 0.003342$ and a lift coefficient of $C_L = 0.1775$. In total, drag was reduced by 29.93%, whereas lift was almost precisely kept with a relative loss of only 0.67%. The total amount of time needed for each shape update is around 390 s including the evaluation of the shape derivative for all 113,956 design unknowns and one solution of the surface Laplace gradient smoothing operator. The precise timings are shown in Table 1. Note that the timings do not exactly add up to 390, as some servicing steps and the solve with the surface Laplacian are not accounted for. The flow and adjoint solvers were running on four cores of an AMD Phenom II 2.8 GHz PC, whereas the other steps were computed on one core only. A full convergence to a TAU residual of $10^{-9}$ of the primal solution without optimization requires 1552 iterations or roughly 66 min. Thus, given 151 optimization steps, one can see that the optimal shape for 113,956 design parameters is found in as little as 15 equivalent flow solutions.

### B. Analysis of the Optimized VELA

Looking at Fig. 8 and the $C_P$ plots in Fig. 9, one can see that the shock wave on the upper and lower sides of the wing could be removed for almost the whole span, whereas the pressure distribution of the fuselage is also somewhat improved. Observing the airfoil cuts in Fig. 10, one can see that during optimization the twist of the wing fuselage near the root has slightly decreased, whereas the twist of the wing near the tip as increased, even though there were no design

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Time spent during each VELA optimization step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Time in seconds</td>
</tr>
<tr>
<td>Volume mesh deformation</td>
<td>36</td>
</tr>
<tr>
<td>Dual mesh construction and partitioning</td>
<td>49</td>
</tr>
<tr>
<td>Curvature computation</td>
<td>4</td>
</tr>
<tr>
<td>Primal flow solver (40 iterations)</td>
<td>101</td>
</tr>
<tr>
<td>Adjoint flow solver (drag, 20 iterations)</td>
<td>57</td>
</tr>
<tr>
<td>Adjoint flow solver (lift, 20 iterations)</td>
<td>57</td>
</tr>
<tr>
<td>Shape derivative evaluation</td>
<td>26</td>
</tr>
<tr>
<td>Derivative of volume constraint</td>
<td>4</td>
</tr>
</tbody>
</table>
parameters specifically controlling the twist. However, with such a fine parameterization available, the optimizer can achieve a shock-free or almost shock-free aircraft geometry that is very close to the original layout, which appears to be very beneficial for the actual design process, because, usually, larger deformations for improving aerodynamics are often problematic from a structural point of view. This can also be seen when comparing this optimal solution with the one from [39], in which the optimized wing has a substantially higher sweep and aspect ratio, whereas here, much less dramatic changes have resulted in an optimal shape with a comparable performance increase. Thus, there are potential benefits if the actual design process of the aircraft is already in a more advanced state and major changes can no longer be incorporated. The actual convergence of the objective during the optimization can be seen in Fig. 11.

Furthermore, the spanwise lift distribution is shown in Fig. 12. One can see that the optimized distribution gets closer to the elliptical, sweep and aspect ratio. Fig. 13 shows airfoil cuts for the optimized aircraft based on original and eight-times-adapted meshes.
which means that part of the drag reduction is also due to reducing the induced drag. The graphy axis is the local lift coefficient multiplied with 0.0136

C. Mesh Refinement

Next, we want to check how much the optimized shape depends on the mesh fineness, for the same initial geometry. To this end, we perform the same optimization on a mesh with eight times the number of points, derived by pressure-based adaptation from the original mesh. This adapted mesh contains 462,238 surface mesh points, giving 460,517 design unknowns, and 8,110,568 points in total. At the same angle of attack, the initial drag coefficient on this mesh is $C_D = 0.004378$, which is 8.22% less than on the original mesh; the optimized value is $C_D = 0.002862$ or 14.36% less than that of the optimized original mesh. In spite of these significant differences, the optimized shapes on the original and refined mesh are very close (Fig. 13), much closer than either is to the original shape.

The flow and adjoint solvers for optimization on the refined mesh were running on 4 AMD Opteron 2384 2.7 GHz cluster nodes, each with 8 cores, for 32 processes in total. Two hundred one-shot iteration steps were performed, with each taking about 2550 s. Twice the amount of iterations, 80 primal and 40 for each of the adjoint solvers, were used compared to the original mesh. The smoothing factor was increased to $\epsilon = 40$.

VI. Outlook: Viscous Fluids

The extension of the shape optimization technique presented here also include viscosity is straightforward, and preliminary theoretical studies for the compressible laminar Navier–Stokes equations can be found in [21]. The actual application to viscous compressible fluids is part of current research. The situation becomes somewhat more delicate when turbulent flows are considered. Most of the standard turbulence models have elements for which a formal derivation of the continuous adjoint equation or the partial shape derivative is not straightforward. A good example would be the wall-distance functions of the Spalart–Allmaras turbulence model or some of the boundary conditions in the $k$–$\epsilon$ and the $k$–$\omega$ model. Although these difficulties can easily be circumvented by considering a frozen eddy viscosity, there are also reports of successful uses of analytically adjointed turbulence models [40]. Given the fact that, for example, the partial derivative of the wall-distance functions or even the complete turbulent flow solver could also be treated efficiently on a discrete level using, e.g., algorithmic differentiation [41], the shape optimization method presented here appears to be also applicable to turbulent flows, although the derivation is probably not straightforward and might require some form of hybridization.

VII. Conclusions

Large-scale aerodynamic shape optimization for the compressible Euler equations in three dimensions is considered. By using the Hadamard form of the shape gradient, a sensitivity information for the aerodynamic forces can be computed extremely efficiently, such that each surface mesh node position can be used as a design parameter. Being an analytic exact surface expression, the partial derivatives of the mesh deformation tool and the mesh sensitivity Jacobians are not required. Using these shape gradients as the reduced gradients in a one-shot optimization strategy creates a shape one-shot method for which the Hessian is approximated using the surface or tangential Laplace operator. Feasibility of the method for large-scale aerodynamic problems is shown through the optimization of an Onera M6 wing with 16,792 unknowns of the shape and the optimization of the Very Efficient Large Aircraft blended-wing–body concept aircraft using 113,956 coarse mesh and 460,517 fine mesh surface node positions as design parameters.

Acknowledgments

The authors wish to thank Markus Widhalm at the DLR, German Aerospace Center, Braunschweig, for the discussions and the VELA test case.

References


J. Martins
Associate Editor
EFFICIENT NUMERICAL SOLUTION OF GEOMETRIC INVERSE PROBLEMS INVOLVING MAXWELL’S EQUATIONS USING SHAPE DERIVATIVES AND AUTOMATIC CODE GENERATION

STEPHAN SCHMIDT\textsuperscript{1}, MARIA SCHÜTTE\textsuperscript{2}, AND ANDREA WALThER\textsuperscript{1}

Abstract. We propose a novel approach using shape derivatives to solve sharp interface geometric inverse optimization problems governed by Maxwell’s equations. A tracking-type target functional determines the distance between the solution of a 3D time-dependent Maxwell problem and given measured data in an $L_2$-norm. Minimization is conducted using $H^1$-gradient information based on shape derivatives, which is related to the shape Hessian of the problem regularization. We describe the underlying formulas and the derivation of appropriate upwind fluxes and arrive at shape gradients for general tracking-type objectives and conservation laws. Subsequently, an explicit boundary gradient formulation based on variational forms is given for the problem at hand. Using such variational forms as domain specific programming languages, the FEniCS environment can then automatically generate the solvers, leading to structure exploiting data efficient transient adjoints. Checkpointing strategies are not necessary. Numerical results of up to $1.2 \cdot 10^9$ state unknowns demonstrate the practicability of the proposed approach.

Key words. geometric inverse problems, large-scale problems, code generation, Maxwell’s equations, shape optimization

AMS subject classifications. 65M32, 65M25, 65K10, 68N19, 35Q61

DOI. 10.1137/16M110602X

1. Introduction. In many research fields, such as biology, medical diagnostics, aircraft design, material testing, earth observation, and remote sensing, the noninvasive analysis to determine the location and/or structure of a hidden object inside a given material becomes more and more important. In some of these applications, depending on the material and imperfection size, radar measurements, X-rays, or a study of the general electromagnetic reflection spectra are the appropriate methods of choice. In any case, a problem-dependent target functional constrained by Maxwell’s equations has to be solved, which can also be interpreted as an inverse problem. In contrast to standard fluoroscopy and medical imaging, where the object being study is placed between sender and receiver, we are primarily interested in a remote sensing approach, that is, a reconstruction mainly based on reflections. Working with a few radiation sources that act as both emitter and receiver, this has the benefit of greatly reducing the experimental setup cost and is essential in situations where a receiver cannot be placed behind the object of interest.

Quite often the time-dependence of Maxwell’s equations has been eliminated using time-harmonic formulations; see, e.g., [42, 43]. This includes also studies on shape sensitivities and their existence; material derivatives are considered in [32], while local derivatives for electromagnetic transmission problems are studied in [13]. Furthermore, [7] discusses shape differentiability in electromagnetic scattering without...
restricting to particular classes of boundary deformations. Common to the abovementioned works on time-harmonic shape differentiability of Maxwell’s equations are restrictions on the applicable boundary conditions. Shape differentiability for perfect conductors was shown in [20] for a setting post discretization involving Raviart–Thomas elements. Furthermore, [18] considers differentiability of domain mappings with application to perfect conductors in star-shaped domains. Scattering of perfect conductors via far-field operators is considered in [21].

However, in a practical setup, one usually faces the propagation of broadband signals and limited abilities to measure the real and imaginary parts of the received signal, resulting in ill-posedness of the considered problem. Further difficulties arise when an inclusion has to be determined within a nonstationary process, e.g., within working machinery or within medical imaging problems. Therefore, a solution of the inverse problem in frequency domain may not be sufficient [1, 41]. Quite recently, there have also appeared contributions to the time-dependent nonshape optimal control of magnetic fields; see, e.g., [27] for optimal control of magnetization processes governed by Maxwell’s equations in eddy current formulation. Furthermore, a shape sensitivity analysis for time-domain Maxwell’s equations in a magnetic vector potential formulation was performed in [16], and the existence of local and material shape derivatives was studied in [4]. In comparison to the time-harmonic considerations above, the aim of the present work is to create a very efficient optimization software to solve geometric inverse electromagnetic scattering problems in three-dimensional time-dependent Maxwell’s equations with sharp interfaces. To this end, we consider a formal derivation of shape derivatives in the general case of time-dependent conservation laws with mixed boundary conditions, where the domain is the unknown to be found. Another novelty of our method is the exploitation of data reduction from using the boundary formulation within a time-dependent case, enabling us to solve problems of up to $1.2 \cdot 10^9$ time-dependent state unknowns without checkpointing.

With respect to time-dependent simulations, the state-of-the-art approach for large-scale applications is to transform the shape identification into a parameter identification problem. Typically, the “forward” or simulation problem is solved using finite differences in time domain (FDTD) methods on structured grids [41]. Then, the location of objects may be obtained by identifying the material distribution of specific electromagnetic properties over the whole computational domain; see, e.g., [22]. This method is also very successfully applied to antenna design problems [11]. Alternatives include the usage of appropriate indicator functions or phase field methods [10]. Common to all of the above approaches is a representation of the unknown domain or inclusion on a nonmatching grid via a smooth interface. Furthermore, this approach transforms the shape optimization problem into a problem of finding a distributed parameter, removing many possibilities for structure exploitation, such as using boundary representations.

Because we want to identify the actual structure of a hidden object, we conduct a numerical shape sensitivity investigation of the inclusion, allowing direct sharp interface manipulation during an optimization process, thereby operating with a fully resolved boundary on matching meshes. For this purpose, we apply discontinuous Galerkin (DG) methods [12] since they have the ability to operate on unstructured grids and their stabilization is very much in line with our modeling approach of the optimization problem. In particular, stabilization, interelement connectivity, and implementation of the desired boundary conditions will all be taken care of by considering the characteristic variables and upwind directions of Maxwell’s equations. Conveniently, these characteristic variables will also be crucial for deriving the respective
goal function for our optimization problem, an approach previously used in [38] for noninverse antenna design problems and in [11] for filtered and smoothed interfaces.

The main part of this paper is the proposition of a novel approach to solve the inverse Maxwell problem based on shape optimization with automatic code generation and high-performance computing techniques. One important part of this new approach is the introduction of shape derivatives for general conservation laws in variational form, which are later specialized to Maxwell’s equations. On the one hand, this allows us to directly interface with the FEniCS framework [24, 25], as variational forms are used as a domain specific programming language there. The capabilities of the FEniCS framework with respect to computational geometry is enhanced using the FEMorph shape optimization toolbox.\footnote{http://fenicsproject.org/applications/} Thus, the forward and adjoint integration of the state equation is realized using a tailor made DG Maxwell solver. The solver is validated against a closed form solution of Maxwell’s equations. On the other hand, with respect to data management within time-dependent adjoint solvers, a boundary representation of the shape derivative is quite desirable, enabling a memory-reduced adjoint computation. This formulation is typically called the Hadamard representation of the shape gradient [39]. With the intention of solving problems with primal trajectories of more than \(1.2 \cdot 10^9\) unknowns, this data reduction is essential for us. However, using a volume formulation or a hybrid approach with deformations reduced to tubular neighborhoods can be beneficial, especially with respect to numerical consistency; see [15] for more details. Furthermore, analytical results concerning existence and uniqueness of local shape derivatives of Maxwell’s equations for a certain set of boundary conditions are stated in [4, 5, 7], further motivating the use of boundary expressions here. We also discuss how using \(H^1\)-descent directions is related to approximative shape-Newton schemes for such problems.

The paper is structured as follows. Section 2 contains the general formulation for the specific kind of problems which we want to consider, the general form of the shape gradient, and the form adapted for Maxwell’s equations. A description of the three-dimensional time-dependent Maxwell’s equations is also given in this section. In section 3 an expression for the upwind fluxes is determined. The study of the boundary conditions can be found in section 4. The specific formulation of the target functional and, subsequently, the derivation of the shape gradient, are given in section 5. As a result, we get a representation of the shape gradient in the Hadamard sense, that is, a boundary representation. Despite having an explicit form of the gradient, several challenges in computing this gradient occur. After a verification of the proposed forward solver, solutions to these problems as well as numerical results are described in section 6. Two applications examples, the direct measurement of the near field and an example with antenna interaction, are considered in section 8. Finally, conclusions and a short outlook are given in section 9.

2. Problem formulation. We consider a PDE-constrained optimization problem of the type \(\min_{(\varphi, \Gamma_{\text{inc}})} J(\Omega)\) with

\[
J(\Omega) := \frac{1}{2} \int_{\Gamma_{\text{obs}}} \int_{t_0}^{t_f} \| A(n)(\varphi(t, s) - \varphi_m(t, s)) \|^2 \, dt \, ds + \delta \int_{\Gamma_{\text{inc}}} 1 \, ds
\]

such that

\[
\begin{align*}
\varphi(t, x) + \text{div} \ F(\varphi(t, x)) & = 0 \quad \text{in} \ (t_0, t_f) \times \Omega, \\
\tilde{F}_b(\varphi(t, s), n(s)) & = 0 \quad \text{on} \ [t_0, t_f] \times \Gamma,
\end{align*}
\]
where $\Omega \subset \mathbb{R}^3$ is a bounded domain, $\Gamma_{\text{inc}} \subset \Gamma := \partial \Omega$ is the boundary of the inclusion, and $\Gamma_{\text{obs}} \subset \Gamma$ is the part of the boundary where observations, i.e., measurements, are taken. Furthermore, $A$ denotes a positive semidefinite matrix, which will be specified later, as a sensible choice of $A$ is directly related to the upwind stabilization approach, $n$ the outer normal, and $\varphi_m$ the given (experimental) data measured using the actual geometry $\Gamma_{\text{inc}}$ to be reconstructed. Furthermore, $\delta > 0$ is a regularization parameter, which scales the perimeter penalization. $F$ represents an operator, which does not contain any time or spatial derivatives. Thus, we consider purely hyperbolic problems. Appropriate boundary conditions are given by the boundary flux $F_b := F_b + g$, where $g$ describes the sources, i.e., the incoming signal. Throughout, we assume that $F$ and $F_b$ are smooth enough to allow differentiation with respect to $\varphi$ and that a shape differentiable solution to (2) exists. We refer the reader to [4, 5] for the precise regularity requirements, in particular when $F$ is used to describe Maxwell’s equations. The predominant situation of $F$ being nonlinear is encountered in fluid dynamics. We refer the reader to [28, 29, 30, 31] for a discussion on the respective regularity requirements when $F$ defines the compressible Navier–Stokes equations. The variable $t$ lies in the time interval $[t_0, t_f]$ with $0 \leq t_0 < t_f$. A typical outline of the geometry is given by Figure 1, where $\Gamma_{\text{inc}}$ denotes the inclusion to be identified. A wave is sent and its reflection is received on $\Gamma_{\text{obs}}$. Here, the wave first travels through a waveguide with boundary $\Gamma_{\text{wav}}$ into a typical horn antenna with boundary $\Gamma_{\text{ant}}$ before being reflected by $\Gamma_{\text{inc}}$. The latter boundaries are thought of as perfect conductors, i.e., made of metal. Later on, those perfect conductors will be unified as boundary $\Gamma_{E}$. The nonreflecting boundary of the remaining domain is denoted by $\Gamma_{\text{nr}}$.

**Fig. 1.** The general layout of the problem. A sender/receiver antenna radiates an inclusion and measures the reflection.

### 2.1. General formulation for the shape gradient.

We briefly recall the basic notion of shape differentiation here. For further details, see e.g., [8, 39]. Let $\Omega \subset D \subset \mathbb{R}^3$ and $V : D \rightarrow \mathbb{R}^3$ be a sufficiently smooth vector field describing a deformation of a domain $\Omega$ of class $C^k$, $k \geq 1$; that is,

$$ \Omega_{\epsilon} := \{ x + \epsilon V(x) : x \in \Omega \} $$
is a deformed domain for $\epsilon > 0$ and the points in $\Omega_\epsilon$ are given by $x_\epsilon := x + \epsilon V(x)$.

For optimization purposes, general target functionals like

$$J_1(\Omega_\epsilon) = \int_{\Omega_\epsilon} h_1(x_\epsilon, \epsilon) \, dx_\epsilon \quad \text{or} \quad J_2(\Omega_\epsilon) = \int_{\Gamma_\epsilon} h_2(s_\epsilon, \epsilon, n_\epsilon) \, ds_\epsilon$$

may be used, where $h_1$ and $h_2$ may depend on the deformation via $\epsilon$ and $n_\epsilon$ denotes the outward unit normal to the perturbed surface $\Gamma_\epsilon$.

Using standard shape differentiation techniques, the shape derivative of $J_1$ at $\Omega$ in direction $V$ is well known (cf. [39, eq. (2.168), p. 113] and given by

$$dJ_1(\Omega)[V] = \int_{\Gamma} \langle V, n \rangle h_1(s, 0) \, ds + \int_{\Omega} h_1'(x) \, dx,$$

provided $h_1$, $h_1'[V]$, and $\|\nabla_x h_1\|$ are all in $L^1(\Omega)$, where $h_1'$ denotes the local shape derivative of $h_1$. It is typically defined via the material derivative

$$dh_1[V](x) := \lim_{\epsilon \to 0^+} \frac{h_1(x_\epsilon, \epsilon) - h_1(x, 0)}{\epsilon},$$

$$dh_2[V](x) := \lim_{\epsilon \to 0^+} \frac{h_2(x_\epsilon, \epsilon, n_\epsilon) - h_2(x, 0, n)}{\epsilon}.$$

The local shape derivative is then recovered via

$$h_1'[V] := dh_1[V] - \langle V, \nabla_x h_1(., 0) \rangle \quad \text{and} \quad h_2'[V] := dh_2[V] - \langle V, \nabla_x h_2(., 0, n) \rangle,$$

where $\nabla_x := \nabla_1$ denotes the spatial derivative, i.e., the gradient operator with respect to the first argument. Where applicable, this coincides with the more intuitive definition of $h_1'[V](x) := \frac{\partial}{\partial \epsilon} h_1(x, 0)$, that is, the partial derivative with respect to the second argument.

The shape derivative for the special case where $J_2$ does not depend on $n_\epsilon$ can be found in [39, eq. (2.173), p. 116]. However, with respect to conservation laws, a dependency on the normal is very common, and the extension to the more general case of $J_2$ considered here can be found in [36, Lemma 3.3.14, p. 39] to be

$$dJ_2(\Omega)[V] = \int_{\Gamma} (V, n)[(\nabla h_2(s, n), n) + \kappa(h_2(s, n) - D_n h_2(s, n) \cdot n)$$

$$+ \text{div}_{\Gamma}(D_n h_2(s, n))] \, ds + \int_{\Gamma} h_2'(s, n) \, ds,$$

where we have omitted the argument 0 and the direction $V$ for brevity. Here, $\kappa := \text{div}_{\Gamma} n$ is twice the mean curvature of the 2D surface $\Gamma$. Furthermore, $D_n := D_3$ denotes the Jacobian with respect to the component where the normal $n$ acts, that is, the last argument of $h_2$, not to be confused with the spatial derivative in the normal direction. Very common for conservation laws is a linear dependency of $h_2$ on $n$, i.e., $h_2(s, n) = \langle h_2(s), n \rangle$. Then the curvature term involving $\kappa$ vanishes, and the remaining terms recreate the boundary trace of the “standard” divergence $\text{div}$ from its tangential cousin $\text{div}_\Gamma$, leading to

$$dJ_2(\Omega)[V] = \int_{\Gamma} (V, n) \text{div} h_2(s) \, ds + \int_{\Gamma} h_2'(s, n) \, ds.$$
The derivation of (7) from (6) can be found in the appendix of [38] or in [40]. The shape derivative of the perimeter penalization can directly be found as

$$\delta \int_{\Gamma_{inc}} \langle V, n \rangle \kappa \, ds,$$

due to (6). Hence, we obtain the shape linearization of a general conservation law to find \( \varphi' \) such that

$$0 = \int_{t_0}^{t_f} \int_{\Omega} \langle \xi, \varphi' \rangle - \langle F(\varphi), \nabla \xi \rangle \, dx \, dt + \int_{t_0}^{t_f} \int_{\Gamma} \xi \cdot F_b(\varphi, n_s) \, ds \, dt \forall \xi \in \Sigma(D).$$

We will specify the respective space \( \Sigma \) in section 3. We refer the reader to [3] for a discussion on the requirements on \( \Sigma \) such that Galerkin-type discretizations of material derivatives (4) and local shape derivatives (5) also exist therein. A possible requirement is that \( \Sigma \) be the class of global \( C^1 \)-functions, leading to a finite element discretization using Argyris elements, which violate the stability requirements of numerical schemes for conservation laws. We therefore follow the optimize-then-discretize approach and validate our numerical scheme against the continuous solution.

The first term on the right-hand side of (9) resembles \( J_{1s} \). Thus, a formal shape linearization is given by (3). Likewise, the second term can be linearized in accordance to (6). Hence, we obtain the shape linearization of a general conservation law to find \( \varphi' \) such that

$$0 = \int_{t_0}^{t_f} \int_{\Omega} \langle \xi, \varphi' \rangle - \langle F(\varphi), \nabla \xi \rangle \, dx \, dt + \int_{t_0}^{t_f} \int_{\Gamma} \xi \cdot F_b(\varphi, n_s) \, ds \, dt + \int_{t_0}^{t_f} \int_{\Gamma} \xi \cdot F_b(\varphi, n_n) \, ds \, dt \forall \xi \in \Sigma(D).$$

The variational formulation given by the last equation has to be fulfilled for all suitable test functions \( \xi \in \Sigma \). Hence, it must also hold for a specific \( \lambda \in \Sigma \), which will later serve as the adjoint state. Hence, we postulate the following adjoint equation by formally comparing the derivative of the target function (1), with respect to the PDE state, to the local derivatives of the shape differentiation process, arriving at the problem to find \( \lambda \) such that

$$\int_{t_0}^{t_f} \int_{\Omega} (\hat{\lambda}, \hat{\varphi}) - \langle \hat{\varphi}, D_{\varphi}^T F(\varphi) \nabla \lambda \rangle \, dx \, dt + \int_{t_0}^{t_f} \int_{\Gamma} (\hat{\varphi}, D_{\varphi}^T F_b(\varphi, n) \cdot \lambda) \, ds \, dt$$

$$+ \int_{t_0}^{t_f} \int_{\Gamma_{obs}} (A^T(n)A(n) \cdot (\varphi - \varphi_m), \hat{\varphi}) \, ds \, dt = 0 \quad \forall \hat{\varphi} \in \Sigma(D)$$

with the terminal condition \( \lambda(t_f) = 0 \). The inhomogeneity of the adjoint boundary condition, which stems from the partial derivative of the objective function, thus cannot be chosen arbitrarily. Hence, the choice of \( A(n) \) must be consistent with the
respective information flow as, for example, $\varphi - \varphi_m$ being within the kernel of $A^\top A$ would result in a source-term free adjoint. This is directly related to the information flow direction and will be discussed in more detail in section 3 in relation to upwinding. Since the adjoint formulation (10) requires a gradient operator on the adjoint state $\lambda$, we will call this the strong form adjoint. Likewise, the following equation:

$$
(11) 
\int_{t_0}^{t_f} \int_\Omega \left( -\lambda \dot{\varphi} + \langle \text{div}(D_\varphi F(\varphi)\dot{\varphi}), \lambda \rangle \right) dx dt + \int_{t_0}^{t_f} \int_\Gamma \langle \dot{\varphi}, D_\varphi^\top F_\varphi(\varphi,n) \cdot \lambda - D_\varphi^\top F(\varphi)\lambda \cdot n \rangle ds dt
$$

$$
+ \int_{t_0}^{t_f} \int_{\Gamma_{inc}} (A^\top(n)A(n) \cdot (\varphi - \varphi_m, \dot{\varphi}) ds dt = 0 \quad \forall \dot{\varphi} \in \Sigma(D)
$$

will be called the weak form of the adjoint equation.

The objective function (1) evaluates the mismatch only on a certain part of the boundary, which is constant and thus does not vary during the optimization process. Therefore, one has

$$
dJ(\Omega)[V] = \int_{t_0}^{t_f} \int_{\Gamma_{inc}} (A^\top(n)A(n) \cdot (\varphi - \varphi_m, V')) ds dt,
$$

where $dJ(\Omega)[V]$ represents the total derivative of the objective with respect to a change in $\Omega$. Now, one can combine (10) with the shape linearization of a general conservation law given above. If we assume that $F_b(\varphi,n) = F_b(\varphi) \cdot n$, i.e., $F_b(\varphi,n) \kappa$ vanishes. Provided that $\varphi$ and the respective derivatives can be continuously extended onto $\Gamma_{inc}$, the resulting expression simplifies to

$$
(12) 
dJ(\Omega)[V] = \int_{t_0}^{t_f} \int_{\Gamma_{inc}} \langle V, n \rangle \text{div}(D_n^\top(\lambda \cdot F_b(\varphi) \cdot n)) ds dt
$$

due to the state equation being fulfilled. It is worth noting that hence the shape derivative $\varphi'$ is associated with one particular test function $\varphi$, which typically necessitates higher regularity prerequisites [3]. Nevertheless, the memory reduction of having a boundary source term in the time-dependent adjoint (11) enables the creation of highly efficient software, which in return makes addressing the application problems presented herein possible on standard office workstations without checkpointing.

### 2.3. Adaptation for Maxwell’s equations

From now on, we assume that the state equation in the PDE-constrained optimization problem is given by the time-dependent Maxwell’s equations, i.e., one has

$$
\frac{\partial B(t,x)}{\partial t} = -\nabla \times E(t,x),
$$

$$
\frac{\partial D(t,x)}{\partial t} = \nabla \times H(t,x) - \sigma E,
$$

$$
\nabla \cdot D(t,x) = \rho,
$$

$$
\nabla \cdot B(t,x) = 0,
$$

where $x \in \Omega$, $B(t,.) \in (L^2(\Omega))^3$ describes the magnetic flux density, $E(t,.) \in (L^2(\Omega))^3$ the electric flux strength, $D(t,.)$ the time-dependent displacement of the
current, and \( H(t, \cdot) \) the magnetic flux strength. The related material equations for linear, isotropic, and nondispersive media are defined by

\[
D = \varepsilon E \quad \text{and} \quad B = \mu H,
\]

with \( \varepsilon = \varepsilon_0 \varepsilon_r \) and \( \mu = \mu_0 \mu_r \). Furthermore, \( \varepsilon_0 \) and \( \mu_0 \) are the constant vacuum permittivity and permeability and \( \varepsilon_r, \mu_r \) denote the relative, material-dependent counterparts, which may vary over the domain for different applications. The same is true for \( \sigma \), which describes the conductivity. Coupling Faraday’s law and Ampere’s law with the material equation yields

\[
\begin{align*}
\mu \frac{\partial H}{\partial t} &= -\nabla \times E, \\
\varepsilon \frac{\partial E}{\partial t} &= \nabla \times H - \sigma E.
\end{align*}
\]

To reuse the results obtained in subsection 2.1, Maxwell’s equations can be reformulated in the divergence form as in (2),

\[
(\mu, \varepsilon) \frac{\partial Q}{\partial t} + \text{div}(F(Q)) = S,
\]

\[
Q = (H_1, H_2, H_3, E_1, E_2, E_3)^T, \quad S = (0, 0, 0, \sigma E_1, \sigma E_2, \sigma E_3)^T,
\]

where \( Q \) contains the time-dependent variables and is interpreted as a 6-component column vector, and likewise for \( S \).

The Levi–Civita permutation symbol \( \epsilon_{ijk} \in \mathbb{R}^{3\times3 \times 3} \) is defined as +1 if \( i, j, k \) is an even permutation of \( (1, 2, 3) \) or as −1 if it is an odd permutation. In any other case, it is zero. Thus, one can also express the \( k \)th component of the cross product via

\[
(a \times b)_k = \left( \sum_{i,j=1}^{3} \epsilon_{ijk} a_i b_j \right)_k,
\]

and, with a slight abuse of notation, the curl operator can be expressed by

\[
(\nabla \times E)_k = \left( \sum_{i,j=1}^{3} \epsilon_{ijk} \frac{\partial}{\partial x_i} E_j \right)_k = \left( \sum_{i=1}^{3} \frac{\partial}{\partial x_i} \left( \sum_{j=1}^{3} \epsilon_{ijk} E_j \right) \right)_k = \left( \sum_{i=1}^{3} \frac{\partial}{\partial x_i} (R_i^T E) \right)_k,
\]

where \([R_i]_{jk} := [\epsilon_{ijk}]_{jk} \in \mathbb{R}^{3 \times 3} \) is interpreted as a matrix obtained by fixing the first index of the permutation symbol. We follow the definition of the divergence as in [24]; that is, the divergence of any nonscalar expression is defined as the contraction of the partial derivative over the last axis of the expression. Thus, to bring the above formulation of curl into divergence form, we define the following matrix \( M(E) \in \mathbb{R}^{3 \times 3} \):

\[
[M(E)]_{ji} := [R_i^T E]_j
\]

\[
\Rightarrow \text{div}(M(E)) = \text{div}([R_1^T E, R_2^T E, R_3^T E]) = \sum_{i=1}^{3} \frac{\partial}{\partial x_i} (R_i^T E) = \text{curl} E.
\]

Because \( R_i^T = -R_i \), \( F(Q) \in \mathbb{R}^{6 \times 3} \) in the conservative form of Maxwell’s equations is defined by

\[
F(Q) := \begin{bmatrix}
-R_1 E & -R_2 E & -R_3 E \\
R_1 H & R_2 H & R_3 H
\end{bmatrix} = \sum_{i=1}^{3} \begin{bmatrix}
0 & -R_i \\
R_i & 0
\end{bmatrix} \begin{bmatrix}
H \\
E
\end{bmatrix} \epsilon_i^T,
\]
where \( \hat{e}_i \) is the \( i \)th unit vector in \( \mathbb{R}^3 \). We will revisit this structure of \( F(Q) \) when we discuss boundary fluxes and upwinding, in particular during the definition of the operator \( A_n \) in (19). For a complete problem formulation, appropriate boundary conditions containing also the incoming pulse must be defined. They are derived in detail in section 4.

3. Upwinding and transport of information. We have to distinguish between reflections and primary waves. A convenient way of accomplishing this is the consideration of an upwind approach. Therefore, we will choose our measurement operator \( A(n) \) to coincide with the positive definite part of the upwind splitting of the conservation law, which enables us to identify the information flow through facets. Since the consideration of upwinding is already necessary for our modeling approach and we seek resolved sharp interfaces, the discretization by a DG method suggests itself over the more predominant FDTD schemes. Another alternative to DG is to use \( H(\text{curl}) \) conforming spaces with tangential continuity. We refer the reader to [12, 23, 35] for a better overview of the respective discretizations.

Assuming there is a triangulation \( \mathcal{T}_h \) of \( \Omega \) of nonoverlapping open tetrahedrons \( K \), multiplication of the strong form Maxwell’s equations (13) by functions \( (v, w) \), integration over \( \Omega \), and conducting per-element integration by parts of the curl operator leads to

\[
\int_{t_0}^{t_f} \int_K \left\langle v, \frac{\partial H}{\partial t} \right\rangle + \left\langle E, 1/\mu \, \text{curl} \, v \right\rangle + \left\langle w, \frac{\partial E}{\partial t} \right\rangle - \left\langle H, 1/\varepsilon \, \text{curl} \, w \right\rangle + \frac{\sigma}{\varepsilon} \langle w, E \rangle \, dx \, dt \\
- \int_{t_0}^{t_f} \int_{\partial K} \frac{1}{\mu} \langle v, E \times n \rangle - \frac{1}{\varepsilon} \langle w, H \times n \rangle \, ds \, dt = 0,
\]

where \( \partial K \) denotes the facets of \( K \). To obtain the general form as stated in section 2, we first define

\[
A_n := \begin{bmatrix}
0 & -n_z & n_y \\
-n_z & 0 & -n_x \\
-n_y & n_x & 0
\end{bmatrix}
\]

and use the above definition likewise for different components, such as \( A_E \) or \( A_H \). It is worth noting that for any vector \( v \), one has

\[ A_n v = n \times v = -v \times n = A_{-n} n = -A_n n = -A_n^T v. \]

Furthermore, this formulation of Maxwell’s equations is related to the flux formulation in (17) via the property \( A_n = R_1 n_x \hat{e}_1 + R_2 n_y \hat{e}_2 + R_3 n_z \hat{e}_3 \). Then (18) can be expressed in line with the general expression (9):

\[
\int_{t_0}^{t_f} \int_K \left\langle v, \frac{\partial H}{\partial t} \right\rangle + \frac{1}{\mu} \langle A_E, \nabla v \rangle + \left\langle w, \frac{\partial E}{\partial t} \right\rangle - \frac{1}{\varepsilon} \langle A_H, \nabla w \rangle + \frac{\sigma}{\varepsilon} \langle w, E \rangle \, dx \, dt \\
- \int_{t_0}^{t_f} \int_S (v, w) F_S(H, E, n) \, ds \, dt = 0,
\]

where

\[ F_S(H, E, n) = A \begin{pmatrix}
H \\
E
\end{pmatrix} \quad \text{with} \quad A = \begin{bmatrix}
0 & -\frac{1}{\mu} A_n \\
\frac{1}{\varepsilon} A_n & 0
\end{bmatrix} \]
and the inner product for matrices \((A_E, \nabla v)\) is interpreted as the componentwise sum.

To derive the upwind scheme, the matrix \(\tilde{A}\) must now be split in a positive and a negative semidefinite part. For this purpose, the diagonalization is a convenient tool. For the projection onto the space of eigenvectors of \(\tilde{A}\), one has the eigenvalues

\[
c_1 = c_2 = 0, \quad c_3 = c_4 = -\frac{1}{\sqrt{\mu \varepsilon}}, \quad c_5 = c_6 = \frac{1}{\sqrt{\mu \varepsilon}}
\]

of \(\tilde{A}\) with the eigenvectors

\[
v_1 = (0, 0, 0, n_x, n_y, n_z)^\top, \quad v_2 = (n_x, n_y, n_z, 0, 0, 0)^\top,
\]

\[
v_3 = (-\sqrt{\frac{\mu}{\varepsilon}} n_x n_y, \sqrt{\frac{\mu}{\varepsilon}} (n_x^2 + n_y^2), -\sqrt{\frac{\mu}{\varepsilon}} n_y n_z, n_x, 0, 0)^\top,
\]

\[
v_4 = (\sqrt{\frac{\mu}{\varepsilon}} n_x n_z, \sqrt{\frac{\mu}{\varepsilon}} n_y n_z, -\sqrt{\frac{\mu}{\varepsilon}} (n_x^2 + n_y^2), -n_y, n_x, 0)^\top,
\]

\[
v_5 = (\sqrt{\frac{\mu}{\varepsilon}} n_x n_y, -\sqrt{\frac{\mu}{\varepsilon}} (n_x^2 + n_y^2), \sqrt{\frac{\mu}{\varepsilon}} n_y n_z, n_x, 0, 0)^\top,
\]

\[
v_6 = (-\sqrt{\frac{\mu}{\varepsilon}} n_x n_z, -\sqrt{\frac{\mu}{\varepsilon}} n_y n_z, \sqrt{\frac{\mu}{\varepsilon}} (n_x^2 + n_y^2), -n_y, n_x, 0)^\top.
\]

To simplify notation, throughout we will use the abbreviations

\[
Y = \sqrt{\frac{\mu}{\varepsilon}}, \quad Z = \sqrt{\frac{\mu}{\varepsilon}}, \quad \text{and} \quad c = \frac{1}{\sqrt{\mu \varepsilon}}.
\]

Using the similarity transformation \(\Lambda = P^{-1} \tilde{A} P\), where

\[
P := [v_5, v_3, v_1, v_6, v_4, v_2]
\]

\[
P = \begin{bmatrix}
  n_x n_y Y & -n_x n_y Y & 0 & -n_x n_z Y & n_x n_z Y & n_x \\
  -(n_x^2 + n_y^2) Y & (n_x^2 + n_y^2) Y & 0 & -n_y n_z Y & n_y n_z Y & n_y \\
  n_y n_z Y & -n_y n_z Y & 0 & (n_x^2 + n_y^2) Y & -(n_x^2 + n_y^2) Y & n_z \\
  -n_z & n_z & n_x & -n_y & -n_y & 0 \\
  0 & n_y & n_x & n_z & n_x & 0 \\
  n_x & n_z & 0 & 0 & 0 & 0
\end{bmatrix},
\]

one gets the diagonal representation \(\Lambda = \text{diag}(c, -c, 0, c, -c, 0)\); i.e., the diagonal elements of \(\Lambda\) are given by the corresponding eigenvalues. Defining the new operators \(A^\pm = P A^\pm P^{-1}\), where \(A^+ = \text{diag}(\max(c, 0))\) and \(A^- = \text{diag}(\min(c, 0))\), it is easy to see that \(\tilde{A}\) can be split into

\[
A^+ = \frac{1}{2} \begin{bmatrix}
  \tilde{A}_n c & A_n Y c \\
  -A_n Z c & \tilde{A}_n c
\end{bmatrix}, \quad A^- = \frac{1}{2} \begin{bmatrix}
  -\tilde{A}_n c & A_n Y c \\
  -A_n Z c & -\tilde{A}_n c
\end{bmatrix},
\]

i.e., a positive semidefinite part and a negative semidefinite part, where

\[
\tilde{A}_n := \begin{bmatrix}
  n_x^2 + n_y^2 & -n_x n_y & -n_x n_z \\
  -n_x n_y & n_x^2 + n_y^2 & -n_y n_z \\
  -n_x n_z & -n_y n_z & n_x^2 + n_y^2
\end{bmatrix}
\]

and \(A_n\) is defined as above. Furthermore, \(\tilde{A}_n v = n \times (v \times n) = -A_n^2 v\). In order to solve the problem in the entire domain, (21) is now summed over all elements \(K\) and
child facets $S$ in $\mathcal{T}$. Not having any interelement continuity in $\tilde{\Sigma}$, we can introduce the local and remote state relative to the element $K$ and facet $S$ currently being processed. They are denoted by $^l$ and $^r$, and we specify a stabilized form of (21) by

$$F_S(H,E,n) = \tilde{A} \cdot \left( \begin{array}{c} H \\ E \end{array} \right) = \left[ \begin{array}{cc} 0 & -\frac{1}{\varepsilon} A_n \\ \frac{1}{\varepsilon} A_n & 0 \end{array} \right] \left( \begin{array}{c} H \\ E \end{array} \right) = (PA^{P^{-1}}) \cdot \left( \begin{array}{c} H \\ E \end{array} \right)$$

with

$$\tilde{F}_S(H,E,n) = \left( (PA^{P^{-1}}) \cdot \left( \begin{array}{c} H^l \\ E^l \end{array} \right) + (PA^{P^{-1}}) \cdot \left( \begin{array}{c} H^r \\ E^r \end{array} \right) \right)$$

$$= \frac{1}{2} \left[ \begin{array}{cc} -A_n c & A_n Yc \\ -A_n Zc & A_n c \end{array} \right] \cdot \left( \begin{array}{c} H^l \\ E^l \end{array} \right) + \left[ \begin{array}{cc} -A_n c & A_n Yc \\ -A_n Zc & -A_n c \end{array} \right] \cdot \left( \begin{array}{c} H^r \\ E^r \end{array} \right).$$

Finally, we arrive at the actual variational form being solved as our forward simulation problem. To this end, we introduce the space

$$S^0(S_b \Omega) = \{ \sigma \in L^2(T) : \sigma|_K \in \mathcal{P}_k(K) \forall K \in \mathcal{T}_h \}$$

and define $\tilde{S} := (S^0(S_b \Omega))^3 \times (S^0(S_b \Omega))^3$. It is worth noting that the space $S^0(S_b \Omega)$ does not imply continuity between adjacent elements $K_i$ and $K_j$. Our simulation problem is then to find $(H,E) \in \tilde{S}$ such that

$$\sum_{K \in \mathcal{T}_h} \int_{t_0}^{t_f} \int_K \left( v, \frac{\partial H}{\partial t} \right) + \frac{1}{\mu} \left( A_E, \nabla v \right) + \left( w, \frac{\partial E}{\partial t} \right) - \frac{1}{\varepsilon} \left( A_H, \nabla w \right) + \frac{\sigma}{\varepsilon} (w,E) \, dx \, dt$$

$$- \int_{t_0}^{t_f} \int_{\partial K} (v,w) \tilde{F}_S(H,E,n) \, ds \, dt = 0 \quad \forall (v,w) \in \tilde{S},$$

where $A_H$ and $A_E$ are defined as in (19). Finally, the respective time derivatives are treated by the implicit midpoint rule following the method of lines approach. As such, they are not part of the weak formulation. The expression $\tilde{F}_S(H,E,n)$ of the upwind scheme will now be modified to create expressions for $F_b(H,E,n)$, specifying the boundary conditions holding on the respective surfaces.

4. Boundary conditions. Up to now, $\Gamma$ comprises all physical boundaries, which are now split into parts $\Gamma_E$, where the $E$ field is assumed to be orthogonal, and $\Gamma_H$ with orthogonal $H$ field, $\Gamma_{ob}$, defining the entering pulse and also the area where the measurements take place, and $\Gamma_{nr}$ for the remaining absorbing, i.e., nonreflecting, boundary. Symmetry boundaries are likewise given by $\Gamma_E$ and $\Gamma_H$. Metallic surfaces, such as the considered obstacle $\Gamma_{inc}$, are modeled as perfect conductors and are also contained in $\Gamma_E$. Thus, with respect to the geometry shown in Figure 1, we have $\Gamma_E = \Gamma_{inc} \cup \Gamma_{ant} \cup \Gamma_{wave}$, as all of these boundaries are thought of as perfectly conducting metals.

First, for $\Gamma_E$ we exploit that $A_n E = 0$, meaning the electric field must not have a tangential component. This yields

$$F_b^E(H,E,n) = \frac{1}{2} \left[ \begin{array}{cc} -A_n c & A_n Yc \\ -A_n Zc & -A_n c \end{array} \right] \cdot \left( \begin{array}{c} H \\ E \end{array} \right) + \frac{1}{2} \left[ \begin{array}{cc} -A_n c & A_n Yc \\ -A_n Zc & -A_n c \end{array} \right] \cdot \left( \begin{array}{c} H \\ E \end{array} \right)$$

$$= \left[ \begin{array}{cc} 0 & 0 \\ -A_n Zc & 0 \end{array} \right] \cdot \left( \begin{array}{c} H \\ E \end{array} \right) = \left( \begin{array}{c} 0 \\ -A_n Zc H \end{array} \right).$$
Likewise, one obtains for $\Gamma_H$ with $A_n H = 0$

$$F^H_b (H, E, n) = \begin{bmatrix} 0 & A_n Yc \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} H \\ E \end{bmatrix} = \begin{bmatrix} A_n Yc E \\ 0 \end{bmatrix}. $$

For $\Gamma_{inc}$, one can define the following boundary conditions:

$$F^{inc}_b (H, E, n) = \frac{1}{2} \begin{bmatrix} \tilde{A}_c & A_n Yc \\ -A_n Zc & A_n c \end{bmatrix} \cdot \begin{bmatrix} H \\ E \end{bmatrix} + \frac{1}{2} \begin{bmatrix} -\tilde{A}_c & A_n Yc \\ -A_n Zc & -A_n c \end{bmatrix} \cdot \begin{bmatrix} g_H \\ g_E \end{bmatrix}. $$

For nonreflective boundaries $\Gamma_{nr}$, the same flux holds with $g_H = g_E \equiv 0$.

**5. Derivation of the shape gradient.** Our goal is to determine the actual structure of a hidden object inside a given domain. Based on an initial geometry, the forward simulation is performed and the reflected electric and magnetic fields are simulated and measured on the observation boundary $\Gamma_{obs}$. To get an analytical expression for the simulated field values, we only consider the part leaving the computational domain, which is the first term on the right-hand side of (23). Consequently, we obtain

$$A(n) := \frac{1}{2} \begin{bmatrix} \tilde{A}_c & A_n Yc \\ -A_n Zc & A_n c \end{bmatrix},$$

which forms the right-hand side in the adjoint equation (10). For the shape derivative without the regularization parameter $\delta$, we insert the function $F^E_b$ into the shape gradient (12), arriving at

$$dJ(\Omega)[V] = \int_{t_0}^{t_f} \int_{\Gamma_{inc}} (V, n) \text{div}(D^n (\lambda \cdot \tilde{F}_b (H, E) \cdot n)) \, ds \, dt$$

$$= \int_{t_0}^{t_f} \int_{\Gamma_{inc}} (V, n) \text{div}(D^n (\lambda E \cdot (-A_n Zc H))) \, ds \, dt = \int_{t_0}^{t_f} \int_{\Gamma_{inc}} (V, n) Zc \text{div}(\lambda E \times H) \, ds \, dt,$$

where we exploited that the differentiation of $A_n$ with respect to the normal $n$ yields the cross product.

**6. Implementations.**

6.1. The primal solver. To solve Maxwell’s equations, we employ a DG scheme on an unstructured tetrahedral mesh using the boundary and upwind stabilization fluxes as described in sections 3 and 4. Time-domain DG Maxwell solvers are a rapidly developing field with a variety of different formulations; see, e.g., [6, 12, 23]. As such, the verification of our DG solver is one of the major aspects of this section. Although the use of higher order test and ansatz functions on curved elements can be quite beneficial, we limit ourselves to straight-sided tetrahedrons. In order to get a high resolution of the domain, which directly corresponds to the resolution of our unknown domain to be found, we use first order test and ansatz functions and rather prefer spatially refined meshes over higher polynomial degrees, as otherwise we would have a low resolution of the geometry, unless we were to incorporate curved elements in both the simulation and shape optimization components. Time derivatives are discretized using the midpoint/trapezoidal rule, which for a linear problem such as Maxwell’s equations also coincides with Crank–Nicolson. Thus, our time integration
is symplectic, but we need to solve an implicit system for each time step, which is
done in distributed memory parallelism using GMRES preconditioned by a parallel
ILU solver, a functionality offered by PETSc and Hypre Euclid and based on the PILU
algorithm [17]. Using an implicit time integration has proven to feature additional
robustness when the mesh quality decreases slightly during optimization.

We first consider a verification of the primal solver. To the best of our knowledge,
there does not seem to be an accepted standard verification problem for time-domain
Maxwell solvers, and thus we revisit the approach from [26] with the same physical
dimensions and material parameters. There, the simulation of a scattering of a wave
entering from a vacuum into a zone of 2.5 times the vacuum permittivity is suggested
to be used for verification purposes. We consider a cuboid-shaped domain of nondi-

mensional size $5 \times 5 \times 200$. On this domain, we define a changing relative permittivity
$r$ via

$$
\epsilon_{r}^{\text{smooth}} = 1.75 + 0.75 \tanh(x_3 - 100);
$$

that is, we change the relative permittivity smoothly but rapidly at the center of
the domain to 2.5 the vacuum permittivity. We also consider the nonsmooth case
$\epsilon_{r}^{\text{nonsmooth}}$ to be 1.0 left of the middle of the domain and 2.5 right of the middle. Into
this domain we place an initial electromagnetic field in accordance to

$$
E = (E_0 \cdot \exp(-\alpha(x_3 - 40)^2), 0, 0)^T,
$$

$$
H = (0, H_0 \cdot \exp(-\alpha(x_3 - 40)^2), 0)^T,
$$

with $E_0 = 0.001$ and $H_0 = \frac{\sqrt{\mu_r}}{\sqrt{\mu_0}}E_0$ with constant relative permeability $\mu_r = 1$ and $\alpha =
0.01$. The situation is also shown in Figure 2. As time progresses, the electromagnetic
pulse will enter the zone of slower wave propagation, and part of the pulse will be
reflected back, creating one weaker right-going peak of magnitude $\tilde{E}_0$ and one left-
going reflection of magnitude $\tilde{E}_0$. According to [19, 26], there is a closed form solution
for the magnitude of $\tilde{E}_0$ and $E_0$ given by

$$
\frac{\tilde{E}_0}{E_0} = \frac{2}{\sqrt{\epsilon_r} + 1}, \quad \frac{\tilde{E}_0}{E_0} = \frac{\sqrt{\epsilon_r} - 1}{\sqrt{\epsilon_r} + 1},
$$

where $\epsilon_r$ denotes the value of $\epsilon_r$ to the right of the jump, in this case 2.5, and $\epsilon_r$
denotes the value to the left of the jump, here $\epsilon_r = 1.0$. In particular, the closed form
solution here leads to a relative magnitude of 0.77485 for the right-going peak and
0.22515 for the left-going peak. We conjecture that this verification is well suited for
constructing simulation tools for the reconstruction problem, as the degeneration of
signal quality due to numerical diffusivity is critical. The measured pulse, $\tilde{E}_0$ here,
will form the source term of the adjoint. Thus, a signal degeneration will directly effect
this source, and in the worst case the reflection could diffuse completely. This leads
to a zero adjoint forcing and consequently a zero gradient, suggesting false optimality.

The respective simulation results and a mesh refinement study are shown in
Table 1, where we obtained scalar values by integrating $E_z$ crosssectionally. In general,
one can see an improved behavior when using a jumping material coefficient.
Furthermore, we can see that for a comparatively coarse mesh of only 13,889 tetra-
hedra, the relative error of the reflection magnitude is already below 4%, which after
one global refinement step falls into the 1% regime. We conclude that our Maxwell
DG solver features reasonably little numerical diffusivity. The simulated results for
the finer mesh are shown in Figure 3.
Fig. 2. The verification setup with local permittivity $\varepsilon_r$ and initial pulse. Electric (yellow) and magnetic field (red) not to scale. (See online version for color.)

Table 1
Verification results with smoothly and nonsmoothly changing $\varepsilon_r$ for two different meshes.

<table>
<thead>
<tr>
<th>$\varepsilon_r$</th>
<th>Test $E_0/E_0$</th>
<th>Error</th>
<th>$E_0'/E_0'$</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>13,889</td>
<td>0.76851</td>
<td>0.82%</td>
<td>-0.21680</td>
</tr>
<tr>
<td>Smooth</td>
<td>13,889</td>
<td>0.76610</td>
<td>1.13%</td>
<td>-0.20684</td>
</tr>
<tr>
<td>Jump</td>
<td>84,749</td>
<td>0.77667</td>
<td>0.15%</td>
<td>-0.22772</td>
</tr>
<tr>
<td>Smooth</td>
<td>84,749</td>
<td>0.77524</td>
<td>0.05%</td>
<td>-0.21210</td>
</tr>
</tbody>
</table>

Fig. 3. Relative strength of the initial $E_x$ pulse (dashed line) and the simulation result at nondimensional time 100 (solid line).

6.2. The adjoint solver.

6.2.1. The adjoint equation in variational form. Following the same argumentation as when deriving (10) and taking into account the linear nature of the problem, it is easy to see that the volume part of the strong form adjoint is given by the problem to find $(\lambda_H, \lambda_E)$ such that

$$0 = \int_{t_0}^{t_f} \int_\Omega \left( \lambda_H \cdot \frac{\partial H}{\partial t} + \left( \hat{E} \cdot \frac{1}{\mu} \text{curl} \lambda_H \right) + \left( \lambda_E \cdot \frac{\partial \hat{E}}{\partial t} \right) - \left( \hat{H} \cdot \frac{1}{\varepsilon} \text{curl} \lambda_E \right) + \sigma \varepsilon \langle \lambda_E, \hat{E} \rangle \right) dx dt,$$

and consequently the volume component of the weak form of the adjoint is given by

$$0 = \int_{t_0}^{t_f} \int_\Omega \left( v \cdot \frac{\partial \lambda_H}{\partial t} + \left( \lambda_E \cdot \frac{1}{\varepsilon} \text{curl} v \right) + \left( w \cdot \frac{\partial \lambda_E}{\partial t} \right) - \left( \lambda_H \cdot \frac{1}{\mu} \text{curl} w \right) - \sigma \varepsilon \langle \lambda_E, w \rangle \right) dx dt,$$

(25)
where we have reintroduced the symbols \((v, w)\) for the test functions and flipped the signs due to time reversal. Comparing the time-corrected weak form of the adjoint (25) with its primal counterpart (18), one can see that the same solver can be used, provided some signs and coefficients \(\mu\) and \(\epsilon\) are swapped correctly.

We will now discuss the adjoint boundary fluxes exemplified by boundary \(\Gamma_E\). From (10) and (22) we see that the strong form boundary flux for \(\Gamma_E\) is given by

\[
0 = \int_0^{t_f} \int_{\Gamma_E} (\vec{H}, \vec{E}) \left[ \begin{array}{cc} 0 & A_n Zc \\ 0 & 0 \end{array} \right] \cdot \left( \begin{array}{c} \lambda_H \\ \lambda_E \end{array} \right) \, dsdt,
\]

where we have used the antisymmetry of \(A_n\). Although this suggests at first glance that electric and magnetic boundary conditions swap, the impression changes when considering the extra boundary terms due to moving to the weak form:

\[
0 = \int_0^{t_f} \int_{\Gamma_E} (v, w) \left( \left[ \begin{array}{cc} 0 & -A_n Zc \\ A_n Yc & 0 \end{array} \right] + \left[ \begin{array}{cc} 0 & A_n Zc \\ 0 & 0 \end{array} \right] \right) \cdot \left( \begin{array}{c} \lambda_H \\ \lambda_E \end{array} \right) \, dsdt
= \int_0^{t_f} \int_{\Gamma_E} (v, w) \left[ \begin{array}{cc} 0 & 0 \\ A_n Yc & 0 \end{array} \right] \cdot \left( \begin{array}{c} \lambda_H \\ \lambda_E \end{array} \right) \, dsdt.
\]

Thus, the same solver with the same boundary fluxes can again be used, provided one takes into account swapping \(\epsilon\) and \(\mu\).

6.2.2. Adjoint data management. Because the gradient formulation (24) is also based on the corresponding adjoint formulation, one faces the problem of storing the state of the forward problem that is needed for the backward integration of the adjoint equation (10). Since we consider the three-dimensional time-dependent case, the storage of the full trajectory would cause a tremendous memory requirement—too large even for today’s high-performance computing infrastructure.

However, due to the special problem structure here, paired with the boundary representation of the shape gradient, only a fraction of the data needs to be stored. In particular, we only need the offset of the simulated primal state \((H, E)\) on the sender/receiver boundary \(\Gamma_{obs}\) to generate the respective forcing (26)

\[
\int_0^{t_f} \int_{\Gamma_{obs}} (A^T(n)A(n) \cdot (H - H^m, E - E^m)) \cdot (v, w) \, dsdt.
\]

of the adjoint. In order to actually compute the gradient, we furthermore need the primal state only in the first cell layer around \(\Gamma_{inc}\) such that the trace of the volume divergence in (24) can be computed correctly. This leads to a tremendous reduction in data such that no checkpointing is needed. For all test cases considered as part of this work, the system RAM suffices to store all necessary data. Furthermore, we use a highly robust CVT-based mesh deformation [37] such that all meshes, meaning the mesh for generating the reference solution, the initial mesh, and all meshes generated during optimization, are topologically equivalent with the same connectivity. Because \(\Gamma_{obs}\) is fixed, the offset, (26), can always be evaluated without interpolation. Furthermore, we can reuse some structures of the ILU elimination process because the nonzero pattern never changes between test cases.

7. Automatic code generation, UFL, and shape derivatives.

7.1. State and adjoint solvers. Our solvers are based on the FEniCS finite element environment, which automatically generates a DG solver by processing the
variational problem specified within the Unified Form Language (UFL), which is a very high level domain specific embedded language in Python [2]. Other finite element packages also able to interpret variational problems specified in UFL are, e.g., Firedrake [33] and DUNE.\(^2\) The abstraction is best demonstrated by considering how the volume integrals (18) are translated to HPC software based on the following code snippet of the sources of our primal and adjoint solver:

```python
from dolfin import *

# load the mesh, read parameters, initialize memory
# declare DG spaces for H and E of appropriate degrees
H = VectorFunctionSpace(mesh, "DG", degreeH)
E = VectorFunctionSpace(mesh, "DG", degreeE)
# create mixed function space
HE = H*E

# declare test and trial functions of variational form
(v, w) = TestFunctions(HE)
(h, e) = TrialFunctions(HE)

# declare state unknown for Maxwell and initialize to zero
q_old = MakeInitialState()
(h_old, e_old) = split(q_old)

# declare variational problem with mid-point time integration (method of lines)
# volume integrals from equation (18), time implicit part
F = inner(v, mu/DeltaT*h)*dx + 0.5*inner(e, curl(v))*dx
F += inner(w, epsilon/DeltaT*e)*dx + 0.5*(-1.0)*inner(h, curl(w))*dx +
    (sigma)*inner(w, e)*dx

# volume integrals from equation (18), time explicit part
L = inner(v, mu/DeltaT*h_old)*dx - 0.5*inner(e_old, curl(v))*dx
L += inner(w, epsilon/DeltaT*e_old)*dx + 0.5*inner(h_old, curl(w))*dx +
    (sigma)*inner(w, e_old)*dx

# Add stabilization terms (facet integrals from (21)) and boundary fluxes

# create constant-in-time system matrix
A = assemble(F)

# time loop
[..] # in each time step: compress and store primal data necessary for adjoint

Due to symmetry considerations, the same code can also be used to the adjoint solver, provided some I/O is reversed, the source terms are adapted, and some constants are changed accordingly. The time integration of the gradient (24) is then also conducted “backwards-in-time” during execution of the adjoint solver.

7.2. Computing the shape derivative using UFL. It is worth noting that in this programming paradigm, which is very close to the mathematical notation of variational problems, the optimize-then-discretize shape derivative formulation of (24) is easily implemented via the following code:

```
# to be executed during the backwards-in-time adjoint solver
c = 1.0/sqrt(mu*epsilon)
Y = sqrt(epsilon/mu)
```

\(^2\)UFL support for DUNE is part of dune-fempy; see https://gitlab.dune-project.org/staging/dune-corepy.
Using the continuous shape derivative formulation is thus very beneficial for high abstraction HPC approaches. One on the one hand, the entire implementation of primal and adjoint solver and the gradient is very robust against human errors, because the programmer need only to supply integrals in mathematical notation. On the other hand, the continuous shape derivative formulation allows such high abstraction frameworks to become tools for shape optimization and computational geometry without introducing new abstractions or changing the sources. In particular, the framework does not need to provide the sensitivity of the PDE solution with respect to the input mesh on the discrete level.

7.3. Computing curvature. Implementing the evaluation of the shape derivative poses additional challenges, in particular the curvature computation for the regularization parameter. To this end, we fully utilize the possibility that FEniCS will generate PDE solvers on manifolds [34]. We first generate a triangle mesh \( \partial T_h \) embedded in \( \mathbb{R}^3 \) out of the facets \( \partial K \) forming the physical surface of the volume mesh \( T_h \). We define the continuous surface finite element space via

\[
S^1_h(\partial \Omega) := \{ \sigma \in C^0(\partial T_h) : \sigma|_{\partial K} \in \mathcal{P}_p(\partial K) \quad \forall \partial K \in \partial T_h \},
\]

The space \( S^1_h(\partial \Omega) \) can be seen as the surface analogue of the standard continuous Lagrange finite element space. The degrees of freedom coincide with vertices. First, we create a vertex normal \( n_h \in (S^1_h(\partial \Omega))^3 \) by averaging the facet normal of all \( (\partial K_i) \) sharing a common vertex into the corresponding degree of freedom. We then compute the discretized curvature \( \kappa_h \in S^1_h(\partial \Omega) \) by solving the following projection: Find \( \kappa_h \in S^1_h(\partial \Omega) \) such that

\[
\sum_{\partial K \in \partial T_h} \int_{\partial K} v \kappa_h \, dS = \sum_{\partial K \in \partial T_h} \int_{\partial K} v \text{div} n_h \, dS \quad \forall v \in S^1_h(\partial \Omega).
\]

On each surface triangle \( \partial K \), we can choose a local coordinate system out of the two orthogonal tangents \( \tau_1, \tau_2 \). The surface divergence of \( n_h \) is then computed intrinsically by applying the definition

\[
\text{div}_{\Gamma} n_h := \left( \frac{\partial n_h}{\partial \tau_1}, \tau_1 \right) + \left( \frac{\partial n_h}{\partial \tau_2}, \tau_2 \right)
\]

to the ansatz functions. For integration and ease of computation, the above integrals are transferred onto a reference triangle. The transformation Jacobian \( J \) in this case is a \( 3 \times 2 \) matrix, and the root of the Gram determinant is used as the pseudodeterminant for the integral transformations.

In addition to having a regularization parameter \( \delta \) in (1), we conduct our gradient descent scheme in the space \( H^1 \), meaning the shape derivative \( dJ \) from (24) is projected into the space \( H^1 \) by solving

\[
(\delta S \Delta_T + I)(W, n) = -dJ(\Omega)[V]
\]

7.3. Computing curvature. Implementing the evaluation of the shape derivative poses additional challenges, in particular the curvature computation for the regularization parameter. To this end, we fully utilize the possibility that FEniCS will generate PDE solvers on manifolds [34]. We first generate a triangle mesh \( \partial T_h \) embedded in \( \mathbb{R}^3 \) out of the facets \( \partial K \) forming the physical surface of the volume mesh \( T_h \). We define the continuous surface finite element space via

\[
S^1_h(\partial \Omega) := \{ \sigma \in C^0(\partial T_h) : \sigma|_{\partial K} \in \mathcal{P}_p(\partial K) \quad \forall \partial K \in \partial T_h \},
\]

The space \( S^1_h(\partial \Omega) \) can be seen as the surface analogue of the standard continuous Lagrange finite element space. The degrees of freedom coincide with vertices. First, we create a vertex normal \( n_h \in (S^1_h(\partial \Omega))^3 \) by averaging the facet normal of all \( (\partial K_i) \) sharing a common vertex into the corresponding degree of freedom. We then compute the discretized curvature \( \kappa_h \in S^1_h(\partial \Omega) \) by solving the following projection: Find \( \kappa_h \in S^1_h(\partial \Omega) \) such that

\[
\sum_{\partial K \in \partial T_h} \int_{\partial K} v \kappa_h \, dS = \sum_{\partial K \in \partial T_h} \int_{\partial K} v \text{div} n_h \, dS \quad \forall v \in S^1_h(\partial \Omega).
\]

On each surface triangle \( \partial K \), we can choose a local coordinate system out of the two orthogonal tangents \( \tau_1, \tau_2 \). The surface divergence of \( n_h \) is then computed intrinsically by applying the definition

\[
\text{div}_{\Gamma} n_h := \left( \frac{\partial n_h}{\partial \tau_1}, \tau_1 \right) + \left( \frac{\partial n_h}{\partial \tau_2}, \tau_2 \right)
\]

to the ansatz functions. For integration and ease of computation, the above integrals are transferred onto a reference triangle. The transformation Jacobian \( J \) in this case is a \( 3 \times 2 \) matrix, and the root of the Gram determinant is used as the pseudodeterminant for the integral transformations.

In addition to having a regularization parameter \( \delta \) in (1), we conduct our gradient descent scheme in the space \( H^1 \), meaning the shape derivative \( dJ \) from (24) is projected into the space \( H^1 \) by solving

\[
(\delta S \Delta_T + I)(W, n) = -dJ(\Omega)[V]
\]
on the boundary $\Gamma_{\text{inc}}$, where $I$ is the identity and $\Delta_{\Gamma}$ is the Laplace–Beltrami operator, i.e., the surface Laplacian. This is done weakly using the same spaces as above. For all our applications, we set $\delta_S = 10^{-3}$. It is worth noting that the $H^1$-descent is related to the shape Hessian of the regularization term used herein. A repeated shape differentiation of (8) in direction $W$ results in

$$
\frac{d^2}{ds^2} \left( \int_{\Gamma_{\text{inc}}} 1 \, ds \right) [V,W] = \int_{\Gamma_{\text{inc}}} (W,n) \text{div}(\kappa V) \, ds + \int_{\Gamma_{\text{inc}}} \langle V,n \rangle \kappa'[W] \, ds
$$

$$
= \int_{\Gamma_{\text{inc}}} (W,n)(V,n)\kappa^2 - \langle V,n \rangle \Delta_{\Gamma} \langle W,n \rangle \, ds
$$

$$
= \int_{\Gamma_{\text{inc}}} \langle W,n \rangle (V,n)\kappa^2 + \langle \nabla_{\Gamma} \langle V,n \rangle, \nabla_{\Gamma} \langle W,n \rangle \rangle \, ds,
$$

where the second equality requires perturbations $V, W$ in normal direction with independence in the sense that $(DV)^T W = 0$. The last equality stems from integration by parts. For further details see also [9, 14]. Thus, we solve (27) weakly using the normal component of trial functions $W$ and test functions $V$ within the boundary mesh. The tangential motion $W_{\Gamma}$ is then given by the CVT reparameterization [37], leading to a complete update of the type

$$
\Gamma^{k+1}_{\text{inc}} : = \{ x + \tau_s \cdot (W,n)n + W_{\Gamma} : x \in \Gamma^k_{\text{inc}} \},
$$

where $\tau_s$ is the step length of the descent scheme. Thus, the step in optimality is damped via $\tau_s$, whereas the step in mesh quality is applied fully. The volume mesh is adapted to this new boundary by solving a componentwise Laplacian with a successive CVT smoothing. This enables the use of topologically equivalent meshes for all computations. No remeshing is necessary.


8.1. Near-field reconstruction. Our first test case is the reconstruction of an obstacle by prescribing and measuring the near-field pulse. The base geometry is a rectangular cuboid of $24 \text{ cm by 24 cm by 12 cm edge length}$, where the rear side is deformed by the sigmoid function

$$
z_{\text{new}} = z + \frac{-W}{2 + 2 \exp (100(\sqrt{x^2 + y^2} - W))},
$$

where $W = \frac{299 792 458 \text{ m s}^{-1}}{3.1 \text{ GHz}} \approx 7.31 \text{ cm}$ is the wavelength of a sinusoidal pulse at $4.1 \text{ GHz}$, which creates a smooth bump of roughly $3.65 \text{ cm}$ elevation on the rear side of the cuboid. These numbers are chosen such that the physical dimensions of the inclusion are within the centimeter magnitude, which will be the natural wavelength for the antenna we are going to consider in the second test case. The domain is shown in Figure 4. We want to exploit the symmetry of this domain by computing only a quarter section, which is discretized using 35,945 elements. Comparing this to the verification results above suggests a good trade-off between accuracy, storage requirements and speed. Because the obstacle $\Gamma_{\text{inc}}$ assumed to be illuminated is a perfect conductor, only orthogonal electric fields can traverse over $\Gamma_{\text{inc}}$. Thus, we send two linearly polarized electromagnetic fields originating from $\Gamma_{\text{obs}}^1$ and $\Gamma_{\text{obs}}^2$. Consequently, the symmetry planes need to be of $\Gamma_H$ type, leaving only the possibility of sending a radially polarized wave from $\Gamma_{\text{obs}}^3$. 
The amplitude as a function of time of the incoming scanning signals is the same truncated SINC-pulse for all boundaries $\Gamma_{\text{obs}}^1$, $\Gamma_{\text{obs}}^2$, and $\Gamma_{\text{obs}}^3$, namely,

$$g(t) = \frac{\sin \left( 2\pi f_s (t - t_c) \right)}{2\pi f_s (t - t_c)} \sin \left( 2\pi f_c (t - t_c) \right) w(t, t_c),$$

to be multiplied with the respective vectors determining the base polarization. Here, $t_c$ is the time around which the pulse is centered, $f_c$ is the desired center frequency, and $f_s$ is the half bandwidth of the signal. We set $t_c = 3/(2 f_s)$ and use the Hamming window function

$$w(t, t_c) = \begin{cases} 0.54 + 0.46 \cos \left( \frac{\pi (t - t_c)}{t_c} \right) & \text{for } t \in [0, 2 t_c], \\ 0, & \text{otherwise.} \end{cases}$$

In particular, our center frequency equals $f_c = 8.2$ GHz and the spread is $f_s = 4.1$ GHz, meaning we scan using frequencies ranging from 4.1 GHz to 12.3 GHz. With respect to wavelengths, this corresponds to using waves ranging between 2.4 cm and 7.3 cm, the same order of magnitude as the 3.65 cm bump. The average edge length of the mesh is 7.56 mm, well below the wavelength of the major component of the scanning pulse. To accurately resolve these waves, we operate at time steps of 6.1 ps, 36.9% of the allowed CFL maximum. We terminate the simulation after 3.336 ns or, equivalently, 548 time steps. This is the time the electromagnetic wave needs to travel 1 m, and with a domain size of 12 cm we assume that all reflections have been captured by then. With respect to the Maxwell state unknowns, this leads to a total of 862,680 unknowns per time step or 472,748,640 unknowns for one primal simulation trajectory. Storing the complete trajectory information would require roughly 3.52 GB, of which only a fraction needs to be processed due to the boundary representation. The actual quarter domain inclusion is discretized using 607 vertices, of which the position is the primal design variable. Both reconstructed and target shapes are
shown in Figure 5, and their cross-sections are shown in Figure 6, where we have started from a flat backside. Furthermore, a perimeter penalization of $\delta = 5 \cdot 10^{-5}$ is used in (1). It is worth noting that for smaller values of $\delta$ the optimization usually converged to a local optimum with clear discrepancy in both the electromagnetic reflection of the reconstruction and the actual geometry. Finally, the development of the signal offset, that is, the value of (1) without the perimeter penalty term, is shown in Figure 7. Considering that the global optimum of zero signal offset is obtainable, the achieved objective value is very promising. In an actual application, the regularization parameter $\delta$ would probably need to be reduced in a homotopy fashion, matching the signal offset even better than the already achieved $9.22 \cdot 10^{-2}$ as shown in Figure 7. Comparing both the signal offset and the actual geometry, we conclude that the reconstruction has worked surprisingly well in this near-field example, opening the possibility to actually include the antenna interaction as a second test.

8.2. Reconstruction with horn antenna. The second test case is the reconstruction of an obstacle including the actual antenna geometry. We enhance the
obstacle as given in subsection 8.1 with a waveguide and a horn antenna, which acts as both emitter and receiver, creating the geometry as depicted in Figure 1. The waveguide geometry matches an E-band WR229 waveguide, that is, 2.29 inches \(\times\) 1.15 inches or 5.8166 cm \(\times\) 2.921 cm. Attached to this waveguide is a standard gain horn modeled after the Pasternack PE9862-20, which is 59.563 cm long and has a mouth of 31.496 cm \(\times\) 22.225 cm. The quarter domain is discretized using 33,305 elements, leading to 799,320 state unknowns per time step. The average edge length is 1.222 cm. The inclusion is discretized using 616 vertices. To account for the larger geometry, we now use 1,505 time steps. Thus, one complete primal trajectory consists of 1,202,976,600 unknowns, of which, again, only the respective boundary data is required. The end time is 18.34603 ns. It is worth noting that contrary to the near-field situation in subsection 8.1, the horn antenna and waveguide in this setup illuminate the obstacle from only one angle and act as a polarization filter, allowing only linearly polarized waves to pass through. The ideal frequency range for this antenna ranges from 3.3 GHz to 4.9 GHz. As such, we use the same truncated SINC-pulse as in (28) and (29), but with a center frequency of \(f_c = 4.1\) GHz and a spread of 0.41 GHz, which puts us well within the optimal frequency range of this antenna setup. As mentioned above, the corresponding wavelength is 7.31 cm, which is roughly twice the size of the bump depth. Reconstructed and target inclusions are shown in Figure 8. For this application, the penalty term \(\delta\) in (1) is set to zero. Although there is a clear visual discrepancy between the reconstructed and the desired shape, the actual signals as measured by the antenna coincide very well; see Figure 9. As in the near-field case, the initial geometry is a flat plate. Comparing this result with the near-field test, we can see that this particular experiment involving a horn antenna is not the ideal approach, most likely due to the polarization filtering and the single angle illumination. However, the very successful near-field reconstruction suggests the feasibility of the method, and as part of future studies we will include the antenna shape as an optimization variable, leading to an approach involving optimal design of experiments and optimization under uncertainties for these reconstruction problems.

9. Conclusions and outlook. A novel methodology to solve geometric inverse problems governed by Maxwell’s equations has been considered. Contrary to the state-of-the-art approach, the methodology here treats the problem as time-dependent hyperbolic and does not resort to frequency domain. This is achieved by tracking the information flow, which is identified via upwinding, through physical boundaries.

Exploiting the nature of such shape optimization problems by considering a boundary representation of the gradient allows for a considerable amount of data reduction for the backwards-in-time transient adjoint. Furthermore, direct sharp interface reconstruction based on deforming physical boundaries synergizes very well with discontinuous Galerkin (DG) methods and the upwind flux modeling of the tracking. The problem is first considered on a general level and then specialized to Maxwell’s
equations. After discussing a DG Maxwell solver and its adjoint on unstructured tetrahedral meshes, we conclude with two numerical test cases: The reconstruction of a bump based on near-field information alone and with the inclusion of a complete horn antenna setup. The near-field example is very promising, reconstructing not only the simulated and measured signal, but also the actual geometry. When including the horn antenna, we still manage to get a very good signal match, but the geometries are noticeably different, probably due to the polarization filtering and the narrow frequency band of the waveguide and horn antenna. Using only a single radiation source also seems problematic. For both problems, we see an excellent convergence behavior of a $H^1$-gradient descent scheme, which appears to be related to the shape Hessian of the perimeter penalization. Future work will be focused on including the antenna geometry into the optimization problem, thereby finding experimental setups that manage to recreate the very good match of the near-field problem also in the more realistic antenna case.
REFERENCES


LARGE-SCALE THREE-DIMENSIONAL ACOUSTIC HORN OPTIMIZATION

STEPHAN SCHMIDT†, EDDIE WADBRO‡, AND MARTIN BERGGREN‡

Abstract. We consider techniques that enable large-scale gradient-based shape optimization of wave-guiding devices in the context of three-dimensional time-domain simulations. The approach relies on a memory efficient boundary representation of the shape gradient together with primal and adjoint solvers semiautomatically generated by the FEniCS framework. The hyperbolic character of the governing linear wave equation, written as a first-order system, is exploited through systematic use of the characteristic decomposition both to define the objective function and to obtain stable numerical fluxes in the discontinuous Galerkin spatial discretization. The methodology is successfully used to optimize the shape of a midrange acoustic horn, described by 1,762 design variables, for maximum transmission efficiency, where the parallel computations involve a total of $3.5 \times 10^9$ unknowns.

Key words. shape optimization, shape derivatives, acoustic wave equation, computational acoustics, discontinuous Galerkin method, code generation

AMS subject classifications. 35L05, 65K10, 65M25, 65M32, 68N19

DOI. 10.1137/15M1021131

1. Introduction. Horn-like structures appear in devices for both acoustic and electromagnetic waves. A horn serves as an interface between a waveguide or a radiation source and the surrounding space and provides both impedance matching to the source and directivity control of the transmission. These properties are utilized in a variety of applications. The bell in brass instruments constitutes a part of a resonator—the air space within the instrument—and the bell shape is judiciously designed to constrain the most important resonances to integer frequency ratios. Recently, there have been some efforts to use numerical shape optimization in order to design brass bells so that the instrument acquires prescribed resonance characteristics [8, 20]. A type of acoustic inverse horn, that is, a device that possesses a large diameter at the source and a small one at the outlet, is used in ultrasonic machining, a subtractive manufacturing process particularly suitable for hard and brittle materials [1]. Horn loudspeakers, which are at focus here, are routinely used to supply the mid- and high-frequency range in public address systems for large halls, cinemas, and outdoors, often assembled in the form of so-called line arrays [34]. A horn dramatically raises the efficiency of the radiating source and may be used for precise directional control of the coverage area. In spite of its advantages, horn-equipped loudspeakers are often viewed as subpar with respect to sound quality [10, sect. 4.9]. However, some of the sound-quality deficiencies of horns may be due to suboptimal design and not to inherent limitations of the concept of acoustical horns.

The use of simple, classical shapes, such as exponentials, and the classical analysis of wave motion in horns, based on the one-dimensional (1D) Webster horn equation [19], is rapidly being complemented with more accurate numerical methods, which allow
analysis of much more complex shapes. During the last decade or so, a number of studies have appeared that use optimization algorithms together with numerical solutions of the governing equation to design acoustic horns \cite{3, 4, 6, 13, 16, 31, 32}. It turns out that detailed shaping of the horn flare by the use of numerical shape optimization methods can produce horns with much more favorable input impedance properties compared to those associated with classical shapes such as exponentials, as also has been confirmed by measurements on manufactured prototypes \cite{21}. The difference in input impedance properties between simple and optimized shapes is quite large, particularly at the low end of the operational frequency range. Unfavorable input impedance properties will give rise to acoustic resonances in the horn, and the typical "honky" megaphone-like sound associated with horns could be due to such resonances. We believe that there is therefore a great potential in the use of optimization to design horns with much less of such coloring. Some studies consider the directivity properties of the horn in the optimization process \cite{18, 30, 33} instead of, or in addition to, the input impedance. All the above-mentioned studies employ frequency-domain modeling and are confined to 2D geometries, which means that only cylindrical (or laterally infinite) horns can be considered. However, in practice, it is important to be able to separately control the vertical and horizontal directivity properties, which will necessitate full 3D acoustic models. Moreover, since the radiation properties of horns ideally should be uniform throughout their operational range, frequency-domain methods need to be run at a large number of frequencies. It may therefore be beneficial to consider time-domain methods, where only one equation needs to be solved at each design cycle. The frequency content for which the horn should be optimized is then controlled by selecting a suitable input pulse.

In contrast to previous work, our aim here is to apply numerical shape optimization for \textit{detailed design} of an acoustic horn in \textit{three dimensions} using a \textit{time-domain model} of the acoustic wave propagation. We consider here only the impedance-matching aspect, setting the stage for future treatment of more complex problems, such as optimizing with respect to directivity properties. The linear wave equation written in first-order form models the wave propagation. The hyperbolic nature of the equations is respected through the use of a discontinuous Galerkin spatial discretization with upwinding fluxes based on a characteristic decomposition \cite{14}. The characteristic decomposition is also used to define the objective function.

Since we rely on gradient-based optimization and the adjoint-variable method for the sensitivity analysis, the computational complexity for each gradient evaluation is independent of the number of design variables. We use this property to be able to control the shape in detail; each mesh point on the horn's surface is subject to design, and a smoothing strategy is utilized to promote smooth design updates. For efficiency, both computationally and with respect to implementation effort and maintainability of the software, we make use of the FEniCS \cite{17} suite to generate the primal and dual solver for acoustic analysis. The gradient expression used by the optimization routine is assembled from the primal and adjoint solutions using an expression derived from the equations in integral form. This approach yields a boundary-integral representation for the objective function gradient, which means that the gradient expression requires only the time history of the primal and adjoint solutions restricted to the design surface, and not everywhere in the computational domain, which would be extremely memory demanding for a time-domain calculation. The necessity to differentiate the mesh deformation procedure is likewise eliminated by this approach.

The paper is structured as follows. In section 2, we review the governing equations and the concept of characteristic decomposition, which is central to this application.
Sections 3 and 4 introduce the objective function and the resulting first-order optimality system. The discretization scheme is presented in section 5, and results from cross-code verification studies are given in section 6. Finally, section 7 presents results of a large-scale intrinsically 3D horn optimization.

2. Acoustic power balance and the characteristic decomposition. We use the concept of a characteristic decomposition of the acoustic power flux over surfaces extensively throughout the following. The concept, which is briefly reviewed in this section, is used to define physical boundary conditions, lay out the objective function of optimization, and specify interelement fluxes in the numerical scheme.

Acoustic wave propagation in still air under the conditions of uniform static density and temperature is governed by the linear wave equation

\begin{align}
\frac{\partial u}{\partial t} + \nabla p &= 0, \\
\frac{\partial p}{\partial t} + c^2 \text{div } u &= 0,
\end{align}

where \( p \) denotes the acoustic pressure; \( u \) the acoustic momentum density vector, that is, the product of the static air density and the acoustic velocity vector; and \( c \) the speed of sound.

For each open, bounded domain \( D \) in space, solutions to system (1) satisfy the conservation law

\begin{equation}
\frac{d}{dt} \frac{1}{2} \int_D \|u\|^2 + \frac{1}{c^2} p^2 \, dx = -\int_{\partial D} p(u, n) \, d\Gamma,
\end{equation}

where \( n \) is the outward-directed unit normal on \( \partial D \), \( \langle \cdot, \cdot \rangle \) denotes the Euclidean inner product, and \( \|u\|^2 = \langle u, u \rangle \). Conservation law (2) says that the time derivative of the acoustic energy in \( D \) equals the net flux of acoustic power\(^1\) into \( D \) through boundary \( \partial D \). Conservation law (2) is obtained by taking the dot product of equation (1a) with \( u \), multiplying equation (1b) by \( p/c^2 \), summing the equations, integrating, and using the divergence theorem.

The integrand on the right-hand side of conservation law (2) can be split up into a difference of two positive terms as follows:

\begin{equation}
-p(u, n) = \frac{1}{c} \left( \frac{1}{2} (p - c(u, n)) \right)^2 - \frac{1}{c} \left( \frac{1}{2} (p + c(u, n)) \right)^2 = \frac{1}{c} (w_-^2 - w_+^2),
\end{equation}

where

\begin{equation}
w_{\pm} = \frac{1}{2} (p \pm c(u, n))
\end{equation}

are called the characteristic variables. By substituting the splitting (3) into conservation law (2), we find that

\begin{equation}
\frac{d}{dt} \frac{1}{2} \int_D \|u\|^2 + \frac{1}{c^2} p^2 \, dx = \frac{1}{c} \int_{\partial D} w_-^2 - w_+^2 \, d\Gamma,
\end{equation}

\(^1\)To obtain the correct dimensions of power, the terms in equation (2) should be divided by the static air density.

© 2016 Stephan Schmidt, Eddie Wadbro, Martin Berggren
which reveals that the use of characteristic variables yields a splitting of the power flux over \( \partial D \) into the acoustic power that flows into the domain \( D \) (integral of \( w_-^2 \)) and the the power that flows out of the domain (integral of \( w_+^2 \)). The plus and the minus subscripts in the characteristic variables indicate the direction in relation to the (outward-directed) normal \( n \) on \( \partial D \).

### 3. The model problem.

We consider the 3D setup illustrated in Figure 1. To reduce the computational cost, we will compute on only a quarter of the domain. This setup consists of a waveguide attached to the throat of an acoustic horn that is mounted in an infinite baffle. We denote by \( \Gamma_{\text{wall}} \) the sound-hard walls of the waveguide, the acoustic horn, and the baffle, and by \( \Gamma_{\text{symm}} \) the symmetry boundaries. For numerical reasons, we truncate the domain and denote the truncated computational domain by \( \Omega \). The boundary \( \partial \Omega \) can be decomposed as the closure of \( \Gamma_{\text{wall}} \cup \Gamma_{\text{symm}} \cup \Gamma_{\text{in}} \cup \Gamma_{\text{out}} \), where the two latter parts stem from the truncation of the computational domain. More precisely, \( \Gamma_{\text{in}} \) truncates the waveguide, and \( \Gamma_{\text{out}} \) is the boundary that truncates the free-space in front of the horn. The acoustic initial–boundary-value problem we consider is

\[
\begin{align*}
\frac{\partial u}{\partial t} + \nabla p &= 0 & \text{in } \Omega & \text{for } t > 0, \\
\frac{\partial p}{\partial t} + c^2 \text{div } u &= 0 & \text{in } \Omega & \text{for } t > 0, \\
\frac{1}{2} (p - c \langle u, n \rangle) &= g & \text{on } \Gamma_{\text{in}} & \text{for } t > 0, \\
\frac{1}{2} (p - c \langle u, n \rangle) &= 0 & \text{on } \Gamma_{\text{out}} & \text{for } t > 0, \\
\langle u, n \rangle &= 0 & \text{on } \Gamma_{\text{wall}} \cup \Gamma_{\text{symm}} & \text{for } t > 0, \\
u &\equiv 0 & \text{in } \Omega & \text{at } t = 0, \\
p &\equiv 0 & \text{in } \Omega & \text{at } t = 0,
\end{align*}
\]

where \( g \) is a given function with compact support in time. From the discussion in section 2, we see that the boundary condition on \( \Gamma_{\text{in}} \) sets the incoming characteristic \( (w_-) \) to be equal to the given function \( g \), and the boundary condition on \( \Gamma_{\text{out}} \) ensures
that no incoming acoustic power is transmitted into \( \Omega \) from the exterior. The condition of
a sound-hard material is imposed through the boundary condition on \( \Gamma_{\text{wall}} \). The
zero normal velocity condition also imposes the symmetry condition at \( \Gamma_{\text{symm}} \).

Applying the power balance law (5) on solutions to system (6) under the specified
boundary conditions, we find that

\[
\frac{d}{dt} \frac{1}{2} \int_{\Omega} \|u\|^2 + \frac{1}{c^2} \int_{\Gamma_{\text{in}}} g^2 - \int_{\Gamma_{\text{out}}} w^2 d\Gamma,
\]

where we have made use of the characteristic variables as defined in expression (4).

Since we assume that the input \( g \) has compact support in time, it holds for an “open”
system like this one that the solution will vanish identically after a sufficiently long time
\( T \), which is consistent with the fact that the right-hand side of conservation law (7)
becomes negative when \( g \equiv 0 \). Thus, integrating expression (7) over a sufficiently
long time interval \( (0, T) \), utilizing the initial condition \( u(0) = 0 \) and \( p(0) = 0 \) and the
vanishing assumption \( u(T) = 0 \) and \( p(T) = 0 \), we find that

\[
\int_{0}^{T} \int_{\Gamma_{\text{in}}} g^2 d\Gamma = \int_{0}^{T} \int_{\Gamma_{\text{in}}} w^2 d\Gamma + \int_{0}^{T} \int_{\Gamma_{\text{out}}} w^2 d\Gamma.
\]

Expression (8) states the basic energy balance of system (6): the total acoustic energy
of the input signal (left-hand side) equals the energy of the reflected signal (first term
on the right-hand side) plus the energy transmitted to the surroundings (second term
on the right-hand side). Thus, to maximize the total transmitted energy, we may
equivalently minimize the reflected signal, which will be simpler to do in a numerical
implementation.

Here, we let the shape of boundary \( \Gamma_{\text{horn}} \), colored light gray in Figure 1, be
subject to design. The beginning and the end of the horn flare will be fixed, so that
the optimization can change neither the mouth and throat shapes nor the length of
the horn. Each such admissible horn shape will generate a candidate computational
domain \( \Omega \), for which we solve state equation (6) and evaluate the objective function

\[
J(\Omega) = \frac{1}{2} \int_{0}^{T} \int_{\Gamma_{\text{in}}} (p + \langle u, n \rangle)^2 d\Gamma dt,
\]

integrated for a sufficiently long time \( T \) with respect to the time support of the input
signal \( g \). The optimization problem can then be formulated as

\[
\min J(\Omega) \quad \text{subject to} \quad \text{state equation (6)}.
\]

If the final design is such that the objective function, that is, the reflected signal,
vanishes for the given input signal, then the input impedance at the throat of the
horn is real and constant for each of the frequencies contained in the signal. (For a
discussion of the relation between reflections, which is the measure we use here, and
the concept of acoustic impedance, we refer the reader to the acoustics literature, for
instance Rienstra and Hirschberg’s lecture notes [22, sect. 3.2].) This approach is
appropriate for transmission optimization of a horn in isolation, as here. However,
if a model of the wave source, typically a compression driver in the case of a mid-
or high-frequency horn, is available, it would also be possible to maximize the total
transmitted acoustic power for the coupled system driver–horn.
4. Domain variations and optimality system. Conceptually, design problem (10) is a PDE-constrained optimization problem with the added difficulty that there is a need to associate with the shape of $\Gamma_{\text{horn}}$ a suitable set of decision variables in the optimization algorithm. There are several possible choices. One possibility is to introduce an artificial inhomogeneous static density function into the governing equations, so that regions of sound-hard materials are approximated using a dense fluid. This approach transforms the problem from one using a varying domain to one that involves a variable coefficient in the governing equations [5, 27, 32]. Another possibility is to use an explicit parameterization of feasible domain shapes through, for instance, a finite set of smooth ansatz functions [15]. Here, we consider domain deformations that are assumed to be generated by a sufficiently smooth vector field $V : \mathbb{R}^3 \to \mathbb{R}^3$. This approach can be exploited to derive a computationally efficient boundary representation of the shape derivative, a property related to the so-called Hadamard–Zolésio structure theorem [11, 28].

A boundary representation of the shape gradient allows for very efficient numerical schemes utilizing the maximum degrees of freedom by exercising the position of every surface vertex as a design parameter. Furthermore, all terms stemming from a deformation of the mesh and those related to the differentiation of the PDE solution procedure with respect to the input mesh are treated on an analytic level, thereby circumventing the need to actually compute them on a discrete level. The resulting methodology is independent of the actual PDE solver, and the sensitivities can be computed using any methodology to solve the state equation, provided an adjoint is also available. The applicability of this approach to large-scale 3D problems in aerodynamics was, for example, considered previously in [25].

The appendix carries out the shape sensitivity analysis for optimization problem (10) with state equation (6) written in a suitable integral form. To obtain the final boundary representation for the shape derivative, quite delicate conditions on the regularity of the solution have to be assumed, as indicated in the appendix. Such high regularity does not hold for typical finite-element functions, which means that the boundary representation of the shape derivative, as derived in the appendix, will not be fully consistent with a differentiation of the objective function actually used after discretization [7], [11, Remark 2.3]. However, this gap in consistency is usually reduced as the mesh is refined, an effect also visible in our finite difference verification in section 6.2.

For the studied problem, the admissible shape changes of $\Gamma_{\text{horn}}$ are generated through a smooth vector field $V : \mathbb{R}^3 \to \mathbb{R}^3$ that vanishes everywhere on $\partial \Omega$ except on $\Gamma_{\text{horn}}$. After a sensitivity analysis, as outlined in the appendix, we find that the shape derivative of objective function (9) with respect to $V$ has the boundary representation

$$dJ(\Omega)[V] = 2c \int_0^T \int_{\Gamma_{\text{horn}}} (V, n) \text{div} (u^*) \, d\Gamma \, dt,$$

where $u$ solves system (6) and $p^*$ the adjoint system

$$-\frac{\partial u^*}{\partial t} + \nabla p^* = 0 \quad \text{in } \Omega \text{ for } t < T,$$

$$\frac{\partial p^*}{\partial t} + c^2 \text{div} u^* = 0 \quad \text{in } \Omega \text{ for } t < T,$$

$$\frac{1}{2} (p^* - c(u^*, n)) = \frac{1}{2} (p + c(u, n)) \quad \text{on } \Gamma_{\text{in}} \text{ for } t < T.$$
THREE-DIMENSIONAL HORN OPTIMIZATION

\[
\frac{1}{2} (p^* - c(u^*, n)) = 0 \quad \text{on } \Gamma_{\text{out}} \text{ for } t < T,
\]
\[
\langle u^*, n \rangle = 0 \quad \text{on } \Gamma_{\text{wall}} \cup \Gamma_{\text{symm}} \text{ for } t < T,
\]
\[
u^* \equiv 0, \quad p^* \equiv 0 \quad \text{in } \Omega \text{ at } t = T.
\]

We note that the adjoint system (12) after the variable change \( t \mapsto T - t \) equals the state equation (6) with \( w_+(T - t)|_{\Gamma_{\text{in}}} \) instead of \( g(t) \) as source. That is, the state and adjoint systems are the same, but the state equation is driven by the given source function \( g \), and the adjoint equation by the time convolution of the reflected signal at the inlet.

5. Spatial discretization. To introduce the variational form that is the basis for our discontinuous Galerkin discretization, we consider an open, bounded, and connected set \( K \subset \Omega \), representing what later will be an element in our triangulation. Assume that \( u \) and \( p \) satisfy system (6). Multiplying the first and second equations in system (6) with arbitrary smooth test functions \( v \) and \( q \), respectively, integrating over \( K \), and integrating by parts in space, we find that \( u(t)|_K \) and \( p(t)|_K \) satisfy

\[
\int_K \langle v, \partial_t u \rangle - p \, \text{div} \, v + q \, \partial_t p - c^2 \langle u, \nabla q \rangle \, dx + \int_{\partial K} \langle v, n \rangle p + c^2 \langle u, n \rangle q \, d\Gamma = 0
\]

for each pair of smooth test functions \( v \) and \( q \) with support in \( K \). Next, we introduce the flux functions

\[
f_1(u, p, n) := np = n(w_+ + w_-),
\]
\[
f_2(u, p, n) := c\langle u, n \rangle = (w_+ - w_-),
\]

which, as we see above, can be written in terms of sums and differences of the characteristic variables (4), a property that will be exploited below to define the numerical flux functions. Using definitions (14), equation (13) can be written as

\[
\int_K \langle v, \partial_t u \rangle - p \, \text{div} \, v + q \, \partial_t p - c^2 \langle u, \nabla q \rangle \, dx + \int_{\partial K} \langle v, f_1 \rangle + c q f_2 \, d\Gamma = 0.
\]

Now introduce a triangulation \( \mathcal{T}_h \) of the domain \( \Omega \) consisting of nonoverlapping open tetrahedrons \( K \) such that \( \overline{\Omega} = \bigcup_{K \in \mathcal{T}_h} K \). The numerical scheme is defined by requiring \( p(t) \) and the components of vector \( u(t) \) to be functions whose restrictions on each \( K \) are polynomials that satisfy a modified version of variational expression (15) for all polynomial test functions \( v \) and \( q \). Since \( p \) and \( u \) will then in general possess jump discontinuities over each interface between two neighboring elements, the question is what values to use in the flux functions \( f_1, f_2 \), since they are evaluated exactly where the functions are discontinuous. A standard choice that leads to a consistent and stable scheme is upwinding. This method exploits the directivity information in the characteristic variables and uses the local values in \( K \) in the outgoing characteristic variable \( w_+ \) and the remote values—from neighboring cells or from the boundary conditions—in the incoming characteristic variable \( w_- \). More precisely, let \( P_r \) be the space of polynomials of maximum degree \( r \). For each \( K \in \mathcal{T}_h \) we require \( u(t)|_K \in P_r^u \), \( p(t)|_K \in P_r \), such that for each \( v \in P_r^u, q \in P_r \),

\[
\int_K \langle v, \partial_t u \rangle - p \, \text{div} \, v + q \, \partial_t p - c^2 \langle u, \nabla q \rangle \, dx + \int_{\partial K} \langle v, f_1 \rangle + c q f_2 \, d\Gamma = 0,
\]

\( \copyright \ 2016 \) Stephan Schmidt, Eddie Wadbro, Martin Berggren
where the upwind numerical flux functions are defined by

\[
\begin{align*}
f_1^* &:= n(u^L_+ + w^R_+) = n \left( \frac{1}{2} (p^L + p^R) + \frac{c}{2} \left( \langle u^L, n \rangle - \langle u^R, n \rangle \right) \right), \\
f_2^* &:= u^L_+ - w^R_+ = \frac{1}{2} (p^L - p^R) + \frac{c}{2} \left( \langle u^L, n \rangle + \langle u^R, n \rangle \right),
\end{align*}
\]  

(17)

where the superscripts \(L\) and \(R\) denote local and remote values. The local values are given by the values of \(u|_K\) and \(p|_K\) on \(\partial K\). The remote values are given either in order to assign boundary conditions, if \(\partial K\) coincides with the boundary, or by the values of \(u|_{K'}\) and \(p|_{K'}\), if there is a neighboring \(K' \in \mathcal{B}_h\) such that \(K' \cap K \subset \partial K\), that is, we are at a face that is shared between \(K\) and \(K'\). The term “upwind” for the flux function (17) is borrowed from fluid mechanics and is motivated by the fact that the characteristic variables are evaluated according the direction of power transport across \(\partial K\).

By imposing the remote states

\[
\begin{align*}
u^R := -\frac{1}{c} gn, \quad p^R := g & \text{ on } \Gamma_{in}, \\
u^R := 0, \quad p^R := 0 & \text{ on } \Gamma_{out},
\end{align*}
\]  

(18)

we see that the incoming characteristic \(w^R\) on \(\Gamma_{in}\) and \(\Gamma_{out}\) will be set to the correct values as specified by system (6).

Since we compute on only a quarter of the horn, we impose that the solution should be symmetric across planes \(\Gamma_{symm}\). The symmetry and sound-hard boundary condition \(\langle u, n \rangle = 0\) is here imposed by the custom numerical flux function

\[
f_1^* |_{ws} = np^L, \quad f_2^* |_{ws} = 0,
\]

(19)

which can be motivated by substituting condition \(\langle u, n \rangle = 0\) into the exact flux functions (14). An alternative strategy to impose condition \(\langle u, n \rangle = 0\) is to use the upwind flux (17) and specify a remote state that mirrors the local state according to the formula \(u^R := u^L - 2 \langle u^L, n \rangle n\) and \(p^R := p^L\) on \(\Gamma_{symm}\). However, in the numerical experiments below, we have chosen the explicit wall flux function (19), since this choice will lead to a boundary integral over \(\Gamma_{horn}\) in the variational form whose integrand is linear in the wall normal. This property is consistent with the choice made to enforce the wall boundary condition in the integral form used for sensitivity analysis, equation (40), and leads to a much simpler expression for the shape gradient compared to the case when mirroring the local state; see the discussion in Appendix A.1.

To specify the complete scheme, based on the discussion above, we need a number of definitions. Let \(S = \bigcup_{k=1}^M S_m\) be the union of all open triangular element faces \(S_m\) that are shared between two elements, that is, the set of element faces that are not part of the boundary. For each such internal element face \(S_m\), it holds that \(\overline{S}_m = \partial K_{m_1} \cap \partial K_{m_2}\) for some distinct neighboring elements \(K_{m_1}, K_{m_2} \in \mathcal{T}_h\). We denote by \(n_{m_1}, n_{m_2} = -n_{m_1}\) the outward-directed unit normals to \(K_{m_1}\) and \(K_{m_2}\) on \(S_m\). For each piecewise-polynomial \(f\), we define \(f_1 = f|_{K_{m_1}}, f_2 = f|_{K_{m_2}}\), and the face-sum and face-jump operators

\[
\{ f \} := f_1|_{S_m} + f_2|_{S_m},
\]

\[
\langle f \rangle := \begin{cases} 
  n_{m_1} f_1|_{S_m} + n_{m_2} f_2|_{S_m} & \text{if } f \in \mathbb{R}, \\
  \langle n_{m_1}, f_1|_{S_m} \rangle + \langle n_{m_2}, f_2|_{S_m} \rangle & \text{if } f \in \mathbb{R}^3.
\end{cases}
\]

(20)

© 2016 Stephan Schmidt, Eddie Wadbro, Martin Berggren
Moreover, we define $\Omega_0 := \Omega \setminus S$ and the function

\[
\hat{g} := \begin{cases} 
  g & \text{on } \Gamma_{\text{in}}, \\
  0 & \text{on } \Gamma_{\text{out}}.
\end{cases}
\]

Now we sum equation (16) over all $K \in \mathcal{T}_h$, utilizing the upwind flux (17) over the element interfaces, imposing the remote states (18) and the wall/symmetry fluxes (19). We then obtain that $u(t) \in P_3^h(\mathcal{F}_h)$, $p(t) \in P_r(\mathcal{F}_h)$ satisfy

\[
\int_{\Omega_0} (v, \partial_t u) - p \operatorname{div} v + q \partial_t p - c^2(u, \nabla q) \, dx + \int_{\Gamma_{\text{wall}} \cup \Gamma_{\text{symm}}} (v, n p) \, d\Gamma
\]

\[
+ \int_{\Gamma_{\text{in}} \cup \Gamma_{\text{out}}} \left( v, n \left( \hat{g} + \frac{1}{2}(p + c(u, n)) \right) \right) + c q \left( \frac{1}{2}(p + c(u, n)) - \hat{g} \right) \, d\Gamma
\]

\[
+ \frac{1}{2} \int_S [v]\{p\} + [u]\{p\} \, d\Gamma + \frac{1}{2} \int_S c\{\nabla q\}, \{p\} \, d\Gamma = 0
\]

for each $v \in P_3^h(\mathcal{F}_h)$, $q \in P_r(\mathcal{F}_h)$.

6. Implementation and verification.

6.1. Verification of the forward solver. Variational problem (22) is implemented numerically using the FEniCS environment [17], which contains a domain-specific language for variational problems. After postulating the variational expression in Python, a C/C++ discontinuous Galerkin solver of the desired spatial order is automatically generated and compiled by the FEniCS environment. Time derivatives are discretized by the trapezoidal method. (A completely explicit low storage fourth-order Runge–Kutta scheme is also implemented, but this scheme behaves problematically if the mesh quality degrades too much during shape optimization when the mesh is deformed.) The implicit equation to solve for each time-step is linear. Depending on available compute power, spatial order, and mesh refinement, we first attempt to factorize the system once and store this factorization for all time-steps. If this is not tractable, for example due to excessive storage requirements, we solve the implicit equation for each time-step using restarted GMRES with ILU preconditioning, a functionality provided by PETSc and Hypre.

In the frequency domain, a horn’s radiation efficiency can be characterized by its reflection coefficient at the throat, which for each frequency measures the (complex-valued) quotient between the pressure amplitude of an incoming single frequency wave and the reflected wave’s pressure amplitude. To verify our implementation, we compute the reflection coefficient spectrum with our code as well as with Comsol Multiphysics on the cylindrically symmetric domain illustrated in Figure 2. Here, the width and length of the waveguide are set to $a = 19.3$ mm and $d = 100$ mm, respectively. The length of the conical horn flare is $l = 150$ mm, and the half width of the horn mouth is $b = 100$ mm. For all simulations, the speed of sound is $c = 345$ m/s.

The baseline for the verification computations is carried out with Comsol in the frequency domain. That is, we consider single-frequency wave propagation with angular frequency $\omega = 2\pi f$, where $f$ is the ordinal frequency. We use the ansatz $p(x,t) = \text{Re}(\tilde{p}_m(x)e^{i\omega t})$ and solve the following boundary-value problem [30] for the
complex amplitude function $\tilde{p}_\omega$:

$$
\frac{\partial}{\partial r} \left( \frac{1}{r} \frac{\partial \tilde{p}_\omega}{\partial r} \right) + \frac{\partial}{\partial z} \left( \frac{1}{r} \frac{\partial \tilde{p}_\omega}{\partial z} \right) + k^2 r \tilde{p}_\omega = 0 \quad \text{in } \Omega^{(2D)},
$$

$$
\left( ik \tilde{p}_\omega + \frac{1}{R} \right) \tilde{p}_\omega + \frac{\partial \tilde{p}_\omega}{\partial n} = 0 \quad \text{on } \Gamma^{(2D)}_{out},
$$

$$
\left( ik \tilde{p}_\omega + \frac{1}{R} \right) \tilde{p}_\omega + \frac{\partial \tilde{p}_\omega}{\partial n} = 2ikA \quad \text{on } \Gamma^{(2D)}_{in},
$$

$$
\frac{\partial \tilde{p}_\omega}{\partial n} = 0 \quad \text{on } \Gamma^{(2D)}_{horn} \cup \Gamma^{(2D)}_{symm},
$$

where $k = \omega/c$. The boundary conditions on $\Gamma^{(2D)}_{in}$ and $\Gamma^{(2D)}_{out}$ are frequency-domain analogues of characteristic boundary conditions used in the discontinuous Galerkin code at corresponding boundaries. The condition on $\Gamma^{(2D)}_{in}$ imposes a planar right-going wave with amplitude $A$ while absorbing planar left-going waves. At $\Gamma^{(2D)}_{out}$, the boundary condition absorbs any outgoing waves propagating in the normal direction with respect to the boundary. The reflection coefficient is given by

$$
R^{(2D)}_\omega = \frac{2}{a^2 A} \int_{\Gamma^{(2D)}_{in}} r(\tilde{p}_\omega - A) d\Gamma.
$$

For the 3D time domain discontinuous Galerkin computation, we use a truncated sinc pulse as input signal,

$$
g(t) = \frac{\sin \left( 2\pi f_s (t - t_c) \right)}{2\pi f_s (t - t_c)} \sin \left( 2\pi f_c (t - t_c) \right) w(t, t_c),
$$

where $t_c$ is the time around which the pulse is centered, $f_c$ is the desired center frequency, $f_s$ is the half bandwidth of the signal, and $w(t, t_c)$ is a window function. Here, we set $t_c = 3/(2f_s)$ and use the Hamming window function

$$
w(t, t_c) = \begin{cases} 
0.54 + 0.46 \cos \left( \frac{\pi(t-t_c)}{t_c} \right) & \text{for } t \in [0, 2t_c], \\
0 & \text{otherwise}.
\end{cases}
$$

Without a window function, the spectrum of the time-infinite sinc pulse is flat within $[f_c - f_s, f_c + f_s]$ and zero outside. The use of a window function converts the infinite
signal to a signal of finite duration but also impacts the spectrum. In all numerical experiments using the time domain solver, we use an input signal with parameters $f_c = 5.3$ kHz and $f_s = 3.7$ kHz. Figure 3 illustrates the input signal as a function of time (left diagram) and its normalized magnitude spectrum computed by the discrete Fourier transform (DFT) (right diagram).

For the baseline 2D Comsol simulations, the radius of the computational domain is $R = R_z = R_r = 1500$ mm, and we use continuous piecewise-quadratic elements on a triangular mesh with maximum side length 10 mm, which yields 97,769 unknowns for the linear system. We carry out the 3D time domain verification simulations on a cylindrical symmetric horn model in quarter symmetry, using discontinuous piecewise-linear elements on two different unstructured tetrahedral meshes, a fine and a crude resolution mesh.

The fine resolution mesh uses $R_z = 240$ mm, $R_r = 220$ mm and consists of 114,417 elements, which yields a total of $(3 + 1) \cdot 4 \cdot 114,417 = 1,830,672$ unknowns per time-step. The shortest edge in the mesh is 0.78 mm, the longest is 29.65 mm, and the average is 5.46 mm. As a comparison, note that the wavelength at 9 kHz is about 38 mm. For the fine resolution mesh, we use the end time $T = 18$ ms and the time-step $dt = 5.0 \cdot 10^{-7}$. The crude resolution mesh uses $R_z = 80$ mm and $R_r = 140$ mm and consists of 8,356 elements, which yields 133,696 unknowns per time-step. The shortest edge in this mesh is 2.51 mm, the longest edge is 36.16 mm, and the average is 9.22 mm. Here, the end time $T = 4.5$ ms, and the time-step $dt = 2.5 \cdot 10^{-6}$.

The initial 3D experiments focus on studying the transient behavior of the wave propagation and the decay of energy inside the domain. For our input signal, these experiments confirm that the acoustic energy inside the domain is exponentially decreasing, and we conclude that after about 4 ms—corresponding approximately to 10 $t_c$—the energy inside the computational domain is less then $10^{-6}$ of the energy of the input pulse. (Here, the energy inside the computational domain is evaluated as the difference between the numerically evaluated left- and right-hand sides of energy balance (8).) We also experimented with different time-steps for the implicit time stepper and found that the above mentioned $dt = 2.5 \cdot 10^{-6}$ provides a good trade-off between computational effort and accuracy for the standard resolution mesh.

During the time domain simulations, we record the outgoing signal at $\Gamma_{\text{in}}$. After the simulation is completed, we compute the spectrum of the outgoing signal using the
DFT. The reflection coefficient spectrum at the horn’s throat can then be computed by pointwise division of the Fourier coefficients of the input signal and the reflected signal. Figure 4 shows the magnitude of the reflection coefficient versus frequency computed by Comsol Multiphysics for the cylindrically symmetric frequency domain setup (solid line) and by our discontinuous Galerkin 3D time domain method on the two different meshes. The spectrum computed using the fine-mesh fully 3D time domain method agrees well with the reference computation, and we note that the use of a too-crude mesh tends to overestimate variations in the reflection coefficient, which is likely due to a too-crude and nonsmooth approximation of the horn geometry.

6.2. Finite difference verification of the shape derivative. Following the previous verification of the solver, we next verify the shape derivatives computed by (51) and (52) against finite difference approximations. The geometry is as in the previous section, and we again perform the test using two different meshes and far-field locations. For each vertex $x_i$ on the design boundary, we define a vector-valued perturbation function $V_i : \Gamma_{\text{horn}} \rightarrow \mathbb{R}^3$ by

$$ V_i(x) = \phi_i(x) n_{\text{vertex}}(x_i), $$

where $\phi_i$ is the linear Lagrangian basis function (the standard “hat” function centered at $x_i$). Thus, $V_i$ equals the vertex normal at vertex $x_i$; it has support only within $\text{Patch}(x_i)$, the surface patch formed around $x_i$, and it vanishes outside this patch. We compute each component of the adjoint-based directional derivative via (51) and (52) by integrating

$$ dJ_1 := dJ(\Gamma_{\text{horn}})[V_i] = 2c \int_0^T \int_{\text{Patch}(x_i)} \langle V_i, n_{\text{facet}} \rangle \, \text{div} \, (u^*) \, d\Gamma \, dt, $$

$$ dJ_2 := dJ(\Gamma_{\text{horn}})[V_i] = 2c \int_0^T \int_{\text{Patch}(x_i)} \langle V_i, n_{\text{facet}} \rangle \, \text{div} \, (u^* p) \, d\Gamma \, dt. $$

Note that for the adjoint-based directional derivative, the primal and dual states need to be computed only once on the unperturbed geometry. To compute the adjoint,
we restrict the storage of the primal state to a tubular neighborhood of two cells around \( \Gamma_{\text{wall}} \), which reduces the memory demand for the primal trajectory considerably yet still allows for a correct computation of the source term in the adjoint and the divergence operator constituting the gradient. Due to the similarity of the primal and dual equations, we can easily re-use most components of the primal solver. The same strategy is used during optimization.

For the finite difference computation used for verification purposes, we extend a damped perturbation \( \tau \cdot V_i \) into the domain by solving a noncoupled componentwise Laplace equation for the displacement of the mesh vertices, as we also do during the optimization to deform the volume mesh. We use \( \tau = 10^{-4} \) as the constant finite difference step length. The objective function (9) is evaluated on both the perturbed and original grids in order to calculate a one-sided finite difference.

As in the preceding section, we verify both formulas on two different meshes, the crude resolution mesh with a close far-field boundary (\( R_z = 80 \) mm and \( R_r = 140 \) mm) and a fine resolution mesh with the far-field boundary further away (\( R_z = 240 \) mm and \( R_r = 220 \) mm). However, the fine resolution mesh is here derefined once, compared to the mesh used in previous section, to keep the compute time for the finite difference verification manageable. Thus, we consider either 698 design variables on the crude mesh (“mesh 1”) or 1,236 design variables on a finer mesh (“mesh 2”). The results are shown in Table 1. As expected, the difference between \( dJ_1 \) and \( dJ_2 \) is marginal; we thus focus on \( dJ_1 \) here. As discussed in section 4, there are quite restrictive regularity assumptions necessary for a consistent shape derivative. Nevertheless, the higher resolution of the finer mesh shows a direct increase in consistency under mesh refinement, and the results indicate that the quality of the gradient more than suffices to compute reasonable descent directions if the mesh is fine enough.

### Table 1

<table>
<thead>
<tr>
<th></th>
<th>Mesh 1</th>
<th>Mesh 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cells</strong></td>
<td>8,356</td>
<td>49,648</td>
</tr>
<tr>
<td><strong>Vertices</strong></td>
<td>2,337</td>
<td>11,232</td>
</tr>
<tr>
<td><strong>Variable vertices</strong></td>
<td>698</td>
<td>1,236</td>
</tr>
<tr>
<td>( \ell_2 )-norm SD</td>
<td>0.01404</td>
<td>0.01315</td>
</tr>
<tr>
<td>( \ell_2 )-norm FD</td>
<td>0.01353</td>
<td>0.01276</td>
</tr>
<tr>
<td>( \ell_\infty )-norm SD</td>
<td>0.00378</td>
<td>0.00408</td>
</tr>
<tr>
<td>( \ell_\infty )-norm FD</td>
<td>0.00365</td>
<td>0.00376</td>
</tr>
<tr>
<td>( \ell_\infty )-norm error</td>
<td>0.00050</td>
<td>0.00062</td>
</tr>
<tr>
<td>( \ell_1 )-norm SD</td>
<td>0.11000</td>
<td>0.11293</td>
</tr>
<tr>
<td>( \ell_1 )-norm FD</td>
<td>0.10821</td>
<td>0.11109</td>
</tr>
<tr>
<td>( \ell_1 )-norm error</td>
<td>0.00821</td>
<td>0.00499</td>
</tr>
</tbody>
</table>

### 7. Optimization.

#### 7.1. Preliminaries.
When using a boundary representation for the shape derivative, a steepest descent algorithm for the shape optimization problem at hand can simply be implemented by

\[
\Gamma_{\text{horn}}^{k+1} = \{ x - \epsilon \cdot G(x) \cdot n(x) : x \in \Gamma_{\text{horn}}^k \},
\]

where \( \epsilon \) is the step length of the gradient descent scheme, \( k \) denotes the iteration counter, and the normal derivative of the objective with respect to shape perturbations...
can be identified from expression (52), that is,
\begin{equation}
G(x) = \int_{0}^{T} 2c \ \text{div} \ (p(t,x) \ u^*(t,x)) \ dt.
\end{equation}

Due to the boundary formulation, no additional calculations, such as the potentially computationally costly derivative of the mesh deformation process, are necessary. Any mesh deformation or remeshing approach can be used, and they are interchangeable without the need to consider additional derivatives.

However, there are some geometric considerations that have not been addressed so far. First, we would like to steer the optimization towards smooth updates in order not to be trapped in local optima with bad performance [3] and in order to obtain manufacturable shapes. Second, we would like the horn to attach properly to the waveguide and end at a prescribed mouth shape. The latter condition prevents the optimization from gradually elongating the horn; generally, the longer the horn is, the easier it is to reduce the reflections, particularly in the low end of the target spectrum. Both these conditions can be addressed simultaneously by calculating a smoothed descent direction $G_s$ through the solution of
\begin{equation}
(-\delta \Delta \Gamma + I) \cdot G_s = G \quad \text{on } \Gamma_{\text{horn}},
\end{equation}
\begin{equation}
G_s = 0 \quad \text{on } \partial \Gamma_{\text{horn}},
\end{equation}
where $\Delta \Gamma$ denotes the surface or tangential Laplacian, $I$ is the identity, and $\delta = 0.1$ is a parameter controlling the amount of gradient smoothing. Since the left-side operator in equation (29) is positive definite, $G_s$ is a descent direction if $G$ is. More details on the effect of this gradient smoothing procedure can be found, for example, in [2, 26, 29].

The zero Dirichlet boundary condition in problem (29) is applied for nodes on the throat interface between horn and waveguide as well as for nodes on edges constituting the mouth, fixing those edges as discussed above.

Although the use of $G_s$ in the optimization algorithm yields smooth updates and forces the horn to attach to the throat and the mouth, it does not necessarily lead to a good discretization of the unknown surface, meaning that very often surface nodes start to cluster, thereby creating a poor discretization and low quality discontinuous Galerkin mesh, possibly with inverted surface elements. The descent direction (52) is invariant under tangential modifications of $V$. Therefore, we can use the excess degree of freedom in the tangent plane to reparametrize the discretization of the surface by moving nodes in tangential direction $r_1(x) \cdot \tau_1(x) + r_2(x) \cdot \tau_2(x)$, where $\tau_i(x)$ describes a consistent choice of basis vectors of the tangent plane at $x$ and $r_i(x) \in \mathbb{R}$, such that a locally even spacing of vertices is achieved in each optimization iteration. Such a tangential reparameterization is achieved by successively moving the surface vertices to the centroid of the corresponding Voronoi cell of the dual mesh, projected into the tangent plane. Because the centroidal Voronoi tessellation (CVT) can be interpreted as the dual of the Delaunay triangulation [9, 12], this approach can be thought of as turning the Delaunay remeshing procedure into a mesh deformation scheme by only relocating nodes but not changing connectivity. More information on the reparameterization and mesh deformation approach can be found in [24]. After ensuring the quality of the surface tessellation, the volume mesh is made to match the new horn boundary by solving a noncoupled componentwise Laplace equation for the mesh node displacements with a dampening coefficient of 0.1. Deforming the volume mesh offers several advantages over remeshing, because element connectivities and consequently the memory structure and adjoint data flow remain unchanged.
Summarizing the above, we conduct a gradient descent by first computing the new discretized boundary via the new node positions

\[ \Gamma_{\text{horn}}^k + 1 = \{ x - \epsilon \cdot G(x) \cdot n(x) + r_1(x) \cdot \tau_1(x) + r_2(x) \cdot \tau_2(x) : x \in \Gamma_{\text{horn}}^k \}, \]

and we afterwards make the volume mesh match this boundary by solving uncoupled Laplace equations for the mesh node displacements in the volume. Thus, based on the current surface mesh \( \Gamma_{\text{horn}}^k \), we apply a descent step in the normal direction together with a tangential reparameterization, such that the nodes of \( \Gamma_{\text{horn}}^{k+1} \) adhere to the CVT criterion, making equation (30) partly implicit with respect to determining \( r_i(x) \). This procedure works without an analytic description of the surface, such as B-splines or NURBS, which would be necessary for a redistribution via remeshing, and our scheme works completely on the discretized level.

As a stopping criterion, we use the size of the \( L^2(\Gamma_{\text{horn}}) \)-norm of the smoothed gradient, a condition that yields a first-order necessary condition, including the geometric constraints at throat and mouth, and that also measures the actual deformation not including the tangential repair.

7.2. Numerical results. We demonstrate the feasibility of our methodology by an application to a large-scale horn optimization problem of practical relevance and of a kind that cannot be substituted by a 2D problem. To this end, we consider a horn that blends a circular throat of 19.3 mm radius (same as in the verification) into an elliptic mouth with half-axes 100 mm and 60 mm, respectively. The throat dimension is suitable for mounting a standard 1.5 inch compression driver, a type of sound source that is standard for midrange acoustic horns, and the elliptic mouth is chosen to obtain a priori directivity differences in the horizontal and vertical directions. The blending between throat and mouth is linear for the starting geometry. The sinc function—and therefore the spectrum to optimize for—is the same as for the validation calculations above.

We use a high density mesh surpassing the quality of the finest mesh, “Mesh 2,” from the finite difference and the forward solver verification tests by using 121,631 cells and a total of 25,118 vertices for a physically smaller geometry. Thus, the shortest edge is 1.1984 mm, the longest edge is 19.9922 mm, and the average edge length is 5.8856 mm. Physical dimensions are again the same as in “Mesh 2” from the finite difference validation, except that the mouth of the horn is not circular but ellipsoidal, with the vertical radius of the opening scaled to 60%. This leads to the optimization of 1,762 node positions constituting the mantle surface. Using a time-step of \( 2.5 \times 10^{-6} \) seconds, we operate at roughly 40% of the maximum step length allowed by the CFL criterion. Each simulation then conducts 1,798 time-steps until an end time of \( 4.4975 \times 10^{-3} \) seconds. This amounts to 1,946,096 quarter domain state unknowns per time-step, or 3,499,080,608 unknowns to be determined in total. To compute the adjoint forcing and the shape derivative, we reduce this data to 437,992,800 unknowns by utilizing the boundary representation of the shape derivative, not storing unnecessary volume data. Thus, for each optimization step, we have to be able to handle at least 3.26 GB of double precision data instead of 26 TB for the full trajectory, excluding additional RAM necessary for the preconditioned Krylov solvers needed to solve the implicit system at each time-step. The optimization is conducted fully parallel on 48 cores of a four-node Intel Xeon E5-2630 workstation running at a clock speed of 2.30 GHz, which results in roughly 40 minutes total wall clock time per optimization iteration. Beginning from an objective function value of \( J = 5.875 \times 10^{-7} \) and an initial \( L^2 \)-norm of the smoothed gradient of \( 2.889 \times 10^{-5} \), we need 1,789 optimization steps until the
Isoperimetric, top, and side views of the elliptic test-case. Starting guess to the left, and final shape to the right.

Fig. 5. Isoperimetric, top, and side views of the elliptic test-case. Starting guess to the left, and final shape to the right.

Fig. 6. Left: Magnitude reflection spectra of the initial (dashed lines) as well as the optimized (solid lines) horn. Right: Effective radii of the initial (dashed lines) as well as the optimized (solid lines) horn.

$L^2$-norm of the smoothed gradient falls below $1.5 \cdot 10^{-8}$, terminating at an objective function value of $J = 6.048 \cdot 10^{-9}$. In total, the optimization required a total of roughly 57,248 CPU hours, including parallel overhead. Figure 5 shows the initial (left) as well as the optimized (right) horn shapes.

Magnitude reflection spectra are shown in Figure 6. The spectrum of the optimized horn with its nonintuitive shape is a substantial improvement over the initial geometry over the whole frequency band prescribed by the incoming sinc pulse, with some frequencies actually having a reflection coefficient of almost zero. Figure 6 also shows the effective radii, that is, the radii that would give cylindrically symmetric horns of the same cross-sectional area as the one considered here. The optimized horn’s cross-sectional area grows in the axial direction and avoids the sharp edge at the throat, which the initial horn exhibits. The behavior of the effective radii changes character a few centimeters from the horn mouth (approximately at $z = 0.13$ m) where the...
effective radius quickly increases and then levels out. This is most likely an interesting end effect that stems from the imposed fixed rim of the horn mouth.

8. Discussion and outlook. To the best of our knowledge, this is the first time that numerical shape optimization has been used for detailed design of an acoustic horn using full 3D time domain modeling. The characteristic decomposition of the acoustic power density is a key ingredient in our approach; it is used both to define the objective function for the optimization and to obtain stable numerical fluxes in the numerical method. Another key ingredient is the surface representation of the objective function gradient, which limits the need for storage of time histories of the primal and adjoint variables to layers of the design and inlet boundary. The alternative, to rely on exact gradients to the discrete objective function, would involve the time history of the fields throughout the domain, which would require excessive storage and would also necessitate access to derivatives of the mesh deformation scheme. In particular, for the mesh used here, our use of a surface representation of the gradient implies a data reduction from 26 TB to 3.26 GB. Smoothness requirements on the design updates as well as the fixed throat and mouth geometries are imposed through use of the Laplace–Beltrami operator.

Being able to optimize complex 3D geometries in a time-domain formulation will enable the consideration of more challenging objective functions in the future, such as impedance matching to a specific source and requirements on the far-field radiation pattern.

Appendix A. Sensitivity analysis.
A.1. Basic formulas. Let \( \Omega \) denote an open and bounded domain in \( \mathbb{R}^3 \), and let \( \Gamma \subset \partial \Omega \). We will differentiate integrals of the types

\[
J_1(\Omega) = \int_{\Omega} f \, dx, \quad J_2(\Omega) = \int_{\Gamma} h \, d\Gamma
\]

with respect to domain variations generated by a smooth vector field \( V : \mathbb{R}^3 \to \mathbb{R}^3 \). A family of deformed domains \( \Omega^\epsilon \) and surfaces \( \Gamma^\epsilon \), parametrized by \( \epsilon \geq 0 \), are generated by the formula \( x^\epsilon = x + \epsilon V(x) \) for \( x \in \Omega \) or \( x \in \Gamma \). We consider families of functions \( f^\epsilon \) and \( h^\epsilon \) defined on \( \Omega^\epsilon \) and \( \Gamma^\epsilon \), where \( f = f_0 \) and \( h = h_0 \), and define shape derivatives of integrals (31) as

\[
\begin{align*}
dJ_1(\Omega)[V] &= \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \left( \int_{\Omega^\epsilon} f^\epsilon \, dx - \int_{\Omega} f \, dx \right) := \frac{d}{d\epsilon} \int_{\Omega^0} f \, dx 

dJ_2(\Omega)[V] &= \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \left( \int_{\Gamma^\epsilon} h^\epsilon \, d\Gamma - \int_{\Gamma} h \, d\Gamma \right) := \frac{d}{d\epsilon} \int_{\Gamma^0} h \, d\Gamma 
\end{align*}
\]

if the limits exist. The material derivative of function families \( f^\epsilon \) and \( h^\epsilon \) at \( \epsilon = 0 \) is defined as

\[
\begin{align*}
\dot{f}[V](x) &= \lim_{\epsilon \to 0^+} \frac{f^\epsilon(x + \epsilon V(x)) - f(x)}{\epsilon}, \\
\dot{h}[V](x) &= \lim_{\epsilon \to 0^+} \frac{h^\epsilon(x + \epsilon V(x)) - h(x)}{\epsilon},
\end{align*}
\]

if the limits exist, and the shape derivative of the families is defined as

\[
\begin{align*}
f'[V] &= \dot{f}[V] - \langle V, \nabla f \rangle, \\
h'[V] &= \dot{h}[V] - \langle V, \nabla h \rangle.
\end{align*}
\]
Remark 1. An often used alternative characterization of the shape derivative is the partial derivatives with respect to $\epsilon$; that is, for fixed $x$, we have

\begin{align}
(35a) & \quad f'[V](x) = \lim_{\epsilon \to 0} \frac{f_\epsilon(x) - f(x)}{\epsilon}, \\
(35b) & \quad h'[V](x) = \lim_{\epsilon \to 0} \frac{h_\epsilon(x) - h(x)}{\epsilon},
\end{align}

which can be obtained by a Taylor expansion of $f_\epsilon$ and $h_\epsilon$ in definitions (33). However, a complicating factor with characterizations (35) is that the arguments of $f_\epsilon$ of $h_\epsilon$ in expressions (35) are not necessarily in the domains of definition $\Omega^\epsilon$, $\Gamma^\epsilon$ of the functions.

Sokolowski and Zolésio [28, sect. 2.31] prove the following formula for shape derivative (32a) under the assumptions that $\Omega$ is of class $C^k$, $k \geq 1$, and that $f$, $f'[V]$, and $\|\nabla f\|$ are all in $L^1(\Omega)$:

\begin{equation}
(36) \quad dJ_1(\Omega)[V] = \int_\Omega f'[V] + \text{div}(Vf) \ dx = \int_\Omega f'[V] \ dx + \int_{\partial\Omega} \langle V,n \rangle f \ d\Gamma.
\end{equation}

We will consider shape derivative (32b) in the following three cases:

(i) $(V,n) \equiv 0$ on $\Gamma \subset \partial\Omega$.

(ii) $h(x) = \langle \tilde{h}(x), n(x) \rangle$, $\Gamma = \partial\Omega$, $n$ is the outward-directed normal field on $\Gamma$, and $\tilde{h}$ is the trace of a vector-valued function in $\Omega$ whose divergence satisfies the conditions for $f$ assumed for formula (36).

(iii) Conditions as for case (ii), but $\Gamma \subset \partial\Omega$ such that $\langle V,n \rangle \equiv 0$ on $\partial\Omega \setminus \Gamma$.

For these cases, the formula

\begin{equation}
(37) \quad dJ_2(\Omega)[V] = \begin{cases} 
\int_\Gamma h'[V] \ d\Gamma & \text{in case (i),} \\
\int_\Gamma \langle \tilde{h}'[V], n \rangle + \langle V,n \rangle \text{div} \tilde{h} \ d\Gamma & \text{in cases (ii) and (iii)}
\end{cases}
\end{equation}

holds, where $\text{div} \tilde{h}$ inside the boundary integral denotes the trace on $\Gamma$ of the divergence of $\tilde{h}$ in $\Omega$. In case (i), $\Gamma^\epsilon = \Gamma$, so the formula follows immediately by definitions (33b) and (34b) as long as $h'[V]$ exists and is integrable. In case (ii), the divergence theorem and the substitution $f = \text{div} \tilde{h}$ turn integral $J_2$ into a domain integral of $J_1$ type. Formula (37) then follows from expression (36) using the divergence theorem and the fact that the shape derivative and the spatial gradient commute. The formula for case (iii) follows from cases (i) and (ii).

Remark 2. Due to the particular integral form of the state equation used in Appendix A.2, the conditions of cases (i) and (ii) will be satisfied, and shape derivative formulas (36) and (37) are all that are needed. However, if the hard wall boundary condition on $\Gamma_{\text{borne}}$ is imposed in a different way, for instance through an upwinding flux together with a remote mirrored state, as discussed in section 5, the linear structure assumed in cases (ii) and (iii) does not hold, and a formula admitting a more general form of the integrand $h$ in integral $J_2$ is needed. A more general form is obtained by considering a family, parameterized by $\epsilon$, of functions $\tilde{h}_\epsilon$ defined on $\Omega^\epsilon \times \Omega^\epsilon$, and defining the integrand in $J_2$ to be $h_\epsilon(x_\epsilon) = \tilde{h}_\epsilon(x_\epsilon, n_\epsilon(x_\epsilon))$ for $x_\epsilon \in \Gamma^\epsilon$. The shape derivative formula then becomes

\begin{equation}
(38) \quad dJ_2(\Omega)[V] = \int_\Gamma \tilde{h}'[V] + \langle V,n \rangle \left( \langle \nabla \tilde{h}, n \rangle + \kappa(\tilde{h} - \langle V,\tilde{h} \rangle) + \text{div}_\Gamma (\nabla_\gamma \tilde{h}) \right) \ d\Gamma,
\end{equation}

© 2016 Stephan Schmidt, Eddie Wadbro, Martin Berggren
where \( \kappa \) is the local mean curvature, \( \nabla \) and \( \text{div} \) the gradient and the tangential divergence with respect to the first argument of \( \hat{h} \), and \( \nabla_p \) the gradient with respect to the second argument. Formula (38) is a special case of Lemma 3.3.14 in the first author’s Ph. D. thesis [23]. To the best of our knowledge, this formula does not appear anywhere else in the open literature.

### A.2. Shape derivatives and the adjoint system.

We consider the objective function (9), where \( u, p \) satisfy initial–boundary-value problem (6). We require that the vector field \( V \) that generates the domain variations, as described above, vanishes on \( \partial \Omega \setminus \Gamma_{\text{horn}} \). This field \( V \) will generate a family of perturbed domains \( \Omega^\epsilon \) for which the horn boundary \( \Gamma_{\text{horn}}^\epsilon \) is perturbed. However, the rest of the boundary, \( \partial \Omega \setminus \Gamma_{\text{horn}}^\epsilon \), will be fixed independent of \( \epsilon \).

Differentiating objective function (9), using formula (37), and utilizing that \( V \) vanishes on \( \Gamma_{\text{in}} \), we get

\[
(39) \quad dJ(\Omega)[V] = \int_0^T \int_{\Gamma_{\text{in}}} (p + c(u, n))(p'[V] + c(u'[V], n)) \, d\Gamma \, dt.
\]

In order to continue, we need to determine how the shape derivatives \( p'[V], u'[V] \) in expression (39) depend on \( V \). We will therefore differentiate the state equation with respect to domain perturbations. To prepare for the use of formulas (36) and (37), we first rewrite state equation (6) in the integral form

\[
(40) \quad c \int_\Omega \langle v, \partial_t u + \nabla p \rangle \, dx + \frac{1}{c} \int_\Omega q(\partial_t p + c^2 \text{div} u) \, dx - c \int_\Gamma_v q(u, n) \, d\Gamma + \frac{1}{2} \int_{\Gamma_{\text{inout}}} (q - c(v, n))(p - c(u, n)) \, d\Gamma = \int_{\Gamma_{\text{in}}} (q - c(v, n))g \, d\Gamma,
\]

where \( \Gamma_v = \Gamma_{\text{wall}} \cup \Gamma_{\text{symm}} \) and \( \Gamma_{\text{inout}} = \Gamma_{\text{in}} \cup \Gamma_{\text{out}} \), and where \( v \) and \( q \) are arbitrary smooth test functions.

**Remark 3.** It is immediate, by inspection, that integral form (40) is consistent with system (6); that is, solutions to system (6) satisfy expression (40) for any smooth test functions. Moreover, integral form (40) is designed to satisfy a stability property, namely that the acoustic energy decreases with time when no input is provided to the systems. That is, for \( g = 0 \),

\[
(41) \quad \frac{c}{2} \frac{d}{dt} \int_\Omega |u|^2 \, dx + \frac{1}{2c} \frac{d}{dt} \int_\Omega p^2 \, dx + \frac{1}{2} \int_{\Gamma_{\text{inout}}} (p^2 + c^2(u, n)^2) \, d\Gamma = 0,
\]

which can be seen by substituting \( v = u, q = p \) into equation (40), using the product rule, and integrating by parts.

Let \( v = u^*, q = -p^* \) in (40) for some arbitrary smooth functions \( u^* \) and \( p^* \) (these will later be chosen to satisfy the adjoint equation). Integrating in time over an interval \((0, T)\) yields

\[
(42) \quad 0 = c \int_0^T \int_\Omega \langle u^*, \partial_t u + \nabla p \rangle \, dx \, dt - \frac{1}{c} \int_0^T \int_\Omega p^*(\partial_t p + c^2 \text{div} u) \, dx \, dt
\]

\[+ c \int_0^T \int_{\Gamma_v} p^*(u, n) \, d\Gamma \, dt - \frac{1}{2} \int_0^T \int_{\Gamma_{\text{inout}}} (p^* + c(u^*, n))(p - c(u, n)) \, d\Gamma \, dt
\]

\[+ \int_0^T \int_{\Gamma_{\text{in}}} (p^* + c(u^*, n))g \, d\Gamma \, dt.
\]
Now assume that the domain $\Omega$ is perturbed by domain variations generated by a vector field $V$, chosen as described above. We denote by subscript $\epsilon$ on the state and costate variables the solutions associated with the perturbed domain $\Omega$. Differentiating equation (42) with respect to a domain variation generated by $V$ results in

$$
0 = c \frac{d}{d\epsilon} \left. \int_0^T \int_{\Omega} (u^{\epsilon}_r, \partial_t u^{\epsilon} + \nabla p^{\epsilon}) \, dx \, dt \right|_{\epsilon=0} 
- \frac{1}{c} \frac{d}{d\epsilon} \left. \int_0^T \int_{\Omega} p^{\epsilon}_r(\partial_t p^{\epsilon} + c^2 \text{div} \, u^{\epsilon}) \, dx \, dt \right|_{\epsilon=0} 
+ \frac{1}{2} \frac{d}{d\epsilon} \left. \int_0^T \int_{\Gamma_{\text{inout}}} (p^{\epsilon}_r + c(u^{\epsilon}_r, n))(p^{\epsilon} - c(u^{\epsilon}_r, n)) \, d\Gamma \, dt \right|_{\epsilon=0} 
+ \frac{d}{d\epsilon} \left. \int_0^T \int_{\Gamma_{\text{in}}} (p^{\epsilon}_r + c(u^{\epsilon}_r, n))g \, d\Gamma \, dt \right|_{\epsilon=0}.
$$

(43)

Applying formulas (36) and (37), where case (i) applies to $\Gamma_{\text{inout}}$ and $\Gamma_{\text{in}}$ and case (iii) to $\Gamma_s$, together with the product rule of differentiation and the fact that the shape derivative commutes with time and spatial differentiation, yields that expression (43) expands to

$$
0 = c \int_0^T \int_{\Omega} (u^{\epsilon}_r, \partial_t u + \nabla p) \, dx \, dt 
- \frac{1}{c} \int_0^T \int_{\Omega} p^{\epsilon}_r(\partial_t p + c^2 \text{div} \, u) \, dx \, dt 
+ c \int_0^T \int_{\Gamma_s} p^{\epsilon}(u, n) \, d\Gamma \, dt 
- \frac{1}{2} \int_0^T \int_{\Gamma_{\text{inout}}} (p^{\epsilon} + c(u^{\epsilon}, n))(p - c(u, n)) \, d\Gamma \, dt 
+ \int_0^T \int_{\Gamma_{\text{in}}} (p^{\epsilon} + c(u^{\epsilon}, n))g \, d\Gamma \, dt 
+ c \int_0^T \int_{\Omega} (u^{\epsilon}, \partial_t u' + \nabla p') \, dx \, dt 
- \frac{1}{c} \int_0^T \int_{\Omega} p^{\epsilon}(\partial_t p' + c^2 \text{div} \, u') \, dx \, dt 
+ c \int_0^T \int_{\Gamma_s} p^{\epsilon}(u', n) \, d\Gamma \, dt 
- \frac{1}{2} \int_0^T \int_{\Gamma_{\text{inout}}} (p^{\epsilon} + c(u', n))(p' - c(u', n)) \, d\Gamma \, dt 
+ c \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle (u^{\epsilon}, \partial_t u + \nabla p) \, d\Gamma \, dt 
- \frac{1}{c} \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle p^{\epsilon}(\partial_t p + c^2 \text{div} \, u) \, d\Gamma \, dt 
+ c \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \text{div} (up^{\epsilon}) \, d\Gamma \, dt,
$$

(44)

where the dependency on $V$ of the shape derivatives has been suppressed for brevity.

The first five terms in expression (44) vanish identically since $u$, $p$ satisfy expression (42). Moreover, provided that $u$, $p$, and $\Gamma_{\text{horn}}$ are regular enough, terms 10 and 11 in expression (44) will also vanish identically, due to state equation (6). (A sufficient regularity condition is that $u$ and $p$ as well as their derivatives can be continuously...
extended up to $\Gamma_{\text{horn}}$. Reducing these terms from expression (44) yields

$$0 = c \int_0^T \int_{\Omega} \langle u^*, \partial_t u' + \nabla p' \rangle \, dx \, dt - \frac{1}{c} \int_0^T \int_{\Omega} p^* (\partial_t p' + c^2 \text{div} u') \, dx \, dt$$

(45) $$+ c \int_0^T \int_{\Gamma_s} p^* \langle u', n \rangle \, d\Gamma \, dt - \frac{1}{2} \int_0^T \int_{\Gamma_{\text{inout}}} (p^* + c\langle u^*, n \rangle)(p' - c\langle u', n \rangle) \, d\Gamma \, dt$$

$$+ c \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \text{div} (up^*) \, d\Gamma \, dt.$$

By integrating $\langle u^*, \partial_t u' \rangle$ by parts in time and integrating $\langle u^*, \nabla p' \rangle$ by parts in space, we can write the first term in expression (45) as

$$c \int_0^T \int_{\Omega} \langle u^*, \partial_t u' + \nabla p' \rangle \, dx \, dt = c \int_0^T \int_{\Omega} \langle u'(T), u^*(T) \rangle \, dx \, dt - c \int_0^T \int_{\Omega} \langle u', \partial_t u^* \rangle \, dx \, dt$$

(46) $$+ c \int_0^T \int_{\partial\Omega} p^* \langle u^*, n \rangle \, d\Gamma \, dt - c \int_0^T \int_{\Omega} p' \text{div} u^* \, dx \, dt,$$

where we have used that $u^'|_{t=0}$ and $p^'|_{t=0}$ vanish due to the homogeneous initial conditions for state equation (6). Similarly, by integrating $p^* \partial_t p'$ by parts in time and integrating $p^* \text{div} u'$ by parts in space, the second term in expression (45) becomes

$$\frac{1}{c} \int_0^T \int_{\Omega} p^* (\partial_t p' + c^2 \text{div} u') \, dx \, dt = \frac{1}{c} \int_0^T \int_{\Omega} p'(T)p^*(T) \, dx \, dt - \frac{1}{c} \int_0^T \int_{\Omega} p' \partial_t p^* \, dx \, dt$$

(47) $$+ c \int_0^T \int_{\partial\Omega} \langle u', n \rangle p^* \, d\Gamma \, dt - c \int_0^T \int_{\Omega} \langle u', \nabla p^* \rangle \, dx \, dt.$$

Substituting expressions (46) and (47) into expression (45) and recombining the terms using that $\partial\Omega = \Gamma_s \cup \Gamma_{\text{inout}}$, we find that

$$0 = c \int_0^T \int_{\Omega} \langle u'(T), u^*(T) \rangle \, dx \, dt + c \int_0^T \int_{\Omega} \langle u', -\partial_t u^* + \nabla p^* \rangle \, dx \, dt$$

$$- \frac{1}{c} \int_0^T \int_{\Omega} p'(T)p^*(T) \, dx \, dt - \frac{1}{c} \int_0^T \int_{\Omega} p' (-\partial_t p^* + c^2 \text{div} u^*) \, dx \, dt$$

(48) $$+ c \int_0^T \int_{\Gamma_s} p'(u^*, n) \, d\Gamma \, dt - \frac{1}{2} \int_0^T \int_{\Gamma_{\text{inout}}} (p^* - c\langle u^*, n \rangle)(p' + c\langle u', n \rangle) \, d\Gamma \, dt$$

$$+ c \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \text{div} (up^*) \, d\Gamma \, dt.$$

So far functions $p^*$, $u^*$ have been arbitrary. However, if we assume that they satisfy
the adjoint system
\[- \frac{\partial u^*}{\partial t} + \nabla p^* = 0 \quad \text{in } \Omega \text{ for } t < T,\]
\[- \frac{\partial p^*}{\partial t} + c^2 \text{div } u^* = 0 \quad \text{in } \Omega \text{ for } t < T,\]
\[
\frac{1}{2} \left( p^* - c \langle u^*, n \rangle \right) = \frac{1}{2} \left( p + c \langle u, n \rangle \right) \quad \text{on } \Gamma_{\text{in}} \text{ for } t < T,
\]
\[
\frac{1}{2} \left( p^* - c \langle u^*, n \rangle \right) = 0 \quad \text{on } \Gamma_{\text{out}} \text{ for } t < T,
\]
\[
\langle u^*, n \rangle = 0 \quad \text{on } \Gamma_{\text{wall}} \cup \Gamma_{\text{symm}} \text{ for } t < T,
\]
\[
u^* \equiv 0 \quad \text{in } \Omega \text{ at } t = T,
\]
\[
p^* \equiv 0 \quad \text{in } \Omega \text{ at } t = T,
\]
expression (48) reduces to
\[
\frac{1}{2} \int_0^T \int_{\Gamma_{\text{horn}}} (p + c \langle u, n \rangle) \left( p' + c \langle u', n \rangle \right) d\Gamma dt = c \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \text{div} (up^*) d\Gamma dt,
\]
which, combined with expression (39), means that the shape derivative can be given by the integral representation
\[
dJ(\Gamma_{\text{horn}})[V] = 2c \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \text{div} (up^*) d\Gamma dt.
\]

Remark 4. The shape derivative representation can also be written as
\[
dJ(\Gamma_{\text{horn}})[V] = 2c \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \text{div} (u^* p) d\Gamma dt.
\]

This alternative expression can be derived as follows:
\[
\int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \text{div} (up^*) d\Gamma dt = \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \left( \langle u, \nabla p^* \rangle + p^* \text{div } u \right) d\Gamma dt
\]
\[
= \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \left( \langle u, \partial_t u^* \rangle - \frac{1}{c^2} p^* \partial_t p \right) d\Gamma dt
\]
\[
= - \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \left( \langle \partial_t u, u^* \rangle - \frac{1}{c^2} \partial_t p^* p \right) d\Gamma dt
\]
\[
= \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \left( \langle \nabla p, u^* \rangle + p \text{div } u^* \right) d\Gamma dt = \int_0^T \int_{\Gamma_{\text{horn}}} \langle V, n \rangle \text{div} (pu^*) d\Gamma dt,
\]
where the state (6) and adjoint (49) equations have been used in the second and fourth equalities, and integration by part in time in the third equality, where also the homogeneous initial (or terminal) conditions for \( u, u^*, p, \) and \( p^* \) have been exploited.

REFERENCES

THREE-DIMENSIONAL HORN OPTIMIZATION


© 2016 Stephan Schmidt, Eddie Wadbro, Martin Berggren


Shape derivatives for the compressible Navier–Stokes equations in variational form

Matthias Sonntag\textsuperscript{a,*}, Stephan Schmidt\textsuperscript{b}, Nicolas R. Gauger\textsuperscript{a}

\textsuperscript{a} Chair for Scientific Computing, Technische Universität Kaiserslautern, 67663 Kaiserslautern, Germany
\textsuperscript{b} University of Würzburg, Emil-Fischer-Str. 30, 97074 Würzburg, Germany

\begin{abstract}
Shape optimization based on surface gradients and the Hadamard-form is considered for a compressible viscous fluid. Special attention is given to the difference between the “function composition” approach involving local shape derivatives and an alternate methodology based on the weak form of the state equation. The resulting gradient expressions are found to be equal only if the existence of a strong form solution is assumed. Surface shape derivatives based on both formulations are implemented within a Discontinuous Galerkin flow solver of variable order. The gradient expression stemming from the variational approach is found to give superior accuracy when compared to finite differences.
\end{abstract}

\section{Introduction}
Shape optimization is a research field that has received much attention in the past. In general, any problem where the boundary of the domain is part of the unknown can be considered a shape optimization problem. In most applications, the physics are modeled by partial differential equations, making shape optimization a special sub-class within the field of PDE-constrained optimization. Usually, the derivation of the sensitivities and adjoint equations follows a function composition approach, i.e. some set of design variables defines the geometry and within this geometry the PDE is solved, thereby generating the state variables that enter the objective function \cite{1–3}. Therefore, the necessity to consider sensitivities or derivative information with respect to the geometry adds additional complexity to the shape optimization problem when compared to general PDE-constrained optimization. Because it is often not immediately clear how to compute these “mesh sensitivities”, that is the variation of the PDE with respect to a change in the geometry, there is often a strong desire for a very smooth parameterization of the domain with as few design parameters as possible. Although there have been successful attempts to incorporate problem structure exploitations in order to efficiently compute these partial derivatives for very large problems, such as differentiating the entire design chain at once or by considering the adjoint process of the mesh deformation \cite{4,5}, very often one is still forced into finite differencing, which means the PDE residual at steady state has to be evaluated on meshes that have been perturbed by a variation in each design parameter of the shape, a process that makes large scale optimization usually prohibitive. This negates some of the advantages of the adjoint approach, such as

\footnotesize
\textsuperscript{*} Corresponding author.
\textsuperscript{E-mail addresses: matthias.sonntag@rhrk.uni-kl.de (M. Sonntag), stephan.schmidt@mathematik.uni-wuerzburg.de (S. Schmidt), nicolas.gauger@rhrk.uni-kl.de (N.R. Gauger).}

http://dx.doi.org/10.1016/j.cam.2015.09.010
0377-0427/© 2015 Elsevier B.V. All rights reserved.
the independence of the number of design parameters. More severely, it also makes fast optimization strategies such as the one-shot approach [6–8] somewhat unattractive in terms of wall-clock-time.

A more recent trend to overcome the cumbersome computation of these geometric sensitivities is the use of shape calculus. Shape calculus summarizes the mathematical framework used when considering problems where the shape is the unknown in the continuous setting. Manipulations in the tangent space of the unknown object can be used to circumvent any necessity of knowing discrete geometric sensitivities, because these can be directly included in a surface gradient expression on the continuous level. More details on this theoretical framework can be found in [9, 10]. Traditionally, this methodology was primarily used to address the very difficult question of existence and uniqueness of optimal shapes [11], but more recently this methodology has also been used in very large scale aerodynamic design and computational optimization [12–14]. In [15], for example, the complete optimization of a blended wing–body aircraft in a compressible inviscid fluid is considered. Because this approach solely relies on the problem formulation in the continuous setting and only afterwards discretizes the continuous boundary integral expressions for the shape derivative, great care must be taken when making the initial assumptions and when implementing the respective continuous expressions, especially at singular points in the geometry, such as the trailing edge of an airfoil [16]. Because this approach is indeed truly independent of the number of design parameters, it enables the most detailed possible parameterization, that is using all surface mesh nodes as design unknowns. This is sometimes called “free node parameterization”. However, these highly detailed shape parameterizations usually lack any kind of inherent regularity preservation and as such, one usually finds this approach paired with some sort of smoothing procedure that projects or embeds the respective optimization iterations into a desired regularity class, which can nicely be paired with an SQP or Newton-type optimization scheme, which is sometimes also called a “Sobolev Method” [17, 18].

As part of this work, we study how to further increase the accuracy of shape derivatives when used within viscous compressible aerodynamic design optimization. Within applied aerodynamic shape optimization, it is customary to exploit the above mentioned function composition approach in order to derive and implement the adjoint equation and gradient expression. This has been used with great success, both within the context of continuous and discrete adjoint based aerodynamic shape optimization [19, 20] and general shape optimization [21]. However, common to these approaches is the assumption that the state equation has a strong form solution and each of the steps within the shape differentiation process of the function composition exists, which usually means the existence of so-called local shape derivatives. For elliptic problems, this existence can usually be shown, making the above mentioned approach somewhat of an established procedure, see for example Chapter 3.3 in [10]. However, for the hyperbolic equations governing some compressible fluids, the existence of a strong form solution is not clear. Rather, in the presence of shock waves and discontinuities in the flow, one can usually only expect the variational form of the equation to hold, a property which is very often not taken into account when studying the derivative. Shape differentiation of problems governed by PDEs in weak or variational forms are not very often considered in the literature, except in [23] and especially in [24], where the incompressible Navier–Stokes equations are considered for this purpose from a rigorous theoretical standpoint. Thus, we revisit the shape optimization problem previously considered in [25], but the gradient is derived using elements of the variational approach as shown in [24]. Furthermore, we simultaneously follow the function composition approach, outlining the exact differences comparing these two approaches. One can nicely see how both methodologies reduce to the same gradient expression when assuming the existence of a strong from solution of the state equation. We conclude with a numerical error analysis based on comparing finite differencing with either implementation, demonstrating the higher accuracy of the gradient formulation based on the variational form of the compressible Navier–Stokes equations.

The structure of the paper is as follows. In Section 1, we begin by recapitulating the compressible Navier–Stokes equations in both strong and variational form. Next, Section 2 serves as an introduction and quick overview of shape calculus, including shape derivatives and the Hadamard or Hadamard–Zolésio Structure Theorem, which leads to a preliminary form of the shape derivative of the aerodynamic cost functions. The next section, Section 3, is used to work out the differences between the shape derivative of the compressible Navier–Stokes equations stemming from either the function composition or the variational approach. In Section 4, we then summarize the idea of adjoint calculus. This is used to differentiate the Navier–Stokes equations, thereby discussing the Hadamard form of the respective objective functions both for the strong as well as the variational form of the state constraint. Finally, in the last section, numerical results achieved with both methods are compared to shape derivatives computed by finite differences, showing a considerable gain in accuracy when using shape derivatives based on the variational form.

1. Fluid mechanics

1.1. Flow domain and boundary conditions

In the following $\rho, v = (v_1, v_2)^T, p, E$ and $T$ denote the density, velocity, pressure, total energy and temperature. The domain of the fluid is denoted by $\Omega$, with wall and far-field boundaries $\Gamma_W$ and $\Gamma_\infty$. At the wall $\Gamma_W$, the no-slip boundary condition $v = 0$ is imposed for the velocity. With respect to temperature, either the isothermal boundary condition $T = T_W$ or the adiabatic boundary condition $\nabla T \cdot n = 0$ holds. The isothermal and adiabatic parts of the wall are named $\Gamma_{iso}$ and $\Gamma_{adia}$ and we assume $\Gamma_W = \Gamma_{iso} \cup \Gamma_{adia}$ disjoint.
Furthermore $\kappa$, $e$, $H$, $\mu$, and $\gamma$ denote the thermal conductivity, the internal energy, the enthalpy, the viscosity and the adiabatic exponent. The relation $T\kappa = \frac{\rho u^2}{M} (E - \frac{1}{2} \|v\|^2)$ is fulfilled for the temperature, where $Pr$ is the Prandtl number.

### 2.1. Definition of the shape derivative and the Hadamard theorem

In this subsection, we state the Navier–Stokes equations in both strong and weak form. As discussed later, the shape derivative of the aerodynamic cost functions differs depending on which form of the Navier–Stokes equations is used. Using the viscous stress tensor $\tau$, defined by

$$\tau = \mu \left( \nabla v + (\nabla v)^\top - \frac{2}{3} (\nabla \cdot v) I \right),$$

the compressible Navier–Stokes equations in strong form are given by

$$\nabla \cdot (F^c(u) - F^v(u, \nabla u)) \equiv \sum_k \left( \frac{\partial}{\partial x_k} f_k^c(u) - \frac{\partial}{\partial x_k} f_k^v(u, \nabla u) \right) = 0 \text{ in } \Omega,$$

where $u$ denotes the vector of conserved variables, $F^c = (f_1^c, f_2^c)$ the convective fluxes

$$u = \left( \begin{array}{c} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho H^E \end{array} \right), \quad f_1^c(u) = \left( \begin{array}{c} \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ \rho H v_1 \end{array} \right), \quad f_2^c(u) = \left( \begin{array}{c} \rho v_2 \\ \rho v_2^2 + p \\ \rho v_1 v_2 \\ \rho H v_2 \end{array} \right),$$

and $F^v = (f_1^v, f_2^v)$ denotes the viscous fluxes

$$f_1^v(u, \nabla u) = \left( \begin{array}{c} 0 \\ \tau_{11} v_1 + \kappa \frac{\partial T}{\partial x_1} \\ \tau_{21} v_2 + \kappa \frac{\partial T}{\partial x_2} \\ \sum_j \tau_{1j} v_j + \kappa \frac{\partial T}{\partial x_1} \end{array} \right), \quad f_2^v(u, \nabla u) = \left( \begin{array}{c} 0 \\ \tau_{12} v_1 + \kappa \frac{\partial T}{\partial x_1} \\ \tau_{22} v_2 + \kappa \frac{\partial T}{\partial x_2} \\ \sum_j \tau_{2j} v_j + \kappa \frac{\partial T}{\partial x_2} \end{array} \right).$$

Furthermore, temperature $T$ and pressure $p$ are linked to the state variables using the perfect gas assumption, that is

$$p = \rho RT = \rho R \frac{E - \frac{1}{2} \|v\|^2}{c_v} = \frac{R}{c_v} \rho \left( E - \frac{1}{2} \|v\|^2 \right),$$

where $e = c_v T$, $E = e + \frac{1}{2} \|v\|^2$ and $c_v$ denotes the heat capacity of the gas at constant volume.

Furthermore, the compressible Navier–Stokes equations in variational form are given by the following:

**Definition 1.1 (Variational Form of the Navier–Stokes Equations).** We assume that $\mathcal{H} := H^1 \times H^2 \times H^3 \times H^4$, where $H^i$ is a suitable Hilbert-Space. Multiplication of the pointwise Navier–Stokes (2) with an arbitrary test function $\psi \in \mathcal{H}$ and integration by parts results in the problem to find $u \in \mathcal{H}$, such that

$$\langle F(u, \Omega), \psi \rangle_{\mathcal{H}^* \times \mathcal{H}} := - \langle F^c(u) - F^v(u, \nabla u), \nabla \psi \rangle_{\Omega} + (n \cdot (F^c(u) - F^v(u, \nabla u)), \psi)_{\Gamma} = 0 \quad \forall \psi \in \mathcal{H}$$

with the boundary conditions

$$\begin{aligned}
\left( \begin{array}{c} v_1 \\ v_2 \end{array} \right) &= 0 \quad \text{on } I_W, \\
\nabla T \cdot n &= 0 \quad \text{on } I_{ad}, \\
T &= T_{wall} \quad \text{on } I_{inj}.
\end{aligned}$$

**Definition 1.2 (Cost Function).** The cost functions under consideration are the lift and drag coefficients, given by

$$J(u) = \frac{1}{C_\infty} \int_{I_W} (pn - \tau n) \cdot \psi \ ds,$$

where $C_\infty$ is a constant and $\psi$ is either $\psi_l = (-\sin(\alpha), \cos(\alpha))^\top$ for the lift or $\psi_d = (\cos(\alpha), \sin(\alpha))^\top$ for the drag coefficient.

### 2. Shape calculus

#### 2.1. Definition of the shape derivative and the Hadamard theorem

In this section the concept of shape derivatives and especially the Hadamard Theorem, as stated in [9, 10], are summarized. Let therefore $D$, the so-called hold-all, be an open set in $\mathbb{R}^d$ and the domain $\Omega$ be a measurable subset of $D$. For vector fields
$V \in C^k_0(D; \mathbb{R}^d)$ the perturbation of identity

$T_\varepsilon[V] : D \times [0, \varepsilon) \to \mathbb{R}^d, \ (x, t) \mapsto x + tV(x)$

is a common approach to describe a deformation $\Omega_\varepsilon = T_\varepsilon[V](\Omega)$ of the domain $\Omega$. With such a deformation, the shape derivative of a domain functional $J(\Omega)$ at $\Omega$ in the direction of a vector field $V \in C^k_0(D; \mathbb{R}^d)$ is defined as the Eulerian derivative

$$dJ(\Omega; V) := \lim_{\varepsilon \to 0} \frac{J(\Omega_\varepsilon) - J(\Omega)}{\varepsilon}.$$  

The functional $f(\Omega)$ is called shape differentiable at $\Omega$ if this Eulerian derivative exists for all directions $V$ and the mapping $G(\Omega) : C^k_0(D; \mathbb{R}^d) \to \mathbb{R}, \ V \mapsto dJ(\Omega; V)$ is linear and continuous.

Theorem 2.1 (Hadamard Theorem, Hadamard Formula). For every domain $\Omega \subset D$ of class $C^k$, let $f(\Omega)$ be a shape differentiable function. Furthermore let the boundary $\Gamma$ be of class $C^{k-1}$, there exists the following scalar distribution $g(\Gamma) \in C_0^k(\Gamma)^*$, such that the shape gradient $G(\Omega) \in C_0^k(\Omega; \mathbb{R}^d)^*$ of $f(\Omega)$ is given by

$$G(\Omega) = g(\Gamma),$$

where $g(\Gamma) \in C_0^k(\Omega; \mathbb{R}^d), C_0^k(\Gamma; \mathbb{R}^d)$ and $g^\ast(\Gamma)$ denote the trace operator and its adjoint operator. In this situation, one can show that

$$dJ(\Omega; V) = dJ(\Gamma; V \cdot n) = (g, V \cdot n)^\ast \in C_0^k(\Omega; \mathbb{R}^d).$$

If $g(\Gamma)$ is integrable over $\Gamma$, then the Hadamard Formula

$$dJ(\Omega; V) = \int_{\Gamma} \langle V \cdot n \rangle g(\Gamma) \ ds$$

is fulfilled. In the following we will call terms that are of the structure $\langle (V \cdot n) \ldots \rangle$ to be in Hadamard form.

Proof. A proof can for example be found in [9] or in [10]. □

Remark 2.2. Assuming the boundary $\Gamma_W$ is of sufficient regularity such that the tangential component only describes a reparameterization but no actual change of the shape, then the shape derivative $dJ(\Omega; V)$ only depends on the normal component $(V \cdot n)n$ of the vector field $V$, which will later eliminate certain normal components within the derivation of the surface gradient expression of the cost functions. Further studies of the ramifications of this assumption can be found in [16].

Definition 2.3 (Material Derivative/Local Shape Derivative). The total derivative

$$df(\varepsilon)(x) := \left. \frac{d}{dt} \right|_{t=0} f(t, T_\varepsilon(x))$$

of $f$ is called the material derivative. Furthermore, the partial derivative

$$f'(x) := f'[V](x) := \frac{\partial}{\partial t} f(t, x)$$

is called the local shape derivative of $f$. 
Remark 2.4. The material and the local shape derivative are related to each other by the chain rule if both exist, i.e.
\[ d\psi[f] (x) = f'(V)(x) + \nabla f (0, x) \cdot V (x) = f' + \nabla f \cdot V, \]
where we used \( \frac{d}{dt} \big|_{t=0} T_i (x) \) for the perturbation of identity. If the geometry is as such that the local shape derivative does not exist, a sharp convex corner for example, one usually accepts the above formula as a definition of the local shape derivative instead.

2.2. Tangential calculus

Before we state the shape derivative of volume and boundary integrals, we give a minimal summary of tangential calculus, which will later be used to derive a preliminary shape derivative of the lift and drag coefficients. This is also required to find the local shape derivative of quantities fulfilling Neumann boundary conditions. A more detailed discussion on tangential calculus can be found in [9, Chapter 8, Section 5].

Let \( f \in C^1 (\Gamma^t) \) be a function with a \( C^1 \)-extension \( F \) into a tubular neighborhood of \( \Gamma^t \). The tangential gradient is the ordinary gradient minus its normal component, i.e.
\[ \nabla^t f := \nabla F \big|_{\Gamma^t} - \frac{\partial F}{\partial n} n. \] (7)

Analogously, the tangential divergence of a smooth vector field \( W \in (C^1 (\Gamma^t))^d \cap (C^1 (\Omega))^d \) is defined by
\[ \text{div}^t W := \text{div} W - DW n \cdot n. \] (8)

For \( f \) and \( W \) as defined above, the tangential Green’s formula is given by
\[ \int_{\Gamma^t} W \cdot \nabla^t f \, ds = \int_{\Gamma^t} f K (W \cdot n) - f \left| \text{div}^t W \right| \, ds, \] (9)
where \( K := \text{div}^t n \) denotes the sum of the principal curvatures, the so called additive curvature, or \((d-1)\) times the mean curvature. A proof can be found in [9, Chapter 8].

2.3. Shape derivative for volume and boundary integrals

In this subsection, we recapitulate shape derivatives of general volume and boundary integrals, as they can be found in [9] for example. Furthermore, we show a preliminary shape derivative of the drag and lift coefficients, which is later transformed into Hadamard form using adjoint calculus. The shape derivative of a general volume cost function \( J (\Omega) = \int_{\Omega} f (x) \, dx \) is given by
\[ df (\Omega; V) = \int_{\Omega} f' \, dx + \int_{\Gamma} (V \cdot n) f \, ds \] (10)
and for a general boundary cost function \( f (\Omega) = \int_{\Gamma} f \, ds \) the shape derivative fulfills
\[ df (\Omega; V) = \int_{\Gamma} f' + (V \cdot n) \left( \frac{\partial f}{\partial n} + K f \right) \, ds. \] (11)

Furthermore, if the vector field \( V \) is orthogonal to the boundary \( \Gamma^t \), the material derivative of the normal vector fulfills
\[ d\psi [n] = - \nabla^t (V \cdot n), \] (12)
which can be found in [26].

Theorem 2.5 (Preliminary Shape Derivative of the Cost Functional). If the vector field of the perturbation of identity fulfills \( V = 0 \) in the neighborhood of the farfield boundary \( \Gamma^\infty \), then the shape derivative of the lift and drag coefficients, Eq. (5), is given by
\[ df (\Omega; V) = \frac{1}{C^\infty} \int_{\Gamma^W} \left( p'n - \tau' n \right) \cdot \psi + (V \cdot n) \left( \frac{\partial f}{\partial n} + K f \right) \, ds. \] (13)

Proof. The proof of the above expression is not fully straightforward, because the objective function (5) depends on the local normal, which first needs to be extended into a tubular neighborhood as discussed within the subsection on tangential calculus. Following the same argumentation as in [26], let \( \mathcal{N} \) be an extension of the normal \( n \) into a tubular neighborhood. It is easy to see that as a result of the normalization of this extension, the property
\[ 0 = \nabla \mathcal{N} \cdot \mathcal{N} = \nabla \mathcal{N} \cdot n \quad \text{on } \Gamma^W \] (14)
holds. Because \( pN \cdot \psi \) and \( \tau N \cdot \psi \) have the same structure, we restrict ourselves to \( pN \cdot \psi \), that is we consider the functional 

\[
j = \int_{\Gamma_w} pN \cdot \psi \, ds,
\]

for which the shape derivative of a general boundary integral as stated in Eq. (11) is applicable. Paired with the properties of the normal extension (14), this leads to

\[
dj(\Omega; V) = \int_{\Gamma_w} p'N \cdot \psi + pN' \cdot \psi + (V \cdot N) \left( \frac{\partial (pN \cdot \psi)}{\partial n} + K(pN \cdot \psi) \right) \, ds
\]

\[
= \int_{\Gamma_w} p'n \cdot \psi + pn' \cdot \psi + (V \cdot n) \left( \frac{\partial (p \cdot \psi)}{\partial n} + n + K(pn \cdot \psi) \right) \, ds.
\]

From Remark 2.4 we have \( N' = d_f [N] - \nabla N \cdot V \) for the local shape derivative of the extended normal vector. Combining Remark 2.2, i.e., assuming \( V = (V \cdot n)n \), with Eq. (14), one arrives at \( n' = d_f [n] \). Also using \( d_f [n] = -\nabla_r (V \cdot n) \) from (12) gives us the relation

\[
\int_{\Gamma_w} pn' \cdot \psi \, ds = -\int_{\Gamma_w} p \nabla_r (V \cdot n) \cdot \psi \, ds.
\]

Application of the tangential Green’s formula to the right hand side of the above equation yields

\[
\int_{\Gamma_w} pn' \cdot \psi \, ds = -\int_{\Gamma_w} (V \cdot n)K(p \psi \cdot n) - (V \cdot n) \, div_r(p \psi) \, ds,
\]

where the assumption \( V = 0 \) in the neighborhood of the farfield boundary \( \Gamma_w \) was used to employ the tangential Green’s formula at the wall boundary only. We will now insert the above equation into (15). As one can see, the terms containing the additive curvature \( K \) cancel out and the following expression remains

\[
dj(\Omega; V) = \int_{\Gamma_w} p' n \cdot \psi + (V \cdot n) \left[ div_r(p \psi) + \frac{\partial (p \cdot \psi)}{\partial n} n \right] \, ds.
\]

The terms within the bracket now exactly align with the definition of the tangential divergence, Eq. (8), such that

\[
div_r(p \psi) + \frac{\partial (p \cdot \psi)}{\partial n} n = \nabla_r(p \psi) + D(p \psi)n \cdot n = \nabla(p \psi).
\]

The same argumentation can now be also applied to \( \tau n \cdot \psi \) instead of \( pn \cdot \psi \). \( \square \)

The above theorem already supplies one possible representation of the shape derivative of the objective functional. However, this preliminary shape derivative is not yet in Hadamard form because it still contains the local shape derivatives \( p' \) and \( \tau' \). Computation of these would require one forward flow solution for each design parameter of the parameterization of the shape, which is prohibitively costly. In Section 4, adjoint calculus is used to remove these local shape derivatives \( p' \) and \( \tau' \), thereby transforming the above gradient expression into the Hadamard form.

### 2.4. Shape derivatives of boundary conditions

Transformation of the gradient expression from Theorem 2.5 requires explicit knowledge of the boundary conditions defining the local shape derivatives \( p' \) and \( \tau' \). These are governed by the respective boundary conditions imposed within the forward problem. As such, we now consider how the Dirichlet condition, the Neumann condition and the slip condition of the forward problem determine these local shape derivatives. The general argumentation again follows [10].

#### 2.4.1. Dirichlet boundaries

First, we consider a general Dirichlet boundary condition \( w = w_D \) on the wall \( \Gamma_w \), where \( w_D \) does not depend on the geometry, meaning \( w_D \) is independent of the parameter \( t \) of the deformation \( T_t \). This is especially true for the no-slip condition \( v = 0 \), which should also be fulfilled on the perturbed boundary. Application of the material derivative paired with Remark 2.4 applied to both sides of the Dirichlet boundary condition \( w = w_D \) yields

\[
dv[w] = w' + \nabla w \cdot V = dv[w_D] = \nabla w_D \cdot V,
\]

where the local shape derivative of \( w_D \) equals zero, since \( w_D \) does not depend on \( t \). From this equation, we can extract a condition defining the local shape derivative \( w' = \nabla (w_D - w) \cdot V \). According to Remark 2.2, it is sufficient to consider only the normal direction \( (V \cdot n)n \) of the vector field \( V \), which leads to

\[
w' = \nabla (w_D - w) \cdot n(V \cdot n) = \frac{\partial w_D - w}{\partial n}(V \cdot n).
\]
2.4.2. Neumann boundaries

Similar to the Dirichlet boundary condition, we again would like to consider the material derivative of the boundary condition of the forward problem and then apply the chain rule argument given by Remark 2.4 to find a corresponding expression for the local shape derivatives. This will again require the quantities under consideration to exist within a tubular neighborhood for which we again assume an extension of the normal $\mathcal{N}$ just as in Theorem 2.5. If we apply the material derivative to both sides of the Neumann boundary condition $\frac{\partial w}{\partial t} = \nabla w \cdot \mathcal{N} = w_N$, where $w_N$ does not depend on the shape, meaning on the parameter $t$, we get with Remark 2.4

$$d_v[\nabla w \cdot \mathcal{N}] = (\nabla w \cdot \mathcal{N})' + \nabla (\nabla w \cdot \mathcal{N}) \cdot V = d_v[w_N] = \nabla w_N \cdot V.$$ 

Since $\nabla w' \cdot \mathcal{N} = (\nabla w \cdot \mathcal{N})' - \nabla w \cdot \mathcal{N}'$ holds, we get from this equation

$$\nabla w' \cdot \mathcal{N} = \nabla w_N \cdot V - \nabla w \cdot \mathcal{N}' - \nabla (\nabla w \cdot \mathcal{N}) \cdot V.$$ 

Using the usual orthogonality argumentation again, Remark 2.2 and again employing Remark 2.4, we can insert the relation $n' = \mathcal{N}' = d_v[\mathcal{N}] - \nabla \mathcal{N} \cdot V = d_v[n]$ and furthermore use $\nabla (\nabla w \cdot \mathcal{N}) \cdot V = D^2 w \cdot \mathcal{N} \cdot V + \nabla w \cdot (\nabla \mathcal{N} \cdot V) = D^2 w n \cdot V$ to obtain

$$\nabla w' \cdot n = \nabla w_N \cdot V - \nabla w \cdot d_v[n] - D^2 w n \cdot V$$ 

$$= (V \cdot n) \left[ \nabla w_N \cdot n - D^2 w n \cdot n \right] - \nabla w \cdot d_v[n].$$ 

From Eq. (12) we get the equality $d_v[n] = -\nabla \mathcal{F}(V \cdot n)$, which results in

$$\frac{\partial w}{\partial n} = (V \cdot n) \left[ \frac{\partial w_N}{\partial n} - \frac{\partial^2 w}{\partial n^2} \right] + \nabla w \cdot \nabla \mathcal{F}(V \cdot n)$$ 

$$= (V \cdot n) \left[ \frac{\partial w_N}{\partial n} - \frac{\partial^2 w}{\partial n^2} \right] + \nabla \mathcal{F} \cdot \nabla \mathcal{F}(V \cdot n),$$ 

where the last transformation directly results from $\nabla w \cdot n = 0$ being inserted into the definition of the tangential gradient. This expression is still not entirely in Hadamard form, but we will later use the tangential Green’s formula to conclude this transformation.

3. Shape derivative of the Navier–Stokes equations in strong and variational form

Before adjoint calculus can be used in Section 4 to finalize the Hadamard form, one first needs to establish the corresponding forward problem. Therefore, we now consider the linearization of the compressible Navier–Stokes equations with respect to a variation of the domain. As mentioned above, we distinguish between the Navier–Stokes equations in pointwise and in variational form. Unsurprisingly, both versions of the forward problem lead to distinct linearizations and this section will be used to discuss the respective differences.

Theorem 3.1 (Shape Derivative of the Pointwise Navier–Stokes Equations). The local shape derivative $u'$ of the Navier–Stokes equations in strong form (2) is given by the solution of

$$0 = \nabla \cdot \left( \mathcal{F}_u^c(u) u' - \mathcal{F}_u^v(u, \nabla u) u' - \mathcal{F}_u^v(u, \nabla u) \nabla u' \right) \quad \text{in } \Omega,$$

where $\mathcal{F}_u^c := \frac{\partial \mathcal{F}_u^c}{\partial u}$, $\mathcal{F}_u^v := \frac{\partial \mathcal{F}_u^v}{\partial u}$ and $\mathcal{F}_u^v := \frac{\partial \mathcal{F}_u^v}{\partial \nabla u}$.

Proof. Applying the local shape derivative to both sides of Eq. (2) results in

$$0 = \nabla \cdot \left( \mathcal{F}_u^c(u) u' - \mathcal{F}_u^v(u, \nabla u) u' - \mathcal{F}_u^v(u, \nabla u) \nabla u' \right). \quad \square$$

Theorem 3.2 (Shape Derivative of the Variational Navier–Stokes Equations). The shape derivative of the variational form of the Navier–Stokes equations (3) is given by the problem: Find $u' \in \mathcal{H}$, such that

$$0 = - \left( u', \left[ \mathcal{F}_u^c(u) - \mathcal{F}_u^v(u, \nabla u) \right]^\top \nabla u \right)_{\Omega} - (V, n) \left[ \mathcal{F}_u^c(u) - \mathcal{F}_u^v(u, \nabla u) \right] \cdot \nabla u \right)_{\Gamma_w}$$

$$- \left( u', V \cdot \left[ \mathcal{F}_u^v(u, \nabla u) \right]^\top \nabla u \right)_{\Omega} + \left( u', n \cdot \left( \mathcal{F}_u^v(u, \nabla u) \right)^\top \nabla u \right)_{\Gamma}$$

$$+ \left( u', \left[ n \cdot \left( \mathcal{F}_u^c(u) - \mathcal{F}_u^v(u, \nabla u) \right)^\top \nabla u \right) \right)_{\Gamma \setminus \Gamma_w}$$

$$- \left( \nabla u', \left( n \cdot \mathcal{F}_u^v(u, \nabla u) \right)^\top \nabla u \right)_{\Gamma \setminus \Gamma_w} + \left( n \cdot \left( \mathcal{F}_u^c(u) - \mathcal{F}_u^v(u, \nabla u) \right)^\top \nabla u \right)_{\Gamma_w}$$

$$+ \int_{\Gamma_w} (V, n) \nabla \cdot \left[ \left( \mathcal{F}_u^c(u) - \mathcal{F}_u^v(u, \nabla u) \right) \cdot \nabla u \right] \, ds \quad \forall \mathbf{v} \in \mathcal{H}. \quad (17)$$
**Proof.** A shape differentiation of the variational form (3) results in

\[
0 = - \left. \frac{d}{dt} \right|_{t=0} \left( F^c(u) - F^v(u, \nabla u) \right) + \left. \frac{d}{dt} \right|_{t=0} \left( n \cdot (F^c(u) - F^v(u, \nabla u)) \cdot v \right)_{\Omega} \\
+ \left( n \cdot (F^c_{u}(u) - F^v_{u}(u, \nabla u)) \cdot v \right)_{\Gamma/\Gamma_W} \\
+ \left( n \cdot \left( (F^c_{u}(u) - F^v_{u}(u, \nabla u))' \cdot v \right)_{\Gamma/\Gamma_W} \right) \\
+ \int_{\Gamma_W} (V, n) \cdot \left( \left( (F^c(u) - F^v(u, \nabla u)) \cdot v \right) \right) ds \quad \forall \mathbf{v} \in \mathcal{H},
\]

where we used Eqs. (10) and (11) for the volume and the farfield integrals and Theorem 2.5 for the wall boundary integral. The terms containing \( v' \) vanish due to the forward equation (3) being satisfied. Using the product rule on \( n \cdot (F^c(u) - F^v(u, \nabla u))' \) and the chain rule on \( (F^c(u)') \) and \( (F^v(u, \nabla u))' \) at the farfield boundary and in the volume leads to

\[
0 = - \left( F^c_{u}(u)u' - F^v_{u}(u, \nabla u)u' - \nabla_{\Theta\Theta}(u, \nabla u) \right)_{\Omega} - \left( (V, n) \left[ F^c(u) - F^v(u, \nabla u) \right] \cdot \nabla v \right)_{\Gamma/\Gamma_W} \\
+ \left( n \cdot \left[ F^c_{u}(u)u' - F^v_{u}(u, \nabla u)u' - \nabla_{\Theta\Theta}(u, \nabla u) \right] \cdot v \right)_{\Gamma/\Gamma_W} \\
+ \int_{\Gamma_W} (V, n) \cdot \left( \left[ F^c(u) - F^v(u, \nabla u) \right] \cdot v \right) ds \quad \forall \mathbf{v} \in \mathcal{H}.
\]

Since \( V = 0 \) is fulfilled in the neighborhood of the farfield boundary \( \Gamma \setminus \Gamma_W \) the local shape derivative of normal vector \( n' \) vanishes and it remains

\[
0 = - \left( F^c_{u}(u)u' - F^v_{u}(u, \nabla u)u' - \nabla_{\Theta\Theta}(u, \nabla u) \right)_{\Omega} - \left( (V, n) \left[ F^c(u) - F^v(u, \nabla u) \right] \cdot \nabla v \right)_{\Gamma/\Gamma_W} \\
+ \int_{\Gamma_W} (V, n) \cdot \left( \left[ F^c(u) - F^v(u, \nabla u) \right] \cdot v \right) ds \quad \forall \mathbf{v} \in \mathcal{H}.
\]

We shift \( n, F^c(u), F^v(u, \nabla u) \) and \( F^v_{u}(u, \nabla u) \) to the other side of the products and integrate the volume term containing \( \nabla u' \) by parts to obtain the stated expression. \( \square \)

### 4. Adjoint calculation

We now recall adjoint calculus to reformulate a shape optimization problem as documented in [1, 27] or [28]. In our case the cost function \( J \) to be shape optimized is the drag or lift coefficient

\[
J = J(u, S),
\]

which depends on a function \( S \) describing the shape and the flow solution \( u \) of the governing equation

\[
N(u, S) = 0. \quad \text{(19)}
\]

Since \( u \) depends, through the governing equation, on the shape function \( S \), a variation of the shape \( \delta S \) leads to the following variation of the cost function

\[
\delta J = \frac{\partial J}{\partial u} \delta u + \frac{\partial J}{\partial S} \delta S. \quad \text{(20)}
\]

Therefore, to compute the variation \( \delta J \), one needs to know the sensitivity of the flow solution \( \delta u \) for each degree of freedom within the shape deformation. To calculate this variation for each such parameter, a flow solution has to be computed. This prohibitive numerical effort of multiple flow computations can be omitted if it is possible to eliminate the variation \( \delta u \). For this purpose, we look at the variation of the governing equation

\[
\delta N = \frac{\partial N}{\partial u} \delta u + \frac{\partial N}{\partial S} \delta S = 0, \quad \text{(21)}
\]
which provides another equation determining the variation $\delta u$. Because the variation $\delta N$ equals zero, it can be multiplied by a Lagrange multiplier $z$ and then be subtracted from the variation of the cost function:

$$\delta f = \delta f - z^T \delta N.$$  

Inserting Eqs. (20) and (21) for the variations $\delta f$ and $\delta N$ into this equation yields

$$\delta f = \frac{\delta f}{\delta u} + \frac{\partial f}{\partial S} - z^T \left( \frac{\partial N}{\partial u} + \frac{\partial N}{\partial S} \delta S \right) = \left( \frac{\delta f}{\delta u} - z^T \frac{\partial N}{\partial u} \right) \delta u + \left( \frac{\partial f}{\partial S} - z^T \frac{\partial N}{\partial S} \right) \delta S.$$ 

The first term, containing $\delta u$, is then eliminated if $z$ satisfies the adjoint equation

$$\frac{\partial f}{\partial u} - z^T \frac{\partial N}{\partial u} = 0. \quad (22)$$

Thereafter the variation of the cost function becomes

$$\delta f = \left( \frac{\partial f}{\partial S} - z^T \frac{\partial N}{\partial S} \right) \delta S, \quad (23)$$

which can be computed without multiple primal flow solutions.

Although the nature of Eq. (19) makes the above motivation more reminiscent of the strong form approach, it is nevertheless presented here to illustrate the methodology. Furthermore, as stated before, we are also considering the strong form for our shape optimization problem so we have a procedure to gauge against. Thus, we are going to consider the shape derivative of the objective function as given by Theorem 2.5, but under the assumption of a state equation in weak form, Eq. (3).

### 4.1. Variational formulation of the continuous adjoint problem

We will use this section to derive the variational formulation of the adjoint problem. For more details on the variational approach also see [24] and the respective integral transforms are covered in more depth in [29].

**Theorem 4.1** (Variational Form of the Adjoint Navier–Stokes Equations). The variational formulation of the adjoint Navier–Stokes equations is given by finding $z \in H$ such that

$$- (w, (F_u^c - F_u^v) \nabla z)_\Omega - (w, \nabla \cdot \left( (F_u^v)^T \nabla z \right)) + (w, n \cdot \left( (F_u^v)^T \nabla z \right))_\Gamma$$

$$+ \left( w, (n \cdot (F_u^c - F_u^v))^T z \right)_\Gamma - \left( \nabla w, (n \cdot (F_u^v)^T \nabla z \right)_\Gamma = f'(u)(w) \quad \forall w \in H,
Corollary 4.2. Choosing $\mathbf{w}$ in Eq. (24) with appropriate compact support in either $\Omega$ or on $\Gamma_W$ and $\Gamma' \setminus \Gamma_W$, one can see that

$$- (\mathbf{w}, (\mathcal{F}_u^\epsilon - \mathcal{F}_u^\gamma) \nabla \mathbf{z})_\Omega - (\mathbf{w}, \nabla \cdot ((\mathcal{F}_v^\gamma \mathbf{u})^\top \nabla \mathbf{z}))_\Omega = 0 \quad \forall \mathbf{w} \in \mathcal{H}_0(\Omega)$$

(26)

for the volume. For a test function with compact support on $\Gamma_W$ we see that

$$\left( (\mathbf{w}, n \cdot ((\mathcal{F}_v^\gamma \mathbf{u})^\top \nabla \mathbf{z}))_\Gamma - (\nabla \mathbf{w}, (n \cdot \mathcal{F}_v^\gamma \mathbf{u})^\top \mathbf{z})_\Gamma \right)_{\Gamma_W} = (\frac{1}{C_\infty} (p_u n - \tau_u n) \cdot \psi, \mathbf{w})_{\Gamma_W} \quad \forall \mathbf{w} \in \mathcal{H} \cap \mathcal{H}_0(\Gamma_W)$$

(27)

and finally using the same argumentation on all remaining boundaries $\Gamma' \setminus \Gamma_W$

$$\left( (\mathbf{w}, n \cdot ((\mathcal{F}_v^\gamma \mathbf{u})^\top \nabla \mathbf{z}))_\Gamma - (\nabla \mathbf{w}, (n \cdot \mathcal{F}_v^\gamma \mathbf{u})^\top \mathbf{z})_\Gamma \right)_{\Gamma' \setminus \Gamma_W} = 0 \quad \forall \mathbf{w} \in \mathcal{H} \cap \mathcal{H}_0(\Gamma' \setminus \Gamma_W).$$

(28)

4.2. Application of adjoint equation to the shape derivative of the Navier–Stokes equations

Recalling our goal of eliminating the local shape derivatives $p'$ and $\tau'$ in Eq. (13), we will now derive two intermediate relationships between the adjoint equation and the respective linearizations of the Navier–Stokes equations. One stems from a consideration of a pointwise linearization while the other stems from the same procedure applied to the linearization of the weak form. The resulting two different intermediate expressions (29) and (30) will then be transformed further in Section 4.3. There, the corresponding variations in terms of the pressure-based variables will be made explicit. The variation of the boundary conditions will be taken into consideration in Section 4.4.

We begin by considering the pointwise problem. Multiplying the shape derivative of the pointwise Navier–Stokes equations from Theorem 3.1 by a test function $\mathbf{v}$ and integrating over the domain $\Omega$ gives us

$$0 = \left( \nabla \cdot \left( \mathcal{F}_u^\epsilon (\mathbf{u}) \mathbf{u}' - \mathcal{F}_u^\gamma (\mathbf{u}, \nabla \mathbf{u}) \mathbf{u}' - \mathcal{F}_v^\gamma (\mathbf{u}, \nabla \mathbf{u}) \nabla \mathbf{u}' \right), \mathbf{v} \right)_\Omega \quad \forall \mathbf{v} \in \mathcal{H}.$$

Integration by parts yields

$$0 = - \left( \mathbf{u}' \left[ \mathcal{F}_u^\epsilon (\mathbf{u}) \mathbf{u}' - \mathcal{F}_u^\gamma (\mathbf{u}, \nabla \mathbf{u}) \mathbf{u}' - \mathcal{F}_v^\gamma (\mathbf{u}, \nabla \mathbf{u}) \nabla \mathbf{u}' \right], \nabla \mathbf{v} \right)_\Omega + \left( \mathbf{v}' \left[ n \cdot \left( \mathcal{F}_u^\epsilon (\mathbf{u}) \mathbf{u}' - \mathcal{F}_u^\gamma (\mathbf{u}, \nabla \mathbf{u}) \mathbf{u}' - \mathcal{F}_v^\gamma (\mathbf{u}, \nabla \mathbf{u}) \nabla \mathbf{u}' \right) \right], \mathbf{u}' \right)_\Gamma \Gamma \setminus \Gamma_W \quad \forall \mathbf{v} \in \mathcal{H}.$$

Shifting $n$, $\mathcal{F}_u^\epsilon (\mathbf{u})$, $\mathcal{F}_u^\gamma (\mathbf{u}, \nabla \mathbf{u})$ and $\mathcal{F}_v^\gamma (\mathbf{u}, \nabla \mathbf{u})$ to the other side of the products results in

$$0 = - \left( \mathbf{u}' \left[ \mathcal{F}_u^\epsilon (\mathbf{u}) \mathbf{u}' - \mathcal{F}_u^\gamma (\mathbf{u}, \nabla \mathbf{u}) \mathbf{u}' - \mathcal{F}_v^\gamma (\mathbf{u}, \nabla \mathbf{u}) \nabla \mathbf{u}' \right], \nabla \mathbf{v} \right)_\Omega + \left( \mathbf{v}' \left[ \n \cdot \left( \mathcal{F}_u^\epsilon (\mathbf{u}) \mathbf{u}' - \mathcal{F}_u^\gamma (\mathbf{u}, \nabla \mathbf{u}) \mathbf{u}' - \mathcal{F}_v^\gamma (\mathbf{u}, \nabla \mathbf{u}) \nabla \mathbf{u}' \right) \right], \mathbf{u}' \right)_\Gamma \Gamma \setminus \Gamma_W \quad \forall \mathbf{v} \in \mathcal{H}.$$

Integration by parts in the 2nd volume integral and applying the chain rule backwards to the wall integral leads to

$$0 = - \left( \mathbf{u}' \left[ \mathcal{F}_u^\epsilon (\mathbf{u}) \mathbf{u}' - \mathcal{F}_u^\gamma (\mathbf{u}, \nabla \mathbf{u}) \mathbf{u}' - \mathcal{F}_v^\gamma (\mathbf{u}, \nabla \mathbf{u}) \nabla \mathbf{u}' \right], \nabla \mathbf{v} \right)_\Omega - \left( \mathbf{u}' \left[ \n \cdot \left( \mathcal{F}_u^\gamma (\mathbf{u}, \nabla \mathbf{u}) \nabla \mathbf{u}' \right) \right], \mathbf{v} \right)_\Gamma \Gamma \setminus \Gamma_W + \left( \mathbf{u}' \left[ n \cdot \left( \mathcal{F}_u^\gamma (\mathbf{u}, \nabla \mathbf{u}) \nabla \mathbf{u}' \right) \right], \mathbf{v} \right)_\Gamma \Gamma \setminus \Gamma_W \quad \forall \mathbf{v} \in \mathcal{H}.$$

Using adjoint conditions (26) and (28) and changing the name of the dependent variable from $\mathbf{v}$ to $\mathbf{z}$, one obtains

$$0 = \left( \mathbf{u}' \left[ \n \cdot \left( \mathcal{F}_v^\gamma (\mathbf{u}, \nabla \mathbf{u}) \right) \nabla \mathbf{z} \right], \mathbf{v} \right)_\Gamma \Gamma \setminus \Gamma_W + \left( n \cdot \left( \mathcal{F}_v^\gamma (\mathbf{u}, \nabla \mathbf{u}) \right) \mathbf{z}, \mathbf{v} \right)_\Gamma \Gamma \setminus \Gamma_W \quad \forall \mathbf{z} \in \mathcal{H}.$$

(29) Using Eqs. (26) and (28), which stem from a weak form adjoint equation, within the strong form linearization here might appear counterintuitive at first glance. However, it should be noted that a pointwise interpretation of those does not effect the above equation.
Contrary to the above preliminary result stemming from the strong form of the Navier–Stokes equations, we next follow the same process, now considering the interaction with the adjoint equation and the variational Navier–Stokes equations from Theorem 3.2, which results in

\[
0 = \left( \mathbf{u}', n \cdot \left[ \left( F^{c \gamma}_v \mathbf{u}, \nabla \mathbf{u} \right)^\top \nabla \mathbf{z} \right] \right)_{\Gamma_W} + \left( n \cdot \left( F^{c \gamma}(\mathbf{u}) - F^{c \gamma}(\mathbf{u}) \right), \mathbf{z} \right)_{\Gamma_W} + \int_{\Gamma_W} \left( V, n \right) \nabla \cdot \left( \left[ F^{c}(\mathbf{u}) - F^{c}(\mathbf{u}) \right] \cdot \mathbf{z} \right) ds - \left( \left( V, n \right) \left[ F^{c}(\mathbf{u}) - F^{c}(\mathbf{u}) \right], \nabla \mathbf{z} \right)_{\Gamma_W} \quad \forall \mathbf{z} \in \mathcal{H}.
\]

We apply the product rule to the divergence and get

\[
0 = \left( \mathbf{u}', n \cdot \left[ \left( F^{c \gamma}_v \mathbf{u}, \nabla \mathbf{u} \right)^\top \nabla \mathbf{z} \right] \right)_{\Gamma_W} + \left( n \cdot \left( F^{c \gamma}(\mathbf{u}) - F^{c \gamma}(\mathbf{u}) \right), \mathbf{z} \right)_{\Gamma_W} + \left( (V, n) \nabla \cdot \left[ F^{c}(\mathbf{u}) - F^{c}(\mathbf{u}) \right], \mathbf{z} \right)_{\Gamma_W} \quad \forall \mathbf{z} \in \mathcal{H}.
\]

Comparing Eqs. (29) and (30) we see that for the variational form of the Navier–Stokes equations, there is one extra term (framed), which vanishes if the Navier–Stokes equations are fulfilled pointwise. In the following, we can therefore avoid a fork and always use Eq. (30), keeping in mind that all framed terms only occur in the variational approach. Also, expressing the variation \( \mathbf{u}' \) in terms of the non-conserved variables works likewise, irrespective of the underlying form of the state equation.

4.3. Transformation to non-conserved variables

In this subsection we insert in particular the no-slip condition into the shape derivative of the Navier–Stokes equations (30). Therefore we first state the so called homogeneity tensor \( G = F^{c \gamma}_v \mathbf{u} \) at the no-slip wall. We also use \( v = 0 \) to get the representations of \( \mathbf{u}' \), \( (F^{c \gamma})' \) and \( (F^{c \gamma})' \) at the wall boundary. With these terms the local shape derivative \( \mathbf{u}' \) is then reformulated, such that we form the respective local shape derivative of the pressure \( \rho' \) and the viscous stress tensor \( \tau' \), which will later eliminate their counterparts in the preliminary shape derivative of the cost function, given by Eq. (13).

As in [29], the homogeneity tensor is given by

\[
G = \left[ G^{kl}_{ij} \right]_{\mathbb{R}^{3 \times 3}} \quad \text{where} \quad (f^{c \gamma})_i \quad \text{denotes the ith component of the kth viscous flux vector and} \quad \frac{\partial u_j}{\partial x_i} \quad \text{denotes the derivative of the jth component of the vector of conserved variables with respect to} \quad x_i. \quad \text{Therefore in two dimensions the indices fulfill} \quad i, j \in \{1, \ldots, 4\} \quad \text{and} \quad k, l \in \{1, 2\}. \quad \text{The homogeneity tensor} \quad G \quad \text{at the no-slip wall is given by}
\]

\[
G_{11} = \frac{\mu}{\rho} \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & \frac{4}{3} & 0 & 0 \\
0 & 0 & 1 & 0 \\
-\nu E & 0 & 0 & \frac{\nu E}{\rho}
\end{pmatrix}, \quad G_{12} = \frac{\mu}{\rho} \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & -\frac{2}{3} & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix},
\]

\[
G_{21} = \frac{\mu}{\rho} \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & -\frac{1}{3} & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}, \quad G_{22} = \frac{\mu}{\rho} \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & \frac{4}{3} & 0 \\
-\nu E & 0 & 0 & \frac{\nu E}{\rho}
\end{pmatrix}.
\]

See [29] for a more detailed discourse on the general homogeneity tensor. The shape derivative of the vector of conserved variables \( \mathbf{u}' \) reduces with \( (\rho v_i)' = \rho v_i' + \rho' v_i = \rho v_i' \) at the no-slip wall to

\[
\mathbf{u}' = \begin{pmatrix}
\rho v_1' \\
\rho v_2' \\
\rho v_3' \\
\rho v_4'
\end{pmatrix} \quad \text{on} \quad \Gamma_W.
\]

Because \( (\rho v_i v_j)' = (\rho v_i) v_j + \rho v_i v_j' = 0 \) holds due the no-slip condition, the shape derivative of the convective flux \( (F^{c \gamma})' \) reduces to

\[
(F^{c \gamma})' = \begin{pmatrix}
\rho v_1' + p & \rho v_2' \\
\rho v_1 v_2' + p & \rho v_3' + p \\
\rho H v_1' & \rho H v_2'
\end{pmatrix} \quad \text{on} \quad \Gamma_W.
\]

(31)
For the shape derivative of the viscous flux \((F^v)'\) we can use \((\tau_g v_j)' = \tau_g v_j'\) at the no-slip boundary to obtain

\[
(F^v)' = \begin{pmatrix}
0 & \tau_{i1}' & 0 \\
\tau_{1i}' & \tau_{12}' + \kappa \frac{\partial T'}{\partial x_1} & \tau_{i2}' \\
0 & \tau_{21}' & \tau_{22}' + \kappa \frac{\partial T'}{\partial x_2}
\end{pmatrix}
on \Gamma_W.
\]  

(32)

These terms are now inserted into the shape derivative of the Navier–Stokes equations for each scalar product of Eq. (30) separately. First we consider the second scalar product of Eq. (30) and insert the representations (31) and (32) of \((F^v)'\) and \((F^v)'\) into this integral:

\[
\left( n \cdot \left( (F^v)' - (F^v)' \right), z \right)_W = \int_{\Gamma_W} \rho \left( p'n - \tau'n \right) \cdot z_{2,2} + v' \cdot \left( \rho n z_1 + (\rho Hn - \tau n) z_4 \right) - \nabla T' \cdot n \kappa z_4 \, ds.
\]

(33)

Next we use the homogeneity tensor to rewrite the first scalar product of Eq. (30):

\[
\left( u', n \cdot \left( (F^v_{vu})^\top \nabla z \right) \right)_W = \int_{\Gamma_W} \sum_{j=1,\ldots,4} \sum_{i=1,\ldots,4} u'_j n_i G_{ij} \frac{\partial z_l}{\partial x_k} \, ds.
\]

As we can see, the first component of \(z\) is multiplied with the first lines of the matrices \(G_{i1}\), which are all zero. Therefore there is no contribution for \(z_1\). The non-zero entries in the second and third lines of \(G\) at the no-slip wall look familiar to the coefficients of \(\frac{\partial v_j}{\partial x_k}\) in the viscous stress tensor. If we define the adjoint stress tensor as

\[
\Sigma := \mu \left( \nabla z_{2,3} + (\nabla z_{2,3})^\top - \frac{2}{3} \left( \nabla \cdot z_{2,3} \right) I \right)
\]

one can actually see that the second and third lines yield

\[
\int_{\Gamma_W} \sum_{j=1,\ldots,4} \sum_{k=1,\ldots,3} u'_j n_i G_{ij} \frac{\partial z_l}{\partial x_k} \, ds = \int_{\Gamma_W} u' \cdot (n \cdot \Sigma) \, ds.
\]

For the fourth component of \(z\), meaning the fourth lines of \(G\), we only get contributions for \(k = l = j = 1, 2, 3, 4\). Since the \(j\)th columns of the matrices \(G_{i1}\) are multiplied by the \(j\)th component of \(u\), the remaining expression is

\[
\int_{\Gamma_W} \sum_{j=1,\ldots,4} \sum_{k=1,\ldots,4} u'_j n_i G_{ij} \frac{\partial z_l}{\partial x_k} \, ds = \int_{\Gamma_W} \frac{\mu}{\rho} \frac{v}{Pr} \left( -E u'_1 + u'_4 \right) n \cdot \nabla z_4 \, ds.
\]

With \(u'_j = \rho' \) and \(u'_4 = (\rho E)' = \rho'E + \rho E'\) this integral is simplified to

\[
\int_{\Gamma_W} \frac{\mu}{\rho} \frac{v}{Pr} \rho' E n \cdot \nabla z_4 \, ds = \int_{\Gamma_W} \nabla \kappa n \cdot \nabla z_4 \, ds
\]

where we use \(\nabla \kappa = \frac{\mu v}{Pr} \left( E - \frac{1}{2} v^2 \right)\) and the no-slip condition to substitute

\[
\nabla \kappa = \frac{\mu v}{Pr} \left( E' - \frac{1}{2} v^2 \right) = \frac{\mu v}{Pr} E'.
\]

In total we obtain for the first scalar product of Eq. (30)

\[
\left( u', n \cdot \left( (F^v_{vu})^\top \nabla z \right) \right)_W = \int_{\Gamma_W} u' \cdot (n \cdot \Sigma) \, ds + \int_{\Gamma_W} \nabla \kappa n \cdot \nabla z_4 \, ds.
\]

(34)
Combining the results of (33) and (34) the preliminary shape derivative equation (30), now containing the variation of the boundary condition and the explicit variations of the primal variables, becomes

\[ 0 = \int_{\Gamma_W} v' \cdot (n \cdot \Sigma) \, ds + \int_{\Gamma_W} T' \kappa n \cdot \nabla z_1 \, ds \]

\[ + \int_{\Gamma_W} (p' n - \tau' n) \cdot \mathbf{z}_{2,3} + v' \cdot (\rho n z_1 + (\rho H n - \tau n) z_4) - \nabla T' \cdot n \kappa z_4 \, ds + \left[ \left((V \cdot n)(\nabla \cdot (F^c - F^v)), z\right)_{\Gamma_W} \right]. \]

4.4. Subtraction of the shape derivative of the Navier–Stokes equations from the preliminary shape derivative of the cost function

The above equality is now, in accordance with the adjoint approach, subtracted from the preliminary shape derivative of the cost function (13), thereby obtaining a representation that does not contain any local shape derivatives.

\[ df(\Omega; V) = \frac{1}{C_\infty} \int_{\Gamma_W} (p' n - \tau' n) \cdot \psi \, ds + (V \cdot n) \cdot (p \psi - \tau \psi) \, ds \]

\[ - \int_{\Gamma_W} v' \cdot (n \cdot \Sigma) \, ds - \int_{\Gamma_W} T' \kappa n \cdot \nabla z_1 \, ds \]

\[ - \int_{\Gamma_W} (p' n - \tau' n) \cdot \mathbf{z}_{2,3} \, ds + v' \cdot (\rho n z_1 + (\rho H n - \tau n) z_4) - \nabla T' \cdot n \kappa z_4 \, ds \]

\[ - \left[ \left((V \cdot n)(\nabla \cdot (F^c - F^v)), z\right)_{\Gamma_W} \right]. \]  

(35)

To ease the following discussions we use framed numbers \([1, 2, 3, \ldots]\) to refer to certain terms. Term 3 and term 1 cancel each other if the adjoint boundary condition \(z_{2,3} = C_\infty \psi \) on \( \Gamma_W \) is fulfilled. Furthermore, this boundary condition also implies that the \(z_{2,3}\)-component of expression 4 cancels out term 2 because due to \( \frac{\partial \rho v_i}{\partial x_k} = \frac{\partial \rho v_i}{\partial x_k} + \rho v_i \frac{\partial v_j}{\partial x_k} = 0 \) at the no-slip boundary, the equation

\[ (\nabla \cdot (F^c - F^v))_{2,3} = \nabla \cdot \left( (\rho v_i v_j + pl - \tau) \right) = \text{div}(pl - \tau) \]  

(36)

and therewith \((\nabla \cdot (F^c - F^v))_{2,3} \cdot z_{2,3} = \frac{1}{C_\infty} \text{div}(p \psi - \tau \psi)\) holds.

When considering the pointwise equations the scalar product 4 is not present and one would expect 2 to remain untouched. In this situation, however, Eq. (36), being the pointwise conservation of momentum, equals zero and therefore term 2 vanishes anyway. Summarizing the above, the first and last components of 4 are the only differences between the shape derivatives of the two approaches. We mark them by 4,1,4

\[ df(\Omega; V) = - \int_{\Gamma_W} v' \cdot (n \cdot \Sigma) \, ds - \int_{\Gamma_W} T' \kappa n \cdot \nabla z_1 \, ds - \int_{\Gamma_W} v' \cdot (\rho n z_1 + (\rho H n - \tau n) z_4) - \nabla T' \cdot n \kappa z_4 \, ds \]

\[ - \left[ \left((V \cdot n)(\nabla \cdot (F^c - F^v)), z_{1,4}\right)_{\Gamma_W} \right]. \]  

(37)

To achieve independence of the remaining local shape derivatives \(v'\) and \(T'\), we use the variation of the boundary conditions, as described in Section 2.4, for the no-slip, adiabatic and isothermal boundary condition:

\[ v' = -(V \cdot n) \frac{\partial v}{\partial n} \quad \text{on } \Gamma_W, \]

\[ T' = (V \cdot n) \frac{\partial T_{\text{iso}} - T}{\partial n} \quad \text{on } \Gamma_{\text{iso}}, \]

\[ \nabla T' \cdot n = -(V \cdot n) \frac{\partial^2 T}{\partial n^2} + \nabla T \cdot \nabla (V \cdot n) \quad \text{on } \Gamma_{\text{adi}}. \]

Inserting these conditions on the respective parts of the wall, Eq. (37) becomes

\[ df(\Omega; V) = \int_{\Gamma_W} (V \cdot n) \frac{\partial v}{\partial n} \cdot (n \cdot \Sigma) \, ds - \int_{\Gamma_{\text{Iso}}} (V \cdot n) \frac{\partial T_{\text{Iso}} - T}{\partial n} \kappa n \cdot \nabla z_1 \, ds - \int_{\Gamma_{\text{adi}}} T' \kappa n \cdot \nabla z_4 \, ds \]

\[ + \int_{\Gamma_W} (V \cdot n) \frac{\partial v}{\partial n} \cdot (\rho n z_1 + (\rho H n - \tau n) z_4) \, ds + \int_{\Gamma_{\text{Iso}}} \nabla T' \cdot n \kappa z_4 \, ds + \int_{\Gamma_{\text{adi}}} (V \cdot n) \frac{\partial^2 T}{\partial n^2} z_4 \, ds \]

\[ + \int_{\Gamma_{\text{adi}}} \nabla T \cdot \nabla (V \cdot n) \kappa z_4 \, ds - \left[ \left((V \cdot n)(\nabla \cdot (F^c - F^v)), z_{1,4}\right)_{\Gamma_W} \right]. \]  

(38)
Application of the tangential Green’s formula (9) to integral [7] leads to
\[ \int_{\Gamma_{\text{adia}}} \nabla T \cdot \nabla r (V \cdot n) \kappa z_4 \, ds = \int_{\Gamma_{\text{adia}}} (V \cdot n) K (\nabla T \cdot n) \kappa z_4 - (V \cdot n) \text{div}_r (\nabla T \kappa z_4) \, ds \]
\[ = - \int_{\Gamma_{\text{adia}}} (V \cdot n) \text{div}_r (\nabla T \kappa z_4) \, ds, \]
where the adiabatic wall condition \( \nabla T \cdot n = 0 \) was used in the second line.

Integrals of Eq. (37) containing the remaining local shape derivatives \( T’ \) at the adiabatic wall [5] and \( \nabla T’ \) at the isothermal wall [6] vanish if the following adjoint boundary conditions are fulfilled
\[ z_4 = 0, \quad \text{on } \Gamma_{\text{iso}} \quad \text{and} \quad V z_4, \quad n = 0, \quad \text{on } \Gamma_{\text{adia}}. \]

The last step is to give an expression for the term [41,4] We therefore consider this expression separately for the first and fourth components, denoted by \( \nabla \cdot (\mathcal{F}^c - \mathcal{F}^v)_{1,4} \). Because the first line of the viscous flux \( \mathcal{F}^v \) equals zero anyway and furthermore the no-slip condition holds, the first component simplifies to
\[ \nabla \cdot (\mathcal{F}^c - \mathcal{F}^v)_{1,4} = \nabla \cdot (\rho v_1, \rho v_2) = \rho (\nabla \cdot v). \]

For the fourth component we get, again with the no-slip condition,
\[ \nabla \cdot (\mathcal{F}^c - \mathcal{F}^v)_{4} = \nabla \cdot (\rho H v_1, \rho H v_2) - \nabla \cdot \left( \sum_j \tau_{ij} v_j + \kappa \frac{\partial T}{\partial x_1}, \sum_j \tau_{ij} v_j + \kappa \frac{\partial T}{\partial x_2} \right) \]
\[ = \rho H (\nabla \cdot v) - \sum_{i,j} \tau_{ij} \frac{\partial v_j}{\partial x_i} - \kappa \Delta T. \]

Altogether the shape derivative of the drag or lift coefficient in Hadamard form becomes
\[ df(\Omega; V) = \int_{\Gamma_{\text{iso}}} (V \cdot n) \frac{\partial v}{\partial n} \cdot (n \cdot \Sigma) \, ds - \int_{\Gamma_{\text{adia}}} (V \cdot n) \frac{\partial T_w - T}{\partial n} \kappa n \cdot \nabla z_4 \, ds \]
\[ + \int_{\Gamma_{\text{iso}}} (V \cdot n) \frac{\partial v}{\partial n} \cdot (\rho n z_1 + (\rho H n - \tau n) z_4) \, ds \]
\[ - \int_{\Gamma_{\text{adia}}} (V \cdot n) \left( \frac{\partial^2 T}{\partial n^2} \kappa z_4 + \nabla T (\nabla T \kappa z_4) \right) \, ds - \int_{\Gamma_{\text{iso}}} (V \cdot n) \rho (\nabla \cdot v) z_1 \, ds \]
\[ - \int_{\Gamma_{\text{adia}}} (V \cdot n) \left( \rho H (\nabla \cdot v) - \sum_{i,j} \tau_{ij} \frac{\partial v_j}{\partial x_i} - \kappa \Delta T \right) z_4 \, ds. \]  
\[ (39) \]

where the difference between the variational and the pointwise approach is based on the two framed integrals in the last line, which only appear upon consideration of the variational form of the Navier–Stokes equations.

5. Numerical results

We now present numerical results to show the difference between the pointwise and the variational approach in application. The shape derivative of both drag and lift coefficient is implemented in the discontinuous Galerkin solver PADGE developed primarily at the German Aerospace Center (DLR), Braunschweig. This flow solver is already capable to also compute the necessary adjoint solutions, originally implemented for error estimation and \( p \)-\( h \)-refinement. This adjoint solution, together with the primal solution, is directly used to calculate the Hadamard form (39) of the drag and lift coefficient. This formula only consists of boundary integrals on the airfoil and contains multiple terms depending on geometrical quantities like the normal vector. Hence, an accurate approximation of the shape of the airfoil is necessary to gain maximum accuracy of our high order method, especially with regard to those geometric quantities:

The initial geometry is given by a fine structured hexahedral mesh consisting of straight elements. Before the initial flow calculation, the fine resolution hexahedral elements are merged into spatially coarser elements, but with curved boundaries. Thus, information about the geometry is preserved by transforming many straight sided elements to fewer curved ones. Internally, these curved elements are defined by a 4th order polynomial, mapping the reference element to the physical element. Thus, the actual geometrical information stemming from the fine straight sided mesh is preserved and transformed into spatially coarse elements and a 4th order polynomial mapping link them to the reference element. Consequently, the boundary curve is given by a piecewise per element polynomial of 4th order. In order to calculate the integrals of the weak form, a Gaussian quadrature rule of 8th order is used, resulting in considerably more quadrature points per element than degrees of freedom for the Test- and Ansatz functions of the DG discretization.

Our finite difference validations are then conducted by modifying the mapping of the reference element to the physical element, meaning the spatial position of the vertices of the coarse mesh elements are never actually moved, but rather
 included in the curvature information. To conduct our finite difference studies, we thus add a polynomial $f$ to the reference-to-physical element map satisfying

\begin{align*}
  f(0) &= 0, & f'(0) &= 0, \\
  f(1) &= 0, & f'(1) &= 0, \\
  f(0.5) &= \epsilon,
\end{align*}

where $\epsilon$ is the finite difference step-length and the line segment in physical dimensions is normalized to the interval $[0, 1]$ with midpoint 0.5.

For evaluating the Hadamard-Form, we have 5 quadrature points on the boundary stemming from our 8th order Gaussian quadrature as mentioned above. As a first step, we evaluate the shape Gradient without the $(V, n)$ expression on these 5 quadrature points using the polynomial representation of Ansatz functions of the state and adjoint solution. In order to achieve comparability with the finite difference perturbation and the Hadamard approach, we transform the above polynomial $f$ operating on the element mappings into an effective physical boundary movement, represented by a corresponding vector field $V$, which is then used as the $(V, n)$ term and multiplied with the preliminary Hadamard form. The resulting expression is then integrated over the respective element, creating a discretized gradient vector in $\mathbb{R}^n$ where each component corresponds to one element and consequently one $\epsilon$ in the definition of $f$.

Part of the PADGE environment is the ADIGMA MTC3 test case, which is defined as the calculation of flow around a NACA0012 airfoil at Mach $M = 0.5$, angle of attack $\alpha = 2.0^\circ$ and a Reynolds number of $Re = 5000$. Because this test case is thoroughly verified, it is a good basis gauge our shape derivatives. Our grid consists of 1640 cells and the profile as the boundary of the mesh is represented by 40 curved edges of polynomial order four. To obtain a very accurate solution and study the error and behavior for varying polynomial degree $p$ of the Galerkin ansatz, we use a $p$-refinement from degree three to five. The polynomial degree of the adjoint solution was always chosen to be one degree higher, i.e. four to six. We verify our two respective shape derivative implementations against finite differences, considering lift and drag separately. The finite difference reference solution is created as follows. Each edge is disturbed by a fourth order polynomial individually as described above, such that the profile stays smooth. A flow solution is then computed for each such perturbation. Because the shape gradient can be evaluated in every boundary quadrature point of these degree-4-polynomials representing the edges of the profile, we initially calculate five times as many gradient components as edges. Using this information from all boundary quadrature nodes, we integrate over each edge individually to find the respective value to be used in our figures and comparisons, which corresponds to the center “bump” created by our finite difference approach given by $f$ above. Since the step size of the perturbation for finite differences is unknown in general, we compute the absolute difference of the shape derivative in variational Hadamard form and the finite differences for step sizes ranging from $10^{-12}$ to $10^{-3}$, that is we interpret the finite difference error as a function of the perturbation parameter $\epsilon$ at four given spacial positions on the airfoil. In Fig. 1 these errors are shown and despite a quite substantial difference at the nose of the airfoil, all curves show a step size between about $10^{-3}$ and $10^{-5}$ to be acceptable. Another possible way to find the optimal step size for the finite differences is to consider the finite difference error as a function of spacial position with respect to fixed perturbation size $\epsilon$, which is studied in Fig. 2 and leads to similar conclusions as those above. The spacial finite difference error for all step sizes $10^{-5}$, $10^{-7}$ and $10^{-9}$ coincides perfectly, both for drag and lift. Consequently, we chose a step size of $\epsilon = 10^{-7}$ to be used as a reference for validating the Hadamard representation, because selecting $\epsilon = 10^{-7}$ is between the upper and lower bound of the suitable range and is very robust with respect to cancellation errors.

In Fig. 3 finite differences of the drag and lift coefficient are plotted against the shape derivatives in Hadamard form for the variational and the pointwise approach. Both the finite differences and the two Hadamard forms were calculated for a
solution made of third order polynomials. One can observe that the shape derivative of the variational approach matches very nicely with the finite differences, whereas the shape derivative of the pointwise approach deviates noticeably.

As we increase the polynomial degree of the flow solution, see Fig. 4, both Hadamard forms fit the finite differences better overall. However, around the nose of the profile and the forward pressure stagnation point, the offset of the Hadamard form of the pointwise approach is still unmissable, most likely due to the magnitude of the gradients of the flow solution. Remarkably, the shape derivative stemming from the variational approach aligns nearly perfectly with the finite difference reference.

Further $p$-refinement and therewith enhancement of the accuracy of the solution follows this trend: In Fig. 5 we see an excellent match between the variational approach and finite differences, whereas the pointwise approach still exhibits a fairly substantial gap in the region around the nose, although somewhat diminished when compared to the lower degree case.

The natural next step after deriving the gradients is conducting the actual design optimization. A very popular choice for an optimization strategy is the so called one-shot methodology, where the design update is made simultaneously to the iteration of the primal and adjoint solver [6,15,7]. Consequently, gradients created from an inexact and not fully converged flow and adjoint solution are used initially, before all three residuals are driven to convergence simultaneously. Thus, the precision of the shape derivative for inaccurate flow solutions is of crucial importance.
Despite the shortcomings in accuracy of the polynomial degree three solution, as shown in Fig. 3, it is well worth pointing out that the quality of the variational Hadamard form would most likely suffice to be used in a one-shot optimization without compunction. This can be justified, because the general manner of the shape derivative, especially in terms of sign, is captured. Contrary to this, the shape derivative of the pointwise approach shows a tremendous deviation over the whole profile, which questions its utility for optimization, which we intend to study as future work.

6. Conclusion

Shape optimization under PDE constraints very often follows the “function composition” approach, where the existence of local shape derivatives for each component in the whole chain containing mesh perturbation, PDE variation and the variation of the objective is assumed to exist. For elliptic problems, this approach is known to work very well [10], somewhat contrary to problems within aerodynamic design, where usually great care needs to be taken [16]. The main purpose of the present work was to circumvent this “step-by-step” differentiation and directly shape differentiate the weak form of the governing equations, similar to [24]. Here, however, we focus on the challenging task of shape optimization within compressible, viscous fluids, for which we also demonstrate the respective gain in accuracy numerically. Any approach based on the weak form is also much more inline with the actual flow solver. Thus, a variational methodology greatly benefits the alignment of the continuous shape differentiation process with the discrete implementation.
One aspect of the present work was the direct comparison between both approaches, which revealed extra terms arising if the variational form of the state equation is used as the governing model. Both shape derivatives were implemented into the DLR flow solver PADGE, a discontinuous Galerkin flow solver of variable order operating on fourth order curvilinear meshes. We conclude with numerical accuracy studies where we gauge both derivatives against a reference solution created by finite differences. Although the gap between the shape derivative based on the weak and strong form diminishes slightly with increasing order of the polynomial ansatz functions, we always found the weak form derivative to be of considerably higher accuracy. Finally, we found the general quality of this weak form shape derivative to be very promising for a future application within a one-shot optimization framework.

Acknowledgments

We wish to thank the BMBF for supporting this work as part of the project DGHP OPT BMBF 05M10PAB. We also wish to thank Dr. R. Hartmann, DLR Braunschweig, for his assistance with the flow solver PADGE.

References


