No. 573 **June 2017**

Preconditioning for hyperelasticity-based mesh optimization

J. Paul

ISSN: 2190-1767

Preconditioning for hyperelasticity-based mesh optimisation

Jordi Paul*

Institute of Applied Mathematics (LS III),
TU Dortmund University, Vogelpothsweg 87, D-44227 Dortmund, Germany

June 30, 2017

Summary

A robust mesh optimisation method is presented that directly enforces the resulting deformation to be orientation preserving. Motivated by aspects from mathematical elasticity, the energy functional of the mesh deformation can be related to a stored energy functional of a hyperelastic material. Formulating the functional in the principal invariants of the deformation gradient allows fine grained control over the resulting deformation. Solution techniques for the arising nonconvex and highly nonlinear system are presented. As existing preconditioners are not sufficient, a PDE-based preconditioner is developed.

Keywords: Mesh optimisation, hyperelasticity, nonconvex optimisation, preconditioning, finite elements

1 Introduction

For the numerical solution of partial differential equations, the finite element method is a widely used discretisation technique which requires a mesh on the (computational) domain [11, 15, 47]. Very often, the domain of interest and thus the mesh is not stationary, but moves due to various reasons. This might be a free capillary surface (see [2, 5, 23] for examples) for flow problems, fluid-structure interactions (e.g. [8, 45, 51]) or even phase transformations (e.g. [6, 7, 21]).

A moving mesh introduces the problem of geometric stability, which means that intersections of cells (which are fatal for the finite element solution) cannot be ruled out *a priori*. This is sometimes called *mesh tangling* in the literature, and has to be avoided by employing a suitable mesh optimisation technique. One of the most common scenarios is that the movement of one or more boundaries is known and this movement has to be extended to the interior of the domain (to adjust the vertices) by constructing an extension (or inverse trace) operator. This is called *mesh smoothing* by some authors.

Mesh optimisation is a large field of research (see [34] and the references therein), ranging from combinatorial methods explicitly formulated in vertex movements to general purpose methods using different elliptic or parabolic PDEs, or algebraic notions of mesh quality. One recurring aspect is for methods to ensure that the mesh deformation is *orientation preserving* by different means (e.g. note the definition of the element condition number in [22]) as this rules out cell intersections. However, most methods do not try to enforce this property directly and thus cannot ensure that it holds in general.

The focus of this work is variational mesh optimisation by minimising a functional that expresses the energy required to deform well-shaped cells into their configuration in the current mesh. In [46], a mechanical model is derived that relates the deformation energy for simplex meshes to a polyconvex stored energy function of a hyperelastic material. As this directly enforces the orientation preserving property, it can guarantee geometric stability and has recently been generalised to quadrilateral and hexahedral meshes [41] using theory from mathematical elasticity [3, 14]. In

*Email: jordi.paul@math.tu-dortmund.de

this work, the method will be derived in a more general manner and solution techniques from large scale nonconvex optimisation will be presented for the minimisation of the resulting highly nonlinear functional. Efficient PDE-based multilevel preconditioning techniques will be presented and an application to solving the incompressible Navier-Stokes equations on a complex moving domain will be shown.

2 Variational optimisation of moving meshes

Let $\Omega \subset \mathbb{R}^d, d = 2, 3$ be the region of interest, Ω_h its polygonally bounded approximation and $\mathcal{T}_h$ be a regular, conforming discretisation of Ω_h into intervals ($d = 1$), triangles or quadrilaterals ($d = 2$) or (in the case $d = 3$) tedrahedra or hexahedra. The general task will be to approximately solve some PDE on Ω by discretising it with the Finite Element Method using the mesh $\mathcal{T}_h$.

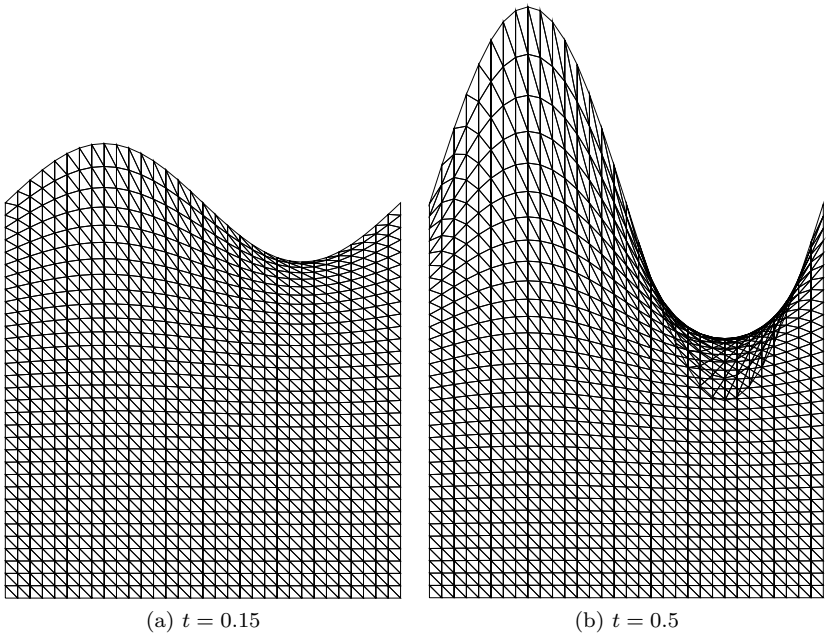
Assume now that the domain is moving: $\Omega = \Omega(t) \forall t \in [0, \bar{t}]$. Let $\Omega_0 =: \Omega(0)$ be given, but at each instant t only the position of the boundary $\partial\Omega(t)$ is known. That means we are looking for $\varphi : [0, \bar{t}] \times \Omega_0 \to \mathbb{R}^d, \Omega(t) := \varphi(t, \Omega_0)$, but only $\operatorname{tr} \varphi : [0, \bar{t}] \times \partial\Omega_0 \to \mathbb{R}^d$ is known. So we have to find an inverse trace operator that extends the boundary movement into the interior, which can be formulated as

$$\text{For } t \in [0, \bar{t}] \text{ find } \Phi : \Omega_0 \to \mathbb{R}^d : \forall x \in \partial\Omega_0 : \Phi(x) = \operatorname{tr} \varphi(t, x).$$

In general, $\operatorname{tr} \varphi$ is unknown and part of the solution to the PDE (like the position of the free capillary boundary (see e.g. [4]), the evolution of the phase boundary (e.g. [7]) etc.). On the discrete level, this can be formulated as

$$\text{For } t \in [0, \bar{t}] \text{ find } \Phi_h : \Omega_h(0) \to \mathbb{R}^d : \forall x \in \partial\Omega_h(0) : \Phi_h(x) = \operatorname{tr} \varphi(t, x).$$

The important part is now to ensure that $\forall t \in [0, \bar{t}]$, the moving mesh $\mathcal{T}_h(t) = \Phi_h(\mathcal{T}_h)$ has the necessary properties for use in a FE discretisation of the original PDE. In particular, cell intersections (also called *mesh tangling* [34, Chapter 1.6]) must be ruled out as a requirement for the construction of parametric FE spaces on $\mathcal{T}_h(t)$. However, this is impossible to do *a priori* if the boundary movement is unknown (see Figure 1).



(a) $t = 0.15$ (b) $t = 0.5$

Figure 1: Moving mesh with eventually intersecting cells.

For this, the mapping Φ will be required to come from a *space of admissible deformations*, or rather a further restricted subspace to account for boundary conditions. It is then easy to show that the mesh resulting from such a deformation is free of intersecting cells.

Because the formulation will be done for the continuous and the discrete case of deformations at the same time, a few definitions are needed so that meshes can be expressed by means of mappings of reference cells.

2.1 Geometric entities and meshes

In the following, *s-dimensional simplex* means intervals, triangles or tetrahedra for $s = 1, 2, 3$ and *s-dimensional hypercube* means intervals, quadrilaterals or hexahedra for $s = 1, 2, 3$ for the sake of brevity, where s is the *shape dimension*.

Definition 1 (Reference cells)**.**

1. *The s-dimensional standard simplex is* $\hat{S}_s := \{\hat{x} \in \mathbb{R}^d : \forall i = 1, \dots, s : \hat{x}^i \geq 0, \sum_{i=1}^{s} \hat{x}^s \leq 1\}$.
2. *The s-dimensional hypercube is* $\hat{Q}_s := [-1, 1]^s \subset \mathbb{R}^s$.

In the following, a reference cell of either shape type shall be denoted by $\hat{K}_s$ *or simply* $\hat{K}$*. All reference cells are always assumed to be positively oriented, meaning* $\mathrm{vol}(\hat{K}) > 0$*.* $\mathcal{E}^0(\hat{K})$ *denotes the vertices of the reference cell* $\hat{K}$*. For* $s \geq 2$*,* $\mathcal{E}^1(\hat{K})$ *denotes the (relatively open) edges of* $\hat{K}$ *and for* $s = 3$*,* $\mathcal{E}^2(\hat{K})$ *its (relatively open) faces.*

Definition 2 (First order reference mappings)**.**

1. *A set of points* $a_0, \dots, a_s \subset \mathbb{R}^d$ *is said to form a non-degenerate s-simplex* K *iff the* $\mathbb{P}_1$ *reference mapping* $R_K : \hat{S}_s \to \mathbb{R}^d, \quad R(\hat{x}) = a_0 + \sum_{i=1}^{s} \hat{x}^i a_i$ *has the property* $\forall x \in \hat{S}_s : \det(\nabla R_K)(\hat{x}) \neq 0$.
2. *If* $\hat{\phi}_i, i = 1, \dots, 2^s$ *are the standard s-linear Lagrangian basis functions on* $\hat{Q}_s$ *with regard to its vertices, the vertices* $a_1, \dots, a_{2^s} \subset \mathbb{R}^d$ *form a non-degenerate hypercube* K *iff the* $\mathbb{Q}_1$ *reference mapping* $R_K : \hat{Q}_s \to \mathbb{R}^d, \quad R(\hat{x}) = a_0 + \sum_{i=1}^{2^s} \hat{\phi}(\hat{x}^i) a_i$ *has the property* $\forall x \in \hat{Q}_s : \det(\nabla R_K)(\hat{x}) \neq 0$.

Without loss of generality, it can be assumed that in both cases $\forall x \in \hat{K}_s : \det(\nabla R_K)(\hat{x}) > 0$ *(otherwise, the local vertex numbering of* K *can be permuted).*

Note that the vertices of the mapped cell are simply the images of the reference cell's vertices under R_K*:* $\mathcal{E}^0(K) = R_K(\mathcal{E}^0(\hat{K}))$ *(edges and faces are analogously).*

Higher order reference mappings are possible but will not be discussed in this work. Note that the non-degeneracy condition already rules out parametrised cells K with $\mathrm{vol}(K) = 0$, and also nonconvex hypercubes or e.g. quadrilaterals degenerated to triangles. Also note that the edges of a mapped cell K will always be straight, but the faces of a 3-hypercube will in general not be planar. If a mapped cell K of shape dimension $s > 1$ is non-degenerate, its $(s-1)$-dimensional faces are also non-degenerate mapped cells of shape dimension $(s-1)$.

Definition 3 (Mesh)**.** *Let* $\Omega_h \subset \mathbb{R}^d, d = 2, 3$ *be a polygonally bounded domain. A finite set* $\mathcal{T}_h$ *of d-simplices or d-hypercubes is called* partitioning of Ω_h *or* mesh on Ω_h *iff* $\bar{\Omega}_h = \bigcup_{K \in \mathcal{T}_h} K$.

This partitioning is called conforming *or* proper *iff*

1. $\forall K_0, K_1 \in \mathcal{T}_h, K_0 \neq K_1 : \mathring{K}_0 \cap \mathring{K}_1 = \emptyset$,
2. $\forall K_0 \in \mathcal{T}_h : \forall$ *(d−1)-dimensional sub-simplices or sub-hypercubes* K'*:* $K' \subset \partial\Omega_h \quad$ or $\exists! K_1 \in \mathcal{T}_h \setminus \{K_0\}$ *such that* K' *is a* $(d-1)$*-dimensional sub-simplex or sub-hypercube of* K_1.

Furthermore

1. $\forall K \in \mathcal{T}_h : \mathring{K}$ *is called a (d-)*cell *of* $\mathcal{T}_h$ *and* $\mathcal{E}^d(\mathcal{T}_h) := \{\mathring{K} : K \in \mathcal{T}_h\}$ *is the set of all d-cells.*
2. *For* $s \in \{1, \dots, d-1\}$*, the s-dimensional sub-hypercubes or sub-simplices* K' *are called s-cells and* $\mathcal{E}^s(\mathcal{T}_h) = \bigcup_{K \in \mathcal{T}_h} \mathcal{E}^s(K)$ *is the set of all s-cells of* $\mathcal{T}_h$.
3. $\mathcal{E}^0(\mathcal{T}_h) = \bigcup_{K \in \mathcal{T}_h} \mathcal{E}^0(K)$ *is the set of all vertices of* $\mathcal{T}_h$.

For all $d = 1, 2, 3$ we will call d-cells simply cells *and $(d-1)$-cells* facets. *Note that all s-cells are open, meaning that we have a disjoint partitioning $\Omega_h = \biguplus_{s=0}^{d} \mathcal{E}^s(\mathcal{T}_h)$.*

Definition 4 (Lagrange-1 spaces)**.** *Let Ω_h as in Definition 3 with conforming mesh $\mathcal{T}_h$. Then define*

$$V_h(\mathcal{T}_h) := \{v \in \mathcal{C}^0(\Omega_h) : \forall K \in \mathcal{T}_h : v \circ R_K^{-1} \in \begin{cases} \mathbb{P}_1(\hat{S}), & \hat{K} = \hat{S} \\ \mathbb{Q}_1(\hat{Q}), & \hat{K} = \hat{Q} \end{cases}\}.$$

As degrees of freedom, the values in the vertices $\mathcal{E}^0(\mathcal{T}_h)$ are taken.

2.2 Orientation preserving mappings and admissible deformations

Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$ and $\partial\Omega \in \mathcal{C}^{0,1}$ with a decomposition $\partial\Omega = \biguplus_{m=0}^{d-1} \partial\Omega^m$, where $\partial\Omega^0$ is a set of singular points and $\partial\Omega^m$ are sets of relatively open, smooth m-dimensional manifolds. Also let Ω_h be a polygonal approximation of Ω with mesh $\mathcal{T}_h$.

Definition 5 (Orientation and boundary preserving mappings)**.** *Let $\Phi : \Omega \to \mathbb{R}^d$.*

1. *Φ is* orientation preserving *iff $\forall x \in \Omega : \nabla\Phi(x) \in \mathrm{SL}_d := \{A \in \mathbb{R}^{d\times d} : \det(A) > 0\}$.*
2. *Φ is* boundary preserving *iff $\forall x \in \partial\Omega : \Phi(x) \in \partial\Phi(\Omega)$.*
3. *Φ is* boundary part preserving *iff $\forall m = 0, \dots, d-1 : \forall \Gamma \in \partial\Omega^m : \forall x \in \Gamma : \Phi(x) \in \Gamma$.*

Definition 6 (The spaces of admissible deformations)**.**

1. *The space of admissible deformations of Ω is*

$$\tilde{\mathcal{D}}(\Omega) := \{\Phi : \Omega \to \mathbb{R}^d : \Phi \text{ is orientation and boundary preserving}\} \tag{1}$$

and its discretisation

$$\tilde{\mathcal{D}}_h(\Omega, \mathcal{T}_h) := \tilde{\mathcal{D}}(\Omega) \cap V_h(\mathcal{T}_h). \tag{2}$$

2. *The space of admissible variations of Ω is*

$$\mathcal{D}(\Omega) := \{\Phi : \Omega \to \mathbb{R}^d : \Phi \text{ is orientation and boundary part preserving}\} \tag{3}$$

and its discretisation

$$\mathcal{D}_h(\Omega, \mathcal{T}_h) := \mathcal{D}(\Omega) \cap V_h(\mathcal{T}_h). \tag{4}$$

To these spaces, boundary conditions can be applied. In this work, only the situation $\Gamma_1, \Gamma_2 \subset \partial\Omega$, $\mathrm{vol}_{d-1}(\partial\Omega \setminus (\Gamma_1 \cup \Gamma_2)) = 0$ will be considered with:

3. *Displacement boundary condition: If $\phi_1 : \Gamma_1 \to \mathbb{R}^d$ is sufficiently smooth, require that*

$$\forall x \in \Gamma_1 \ d\sigma \ a.e. : \Phi(x) = \phi_0(x). \tag{5}$$

4. *Unilateral boundary condition of place: Require that*

$$\forall x \in \Gamma_2 \ d\sigma \ a.e. : \Phi(x) \in \Gamma_2. \tag{6}$$

The notion of boundary part preserving mappings and variations was introduced in [46] and is very useful for dealing with smooth parts of the boundary. Note that in the definition above, the boundary preserving property of $\Phi_h \in \mathcal{D}_h(\Omega, \mathcal{T}_h)$ is still with regard to the real domain Ω. This can be weakened to use Ω_h in practice (e.g. if a smooth parametrisation of $\partial\Omega$ is not available), if $\partial\Omega_h$ is a sufficiently good approximation of $\partial\Omega$.

Lemma 1. *With Ω, Ω_h and $\mathcal{T}_h$ above, let $\Phi_h \in \mathcal{D}(\Omega, \mathcal{T}_h)$. If $\mathcal{T}_h$ is a conforming mesh on Ω_h, so is the mapped mesh $\Phi_h(\mathcal{T}_h)$.*

Proof. Two properties need to be verified.

1. $\forall \Phi_h(K_0) \in \Phi_h(\mathcal{T}_h) : \forall$ $(d-1)$-dimensional sub-simplices or sub-hypercubes $\Phi_h(K')$: $\Phi_h(K') \subset \partial\Omega_h$ or $\exists! \Phi_h(K_1) \in \Phi_h(\mathcal{T}_h) \setminus \{\Phi_h(K_0)\}$ such that $\Phi_h(K')$ is a $(d-1)$-dimensional sub-simplex or sub-hypercube of $\Phi_h(K_1)$.

   If $K' \subset \partial\Omega_h$, all associated vertices lie on $\partial\Omega$ and are thus again mapped to Ω by Φ_h (since it it boundary preserving) so that $\Phi_h(K') \in \partial\Phi_h(\Omega_h)$.

   If K' is a a $(d-1)$-dimensional sub-simplex or sub-hypercube of $\bar{K}_1$, the same is true for $\Phi_h(K')$ and $\overline{\Phi_h(K_1)} = \Phi_h(\bar{K}_1)$ because of the continuity of Φ_h.

   That there is no other $K_2 \neq K_1$ such that $\Phi_h(K') \subset \overline{\Phi_h(K_2)}$ follows from the next property.

2. $\forall \Phi_h(K_0), \Phi_h(K_1) \in \Phi_h(\mathcal{T}_h), K_0 \neq K_1 : \Phi_h(K_0) \cap \Phi_h(K_1) = \emptyset$.

   This immediately follows from the orientation preserving property of Φ_h, as a violation would require at least one cell of negative oriented volume.

□

3 Computing orientation preserving deformations

Given a boundary deformation, computing an extension into the interior of the domain is not a difficult task and can be done with a plethora of linear methods, even in moderately complex geometries. However, ensuring that the deformation is orientation preserving is very difficult.

Let Ω, Ω_h and $\mathcal{T}_h$ as in Section 2.2 and assume for now that only displacement boundary conditions (Definition 6.3) are to be imposed. For this, let $\phi_1(\partial\Omega)$ be the new, known position of the boundary $\partial\Omega$. To this new position, a deformation of the whole mesh $\mathcal{T}_h$ is to be computed to obtain $\mathcal{T}_h'$ and thus Ω_h'.

3.1 Convex functionals not enforcing the orientation preserving property

Let

$$\mathbf{a} : H^1(\Omega)^d \times H^1(\Omega)^d \to \mathbb{R} \tag{7}$$

be a coercive and continuous bilinear form. A popular way to compute extension operators is to discretise (7)

$$\mathbf{a}_h : V_h(\mathcal{T})^d \times V_h(\mathcal{T}_h)^d \to \mathbb{R} \tag{8}$$

and minimise the energy of a deformation in the energy norm induced by $\mathbf{a}_h$ by finding $\Psi \in V_h(\mathcal{T}_h)$ such that $\Psi_{|\partial\Omega_h} = \phi_{1|\partial\Omega_h}$ and $\Psi^* = \frac{1}{2}\operatorname{argmin}_{\Psi \in V_h(\mathcal{T}_h)} \mathbf{a}_h(\Psi, \Psi)$.

Examples for the bilinear form $\mathbf{a}(\cdot,\cdot)$ are:

1. $\mathbf{a}(\Psi, \eta) = \int_\Omega (\nabla\Psi : \nabla\eta) dx$ In this case, the functional is quadratic and measures the harmonic energy of the deformation. This is computationally cheap, gives d decoupled equations and very efficient numerical methods like geometric multigrid are available. However, the decoupling of the components together with the known maximum principles for harmonic functions [19, Chapter 2.2] often results in a violation of the orientation preserving property in the case of large boundary deformations and/or nonconvex domains. In fact, Figure 1 shows meshes generated with this functional.

2. $\mathbf{a}(\Psi, \eta) = \int_\Omega (\mathbf{D}(\Psi) : \mathbf{D}(\eta)) dx$ with the symmetric part $\mathbf{D}(\Psi) = \frac{1}{2}(\nabla\Psi + (\nabla\Psi)^T)$ of the deformation gradient. This can be seen as minimising finite strain energy without incompressibility constraint. The d components are coupled, resulting in considerable more effort, but still efficient numerical methods are available. Coupling the components makes the meshes generated by this functional less prone to cell intersections if the domain becomes nonconvex.

3. $\mathbf{a}(\Psi, \eta) = \int_\Omega (\Delta\Psi : \Delta\eta) dx$ In this case, the functional measures the biharmonic energy of the deformation and requires at least $\Psi, \eta \in H^2(\Omega)$. It has successfully been used for mesh optimisation (see e.g. [51]) but trying to take advantage of the ability to impose boundary conditions for $\partial_\nu \Psi$ to control the spacing in a boundary layer (see [30]) leads to boundary conditions of the third kind. In this case, a mixed formulation becomes coupled and is challenging to solve numerically [29, 38].

These examples have in common that they do not enforce the orientation preserving property directly, so it is entirely coincidental if it holds.

3.2 A class of nonlinear functionals enforcing the orientation preserving property directly

The class of nonlinear mesh quality functionals first presented in [46] expanded is to be exanded here. A mesh quality functional is already discrete, but in the process it is easy to see how its construction carries over to a continuous version, which can be exploited.

We are looking for a functional $\mathcal{F} : \mathcal{D}(\Omega) \to \mathbb{R}$, so it is natural so assume it can be written as

$$\mathcal{F}(\Phi) = \int_\Omega \mathcal{L}(x, \Phi) dx, \tag{A1}$$

which is already a (sensible) assumption on the regularity and takes the form of a *strain energy*. In practice, we need $\mathcal{F}_h : \mathcal{D}_h(\Omega, \mathcal{T}_h) \to \mathbb{R}$, so the basic axiom is formulated (as in [46]) stronger still:

Axiom 1 (Locality)**.** *A mesh quality functional $\mathcal{F}_h$ of a variation $\Phi_h \in \mathcal{D}_h(\Omega, \mathcal{T}_h)$ should be a weighted sum of local functionals of the form*

$$\mathcal{F}_h(\Phi_h) = \int_\Omega \mathcal{L}(x, \Phi_h) dx = \sum_{K \in \mathcal{T}_h} \mu_K \int_K \mathcal{L}_K(x, \Phi_h) dx, \tag{A2}$$

where the weights fulfil $\forall K \in \mathcal{T}_h : \mu_K > 0, \sum_{K \in \mathcal{T}_h} \mu_K = 1$.

This already implies *locality* in the sense that the functional value is comprised of local functionals independent of each other, meaning the quality of a cell only depends on that cell itself.

Define the local functionals by

$$\mathcal{F}_h(K, \Phi_h) := \int_K \mathcal{L}_K(x, \Phi_h) dx.$$

The next assumption is *translation invariance*, which means that the quality of a cell only depends on its shape, not its position.

$$\begin{aligned} &\forall c \in \mathbb{R}^d : && \mathcal{F}_h(K, \Phi_h + c) = \mathcal{F}_h(K, \Phi_h) \\ \Rightarrow\ &\forall K \in \mathcal{T}_h : && \exists \mathcal{L}_K^G : K \times \mathrm{SL}_d \to \mathbb{R} : \mathcal{L}_K(\cdot, \Phi_h) = \mathcal{L}_K^G(\cdot, \nabla\Phi_h) \end{aligned} \tag{A3}$$

From now on, $\mathcal{L}_K^G(\cdot, \nabla\Phi_h)$ will be just denoted by $\mathcal{L}_K(\cdot, \nabla\Phi_h)$ again and $\mathcal{L}(\cdot, \nabla\Phi_h)$ will be used in place of $\mathcal{L}(\cdot, \Phi_h)$.

Assume that K is given by the reference cell $\hat{K}$ and the reference mapping $R_K : \hat{K} \to K$. We have the relations $\Phi : \mathcal{T}_h \to \Phi(\mathcal{T}_h)$, $\Phi_{|K}(x) = \Phi(R_K(\hat{x}))$, where $R_K(\hat{x}) = x$ and $\Phi \circ R_K : \hat{K} \to \Phi(R_K(\hat{K}))$, $R_K(\Phi) = \Phi \circ R_K$, see Figure 2.

Ruling out any other dependencies, we deduce that

$$\mathcal{L}_K(\cdot, \nabla\Phi) = \mathcal{L}(\nabla R_K(\Phi)(\cdot)).$$

Furthermore, *frame indifference* is assumed, which means that a cell's quality does not depend on the observer's position:

$$\forall Q \in SO_d : \mathcal{L}\left(\nabla R_K(\Phi)(\cdot)\right) = \mathcal{L}\left(Q \nabla R_K(\Phi)(\cdot)\right). \tag{A4}$$

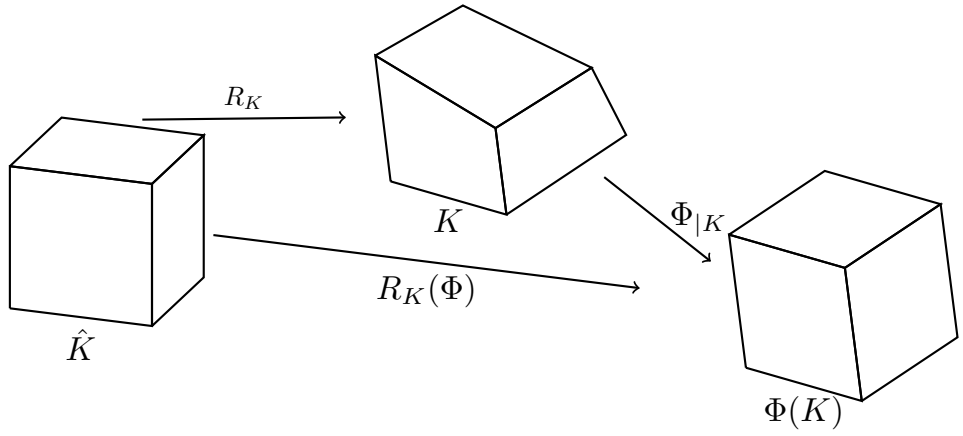


Figure 2: The relations of the mapping R_K.

Finally, the last assumption is the *stability property*

$$\lim_{\det(A)\to 0} \mathcal{L}_K(\cdot, A) = \infty, \tag{A5}$$

which forces minimising sequences to stay in the space of orientation preserving mappings. Note that the assumptions (A1) to (A5) are also valid for the continuous case, replacing the piecewise constant coefficients μ by a function of suitable integrability.

The task is now to establish the existence of a minimiser for this class of functionals (with further assumptions).

3.3 Relation to hyperelastic materials, polyconvexity

The simplest case is that the integrand $\mathcal{L}$ of the continuous functional $\mathcal{F}$ is convex in the sense that $\forall x \in \Omega : \mathcal{L}(x, \cdot) : \mathrm{SL}_d \to \mathbb{R}$ is convex. If it is furthermore strictly convex, the strain energy (A1) has at most one stationary point. In fact, all the functionals in Section 3.1 were convex, and the assumptions (A1) to (A4) hold, but (A5) does not.

However, the property $\lim_{\det(A)\to 0} \mathcal{L}(x, A) = +\infty, A \in \mathrm{SL}_d$ is critical for the deformation to be orientation preserving, and [14, Theorem 4.8-1] shows that convexity of $\mathcal{L}$ contradicts this behaviour.

Apart from the mathematical reasoning, there is also a physical argument from elasticity theory. It turns out that the class of mesh quality functionals presented in Section 3.2 is nothing more than a special case of a stored energy functional of a compressible, hyperelastic material.

The first assumption was *translation invariance* (A3), so the local integrand only depends on $\nabla\Phi(x)$. Since we then ruled out any further dependencies and further assumed *frame indifference*, we basically assumed that the underlying Piola-Kirchhoff stress tensor is of the form

$$\exists \hat{T} : \Omega \times \mathrm{SL}_d : \forall x \in \Omega : T(x) = \hat{T}(x, \nabla\Phi),$$

which means it describes an *elastic material*. Prior to this, assumption (A1) was that the functional value (which is the strain energy of the deformation Φ) can be expressed in integral form. This carries the assumption that the response function $\hat{T}$ is related to a *stored energy function* $\mathcal{L}(x, \nabla\Phi)$ by

$$\forall A \in \mathrm{SL}_d : \hat{T}(x, A) = \frac{\partial \mathcal{L}}{\partial A}(x, A),$$

which means the material is already *hyperelastic*. This makes the constitutive equation

$$\forall x \in \Omega : -\operatorname{div}\left(\frac{\partial \mathcal{L}}{\partial A}(x, \nabla\Phi(x))\right) = 0$$

formally equivalent to the equations

$$\forall \eta : \bar{\Omega} \to \mathbb{R}, \eta_{|\Gamma_0} = 0 : \mathcal{F}'(\Phi^*)\eta = 0 \tag{9}$$

and that a minimiser of the *total energy* (which is the same as the strain energy in our case without body or surface forces) is a solution of the boundary value problem defined by the constitutive equations, see [14, Theorem 4.1-2] for the details. Since we do not pose the incompressibility condition $\det(\nabla\Phi) = 1$, the material is compressible.

In this context the stability property (A5) is the physically motivated notion that "infinite stress must accompany extreme strains" [1] and consequently that volumes can only be annihilated by infinite force, which is not possible if $\mathcal{L}$ is convex. Strict convexity would also imply the uniqueness of the stationary point, which contradicts the lack of uniqueness of solutions to elasticity problems observed in physical situations see [27, 40] for examples.

So indeed it is not feasible to work with a convex stored energy function $\mathcal{L}$. However, Ball [3] was able to replace this by the weaker requirement that $\mathcal{L}$ is *polyconvex* and prove the existence theorems stated in Section 3.4.

Definition 7 (Polyconvexity)**.**

1. *Define the finite dimensional space $E = E_d \times \mathbb{R}$, where*
$$E_1 := \emptyset, \quad E_2 := \mathbb{R}^{2\times 2}, \quad E_3 := \mathbb{R}^{3\times 3} \times \mathbb{R}^{3\times 3}.$$
2. *Define the map $\iota : \mathbb{R}^{d\times d} \to E$ by*
$$\iota(A) = \begin{cases} A, & d = 1 \\ (A, \det(A)), & d = 2 \\ (A, \mathrm{Cof}(A), \det(A)), & d = 3 \end{cases}.$$
3. *A stored energy function $\mathcal{L} : \bar{\Omega} \times \mathrm{SL}_d \to \mathbb{R}$ is called* polyconvex*, iff*
$$\forall x \in \bar{\Omega} : \exists \mathcal{L}_c(x, \cdot) : E \to \mathbb{R} \text{ convex} : \quad \forall A \in \mathrm{SL}_d : \mathcal{L}(x, A) = \mathcal{L}_c(x, \iota(A)). \tag{10}$$

Remark 1. *Going into the details of polyconvex stored energy functions is beyond the scope of this work and the reader is referred to* [3, 14]*. Important cases of materials with polyconvex stored energy functions are Ogden's materials, compressible neo-Hookian materials, compressible Mooney-Rivlin materials and Hadamard-Green materials, see* [14, Chapter 4.10]*.*

With the notion of polyconvexity, it is now possible to apply the theoretical results from [3, Theorems 7.3 and 7.6] or from [14, Theorems 7.7-1, 7.7-2, 7.8-1, 7.8-2] to establish the existence of minimisers.

3.4 Existence of minimisers

The following is Theorem 7.7-1 from [14] and for the case $d = 3$ only, as the other cases are easier to prove (see the proof of [3, Theorem 7.3]).

Theorem 1 (Existence of minimisers for pure displacement problems)**.** *Assume $\Omega \subset \mathbb{R}^3$ to be a given domain such that $\partial\Omega = \Gamma_0 \cup \Gamma_1$, Γ_i $d\sigma$-measurable and $\mathrm{vol}_2(\Gamma_0) > 0$. Assume further that we have $\mathcal{L} : \Omega \times \mathrm{SL}_3 \to \mathbb{R}$ with the following properties:*

1. ***Polyconvexity:***
$$\forall x \in \Omega \text{ a.e.} : \exists \mathcal{L}_c(x, \cdot) : \iota(A) : \quad \mathcal{L}(x, A) = \mathcal{L}_c(x, \iota(A)),$$
where $\forall (F, H, \delta) \in \iota(A) : \mathcal{L}_c(\cdot, F, H, \delta) \in L^1(\Omega)$.
2. ***Stability:***
$$\forall x \in \Omega \text{ a.e} : \lim_{\det(A)\to 0} \mathcal{L}(x, A) = +\infty.$$
3. ***Coerciveness:***
$$\exists \alpha \in \mathbb{R}_+, \beta \in \mathbb{R}, 2 \le p \in \mathbb{N}, \frac{p}{p-1} \le q \in \mathbb{N}, 1 < r \in \mathbb{R} :$$
$$\forall x \in \Omega \text{ a.e.}, \forall A \in \mathrm{SL}_3 :$$
$$\mathcal{L}(x, A) \ge \alpha(\|A\|_F^p + \|\mathrm{Cof}(A)\|_F^q + \det(A)^r) + \beta.$$

Let $\phi_0 \in L^1(\Gamma_0, \mathbb{R}^3)$ *such that*

$$\emptyset \neq \mathcal{D}_{\phi_0} := \{\Phi \in W^{1,p}(\Omega) : \mathrm{Cof}(\nabla\Phi) \in L^q(\Omega), \det(\nabla\Phi) \in L^r(\Omega), \\ \forall x \in \Omega \ a.e. : \det(\nabla\Phi)(x) > 0, \quad \forall x \in \Gamma_0 \ d\sigma \ a.e. : \Phi(x) = \phi_0(x)\}.$$

Let $f \in L^p(\Omega)$ *and* $g \in L^s(\Gamma_1)$ *such that the linear form*

$$l : W^{1,p}(\Omega) \to \mathbb{R}, \quad l(\Phi) := \int_\Omega f \cdot \Phi dx + \int_{\Gamma_1} g \cdot \Phi d\sigma$$

is continuous and define

$$\mathcal{F}(\Phi) := \int_\Omega \mathcal{L}(x, \nabla\Phi(x))dx - l(\Phi).$$

Under the assumption that $\exists \Phi \in \mathcal{D}_{\phi_0} : \mathcal{F}(\Phi) < +\infty$*, there exists at least one*

$$\Phi^* \in \mathcal{D}_{\phi_0} : \mathcal{F}(\Phi^*) = \inf_{\Phi \in \mathcal{D}_{\phi_0}} \mathcal{F}(\Phi).$$

Proof. See [14, Theorem 7.7-1] and the corresponding proof. □

Remark 2.

1. *As the result is from mathematical elasticity, it contains a right hand side* f*, which has not been used in this work.*
2. *The condition* $\mathcal{D}_{\phi_0} \neq \emptyset$ *is a condition on* ϕ_0*, as in general, the traces of* $W^{1,p}(\Omega)$ *functions might not have enough regularity* [14, Theorem 6.1-7]*. In the context of deformations, the boundary deformation still has to admit* some *orientation preserving deformation of* Ω*, meaning it should not lead to self-intersections of the boundary.*
3. *If* $\mathrm{vol}_2(\Gamma_1) = 0$*, the resulting problem is a* pure displacement problem*.*
4. *The minimiser is not unique in general, which is observed for hyperelastic materials in practice.*
5. *Results for more general boundary conditions (e.g. unilateral boundary conditions of place) are available, see* [14, Theorems 7.8-1 and 7.8-2]*.*

Remark 3 (Euler-Lagrange equations)**.** *As 1 does not prove the existence of minimisers in a constructive manner, how to compute them in practice is an open problem. One possibility is to use the Euler-Lagrange equations* $\forall \theta \in \mathcal{C}^\infty(\Omega), \theta_{|\Gamma_1=0}$ *:*

$$\int_\Omega \frac{\partial \mathcal{L}}{\partial A}(x, \nabla\Phi(x)) : \nabla\theta dx = \int_\Omega f dx, \tag{11}$$

but this is only equivalent to the equations

$$\mathcal{F}'(\Phi^*)\theta = 0. \tag{12}$$

under very restricting growth assumptions on $\mathcal{L}$ [3, Theorem 7.12] *or the* a priori *assumption* $\Phi^* \in W^{1,\infty}$*, as noted in* [14, Chapter 7.10]*. To the best of this author's knowledge, no progress has been made in establishing realistic assumptions that guarantee sufficient regularity of the minimiser* Φ^**. However, the existence results are formulated for very general boundary and load data, which might exhibit cavitation or fracture where the minimiser cannot be expected to lie in* $W^{1,\infty}(\Omega)$*.*

If a critical point is found using the implicit function theorem, it cannot be guaranteed that it is indeed a local minimiser in $W^{1,p}(\Omega)$ *for* $1 \leq p < \infty$ [44]*.*

In practice, discretising the functional $\mathcal{F}$ *to* $\mathcal{F}_h$ *also restricts the solution space to a subspace of* $W^{1,\infty}(\Omega)$*. In some cases, the minimiser might not lie in the solution space, so the Euler-Lagrange equations might not be solvable. In the field of mesh optimisation, this is generally not fatal, as having an orientation preserving mapping is more important than finding the optimal mesh.*

4 Mesh quality functionals optimising cell shape and size

In this section, mesh quality functionals that are of practical use are to be derived.

4.1 Optimal cells

Up to this point, the standard reference mapping $R_K : \hat{K} \to K$ was used to define the integrands $\mathcal{L}_K$, where $\hat{K} = \hat{S}$ or $\hat{K} = \hat{Q}$ depending on the mesh's shape type.

Apart from being orientation preserving, no requirements have been made for Φ^*.While it is possible to then construct $\mathcal{L}_K$ such that the mesh defined through the minimiser Φ^* consists of well-shaped and scaled cells, it is easier and more intuitive to follow the approach of [46] and define *optimal reference cells*.

The idea is that for every cell $K \in \mathcal{T}_h$, an optimal reference cell $\hat{K}_R$ can be defined. That means for every cell functional $\mathcal{L}_K$, obviously

$$Q\ id + c = \operatorname{argmin} \mathcal{L}(\nabla R_{K_R}(\Phi)) \tag{13}$$

since this implies $\Phi(K) = Q\hat{K}_R + c$ where $Q \in \mathrm{SO}_d := \{A \in \mathbb{R}^{d\times d} : |\det(A)| = 1\}$, meaning $\Phi(K)$ is just a rotated and translated version of $\hat{K}_R$.

The question is now how to choose the reference cells $\hat{K}_R$. One very important class are reference cells in which all edges have the same length, all interior angles are the same and they are *normalised* in some sense (also see Figure 3).

Hypercubes :

$$\hat{Q}_n = [-1, 1]^d = \hat{Q}. \tag{14}$$

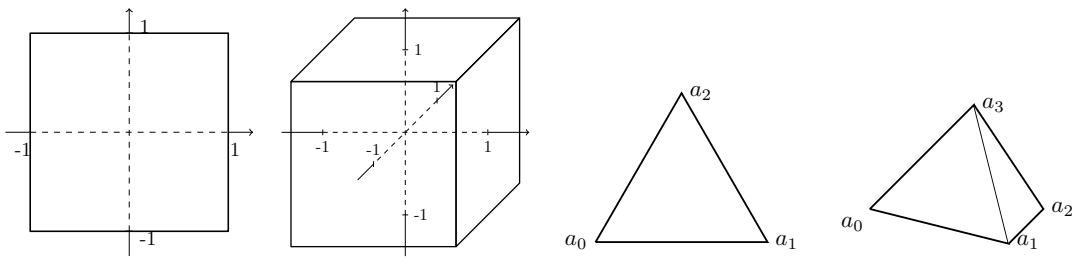
Simplices :

$$\hat{S}_n = \left\{ x \in \mathbb{R}^d : x = \sum_{i=0}^{s} \lambda_i a_i, \text{ where } \forall i \in \{1, \dots, s\} : \lambda_i \in \mathbb{R}_{\geq 0}, \sum_{i=0}^{s} \lambda_i = 1 \right\} \tag{15}$$

with

$$\begin{cases} a_0 = (0,0)^T, a_1 = (1,0)^T, a_2 = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right), & d = 2 \\ a_0 = (0,0,0)^T, a_1 = (1,0,0)^T, a_2 = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}, 0\right), a_3 = \left(\frac{1}{2}, \frac{\sqrt{3}}{6}, \frac{\sqrt{6}}{3}\right) & d = 3 \end{cases}. \tag{16}$$

The hypercubes $\hat{Q}_n$ are just the regular reference d-hypercubes (see Figures 3a and 3b), while the *normalised* simplices $\hat{S}_n$ are the equilateral triangle (Figure 3c) and the standard tetrahedron (Figure 3d).



(a) $\hat{Q}_n$ for $d = 2$. (b) $\hat{Q}_n$ for $d = 3$. (c) $\hat{S}_n$ for $d = 2$. (d) $\hat{S}_n$ for $d = 3$

Figure 3: Reference cells for $d = 2, 3$.

These reference cells have the additional benefit of being *isotropic*, which already implies a special structure of the local functional $\mathcal{L}_K$, which will be discussed in Section 4.2. It is clear that every isotropic reference cell can only be a scaled and rotated version of such a normalised reference cell (see [46, Section 5] for the simplex case and [41, Lemma 3.5] for the generalisation to hypercubes). The aspect of scaling the reference cell will be discussed in Section 4.4.

Remark 4. *For simplicity, the mapping from the optimal reference cell $\hat{K}_R$ for mesh optimisation will also be denoted $R_K : \hat{K}_R \to \mathbb{R}^d$ as the standard reference mapping will not appear again.*

4.2 Isotropy and the Rivlin-Ericksen representation theorem

Additionally to the assumptions (A1) to (A5), we now want the local functional to be *isotropic*:

$$\forall Q \in SO_d : \mathcal{L}(\nabla R_K(\Phi), \cdot) = \mathcal{L}(\nabla R_K(\Phi)(\cdot)Q), \tag{A6}$$

which means that the functional does not depend on the coordinate system of the reference cell $\hat{K}_R$.

For every matrix $A \in \mathbb{R}^{d\times d}$, there exists a left polar decomposition $A = Q(A^TA)^{\frac{1}{2}}$ with $Q \in \mathrm{SL}_d$. From the frame indifference, it follows for fixed $x \in K \in \mathcal{T}_h$ that

$$\exists \mathcal{L}_l : \mathrm{SL}_d \to \mathbb{R} : \mathcal{L}_l\left((\nabla R_K(\Phi)(x))^T(\nabla R_K(\Phi)(x))\right) = \mathcal{L}(\nabla R_K(\Phi)(x)).$$

Similarly, from the isotropy and the existence of the right polar decomposition $A = (A^TA)^{\frac{1}{2}}Q$ with $Q \in \mathrm{SL}_d$ it follows that

$$\exists \mathcal{L}_r : \mathrm{SL}_d \to \mathbb{R} : \mathcal{L}_r\left((\nabla R_K(\Phi)(x))(\nabla R_K(\Phi)(x))^T\right) = \mathcal{L}(\nabla R_K(\Phi)(x)).$$

With the Rivlin-Ericksen Representation Theorem [14, Theorem 3.6-1], we deduce that

$$\begin{aligned} &\exists L : \mathbb{R}\times\mathbb{R}\times\mathbb{R} \to \mathbb{R}\cup\{\infty\} : \\ &\mathcal{L}(x, \nabla R_K(\Phi)) = L(\|\nabla R_K(\Phi)(x)\|_F^2, \|\operatorname{Cof}(\nabla R_K(\Phi)(x))\|_F^2, \det(\nabla R_K(\Phi)(x))) \end{aligned} \tag{17}$$

This is formulated independently of d, since for $A \in \mathrm{SL}_d$:

$$\begin{aligned} d = 1 :& \|A\|_F^2 = \|\operatorname{Cof}(A)\|_F^2 = \det(A) > 0, \\ d = 2 :& \|A\|_F^2 = \|\operatorname{Cof}(A)\|_F^2. \end{aligned}$$

The terms $\|A\|_F^2$, $\|\operatorname{Cof}(A)\|_F^2$, $\det(A)$ are the *principal invariants* of the matrix A. The Rivlin-Ericksen theorem states that a tensor-valued mapping that is frame indifferent and isotropic on SL_d can only be a polynomial in the input argument with coefficients that only depend on these principal invariants. Here we have the special case of a real-valued mapping, so only the term of order zero appears in the representation.

Since $\hat{K}_R$ is the optimal cell for the quality measure defined by $\mathcal{L}_K$

$$L(x, id) = \min_{A\in SL_d} L(\|A\|_F^2, \|\operatorname{Cof}(A)\|_F^2, \det(A)).$$

As $\|id\|_F^2 = d, \|\operatorname{Cof}(id)\| = d, \det(id) = 1$, we demand L to fulfil the (stronger) condition

$$L(d, d, 1) = \min_{a\in\mathbb{R}^3} L(a). \tag{18}$$

For $d = 3$, an example is

$$\begin{aligned} \mathcal{L}_K(\nabla R_K(\Phi)) &= L\left(\|\nabla R_K(\Phi)\|_F^2, \|\operatorname{Cof}(\nabla R_K(\Phi))\|_F^2, \det(\nabla R_K(\Phi))\right), \\ L(a) &= \sum_{i=1}^{3} \alpha_i l_i(a_i), \\ l_1(z) &= (z-d)^2 = l_2(z), \\ l_3(z) &= \det(\nabla R_K(\Phi))^r + \frac{c(r)}{(\det(\nabla R_K(\Phi)) + |\det(\nabla R_K(\Phi))|)^r}, \end{aligned} \tag{19}$$

where $\alpha_1, \alpha_3 > 0$, $\alpha_2 \geq 0$, $r > 1$ and $c(r)$ chosen so that $l_3'(1) = 0, l_3''(1) > 0$ hold.

Remark 5 (Geometric interpretations of the terms)**.** *The terms* $\|\nabla R_K(\Phi)\|_F^2$, $\|\operatorname{Cof}(\nabla R_K(\Phi))\|_F^2$ *and* $\det(\nabla R_K(\Phi))$ *have obvious geometric meanings that are well-known in mechanics and valid or suitable control volumes* K*. For* $d = 3$

1. $\|\nabla R_K(\Phi)\|_F$ *expresses the length change of curves under the deformation* $R_K(\Phi)$ [14, Section 1.8]*,*

2. $\|\operatorname{Cof}(\nabla R_K(\Phi))\nu\|_2$ *expresses the area change of surfaces under the deformation* $R_K(\Phi)$ [14, Theorem 1.7-1]*, where* ν *is the outer unit normal, and*

3. $\det(\nabla\Phi)$ *expresses the volume change of volumes under the deformation* $R_K(\Phi)$ [14, Section 1.5].

Obviously, for $d = 1$ *there is only in the change in* vol_1 *and for* $d = 2$ *we can only consider* vol_1 *and* vol_2*, corresponding to the identities in the principal invariants.*

Note that with the additional assumption of isotropy, a direct proof of the existence of an optimal boundary part preserving deformation is possible for the case of simplices [46, Theorem 1]. This proof is not easily generalised to hypercube meshes with general, d-linear reference mappings, as the notion of orientation preserving deformations is more delicate in this case.

4.3 Anisotropic reference cells

Isotropic reference cells have the additional benefit of requiring no combinatorial testing in the evaluation of the local integrands $\mathcal{L}_K$. For more general reference cells, as they are not invariant with regard to their local vertex, edge and face numbering, certain permutations of these numberings that do not change the orientation of the respective entities have to be tested. This means that $\mathcal{L}_K$ no longer solely depends on $\|\nabla R_K(\Phi)\|_F^2$, $\|\operatorname{Cof}(\nabla R_K(\Phi))\|_F^2$ and $\det(\nabla R_K(\Phi))$, but also on the variant of the reference cell.

Choosing reference cells that are not isotropic can be advantageous in the case of anisotropic meshes that e.g. resolve a thin boundary layer. In this case, one can very often directly chose the variant of the reference cell corresponding to the lowest functional value by examining the cell K itself *if* it already has the right degree of anisotropy (but not necessarily a good shape). For resolving anisotropies in partial differential equations, the use of hypercube meshes with a similar anisotropy with regard to the cell's aspect ratio is quite popular.

Assume we have a cell K for which we can compute its aspect ratios $\alpha_1, \dots, \alpha_d$, e.g. by computing some mean values $\beta_1, \dots, \beta_d$ of lengths of edges whose inverse images are aligned with the $x_1, \dots, x_d$ axis in the reference cell K. Assume that $\beta_i = \max_{j=1,\dots,d} \beta_j$. Then $\alpha_j := \beta_j/\beta_i$ gives us the anisotropic reference cell $\hat{K}_{n,A} = \operatorname{diag}(\alpha_1, \dots, \alpha_d)\hat{K}_n$.

If K is a hypercube, then this is already the reference cell variant associated with the lowest local functional value. For simplices, several possibilities would have to be treated.

As this is very problem specific, only the isotropic reference cells presented in Section 4.1 will be used from here on.

4.4 Scaling reference cells

The next question is if the functional $\mathcal{L}_K$ should be *scaling invariant*, e.g.

$$\forall\lambda \in \mathbb{R}_+ : \mathcal{L}_K(\lambda\nabla R_K(\Phi)) = \mathcal{L}_K(\nabla R_K(\Phi)).$$

Assume for simplicity that Φ is affine, which implies $\nabla\Phi = A \in \mathbb{R}^{d\times d}$.

Recalling that $\forall A \in \mathbb{R}^{d\times d}$

$$\det(\lambda A) = \lambda^d \det(A), \quad \|\lambda A\|_F^2 = \lambda^2\|A\|_F^2, \quad \|\operatorname{Cof}(\lambda A)\|_F^2 = \lambda^{d-1}\|\operatorname{Cof}(A)\|_F^2,$$

we conclude that if $\mathcal{L}_K$ is scaling invariant, there exists a function $f^S : \mathbb{R}^2 \to \mathbb{R}$ such that

$$\mathcal{L}_K(\nabla\Phi) = f^S\left(\|A\|_F^{d-1}/\|\operatorname{Cof}(A)\|_F, \|A\|_F^d/\det(A)\right).$$

A functional based on f^S cannot be polyconvex as f^S is not convex in its argument $A \in \mathbb{R}^{d\times d}$ (see Section 3.3) and does not satisfy the coerciveness condition in Theorem 1 as e.g. in general $\mathcal{L}_K(A) < +\infty$ as $\det(A) \to \infty$. Nevertheless, functionals of this form are used in practice and can apparently be treated by simple numerical methods (see e.g. [22]). However, the existence of minimisers for the discrete (finite dimensional) problem cannot be deduced from results for a continuous problem. The discrete problem does not converge to a well-posed continuous problem, which may lead to difficulties for very fine meshes.

Together with the above mentioned restrictions this means that the notion of scaling invariance it not very useful in this context and we have to define a size for $\hat{K}_R$. Since we are in the isotropic case

$$\forall K \in \mathcal{T}_h : \exists h(K) \in \mathbb{R}_+ : \hat{K}_R = h(K)\hat{K}_n$$

and we call h the *optimal scales*. See Figure 4 for the connection between $R_K, R_{K,n}$ and $\Phi \circ R_K = R_K(\Phi)$.

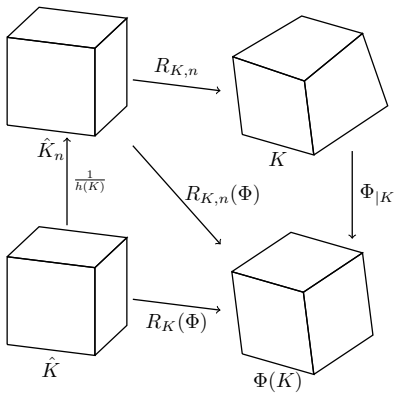


Figure 4: The relations of the mapping $R_{K,n}$.

The question is now how to chose those optimal scales. Assume that we are given a target cell size distribution, meaning we have some λ satisfying

$$\forall K \in \mathcal{T}_h : \lambda(K) > 0, \quad \sum_{K \in \mathcal{T}_h} \lambda(K) = 1.$$

Since the optimal deformation preserves the volume of Ω, this can be interpreted as

$$\lambda(K) = \frac{\mathrm{vol}(\Phi^*(K))}{\mathrm{vol}(\Omega)} = \frac{\mathrm{vol}(\hat{K}_n) \int_K \det(\nabla R_{K,n}(\Phi^*))}{\mathrm{vol}(\Omega)}.$$

In practice, this will not be exactly true since we are working on Ω_h and $\mathcal{T}_h$ instead. But if $\partial\mathcal{T}_h$ (and thus $\partial\Omega_h$) is already a good approximation of $\partial\Omega$, we could approximate this condition by conserving the volume of $\mathcal{T}_h$ instead, meaning that

$$\lambda(K) = \frac{\mathrm{vol}(\Phi^*(K))}{\mathrm{vol}(\Omega_h)} = \frac{\int_{\hat{K}_n} \det(\nabla R_{K,n}(\Phi^*))}{\sum_{T \in \mathcal{T}_h} \int_{\hat{K}_n} \det(\nabla R_{T,n}(id))}.$$

Since the identity minimises the local functional, the optimal deformation applied to a cell should result in a cell that has the corresponding reference cell's volume:

$$\mathrm{vol}(\Phi^*(K)) = \int_{\hat{K}} \det(\nabla R_K(\Phi^*))dx = \int_{\hat{K}_n} \frac{\det(\nabla R_{K,n}(\Phi^*))}{h(K)^d}.$$

For a given cell size distribution λ, we can now compute the optimal scales as

$$h(K) = \sqrt[d]{\lambda(K) \sum_{T \in \mathcal{T}_h} \int_{\hat{K}_n} \det(\nabla R_{K,n}(id))}. \tag{20}$$

Instead of the cell size distribution, one could prescribe a *mesh concentration* $c = c(K)$ and then normalise it to obtain $\lambda(K) = \frac{c(K)}{\sum_{T \in \mathcal{T}_h} c(T)}$.

Example 1 (Some mesh concentration functions)**.**

1. *Equidistribution:* $c \equiv const$*, leading to* $\lambda(K) = 1/\operatorname{card}(\mathcal{E}^d(\mathcal{T}_h))$.

2. *Preservation of the cell volume with regard to a reference configuration $\tilde{\mathcal{T}}_h$: $c(K) = |\tilde{K}|$, where $\tilde{K}$ is the corresponding cell in $\tilde{\mathcal{T}}_h$.*

3. *Preservation of cell volume with regard to the number of (local) refinements: $c(K) = b^{l(K)}$, where $l(K)$ the refinement level and b is the refinement base, e.g. $b = 2$ for bisection-based local refinement of simplices.*

4. *According to the distance to a surface Γ: $c(K) = f(\mathrm{dist}(s_{\Phi^*(K)}, \Gamma))$ for some set Γ, where $s(K)$ is the centre of gravity of K. Here, Γ could be $\partial\Omega$ or an inner boundary like a phase boundary.*

5. *According to an a posteriori error estimate: $c(K) = g(\eta(K))$, where $\eta(K)$ could come from various techniques.*

Remark 6. *Choosing optimal scales is a very direct form of r-adaptivity* [34, Chapter 6] *and is discussed at length in* [41]*, with numerical results available for both simplex and hypercube meshes in* $2d$*.*

r-adaptivity is a powerful tool to increase the space resolution in regions of interest without modifying the underlying finite element spaces, which is an advantage when using multigrid or multilevel methods. It can easily incorporated into linear mesh optimisation methods (see e.g. [50] *for an example in particulate flows). However, when small cell sizes are needed, these are often generated at the expense of poor cell shapes, as these methods cannot enforce the orientation preserving property directly.*

Presenting r-adaptivity in more detail is beyond the scope of this work, so the reader is referred to the given references.

5 Numerical methods

For the minimisation of the mesh quality functional, tools from unconstrained optimisation will be used. Much of the theory can be found in [39], for some details on nonlinear multilevel-based solvers see [12] and also [25], [24] for multilevel trust-region solvers.

Recall that we constructed the discrete functional $\mathcal{F}_h$ according to

$$\mathcal{F}_h(\Phi_h) = \int_\Omega \mathcal{L}(x, \nabla\Phi_h)dx = \sum_{K\in\mathcal{T}_h} \mu_K \int_K \mathcal{L}_K(x, \nabla R_K(\Phi_h))dx.$$

[14, Theorem 4.1-1, Theorem 4.1-2] gives that the minimisation of this functional is *formally* equivalent to the equations

$$\forall \eta \in \mathcal{D}_h^{0,\Gamma_1}(\Omega, \mathcal{T}_h) : \mathcal{F}_h'(\Phi_h^*)\eta_h = 0,$$

which can be formulated (due to the assumed hyperelasticity) as

$$\mathcal{F}_h'(\Phi_h^*)\eta_h = \int_\Omega \frac{\partial \mathcal{L}}{\partial A}(x, \nabla\Phi_h(x)) : \nabla\eta_h(x)dx,$$

where $\mathcal{D}_h^{0,\Gamma_1}(\Omega, \mathcal{T}_h) := \{\eta_h \in \mathcal{D}_h(\Omega, \mathcal{T}_h) : \eta_{h|\Gamma_1} = 0\}$.

Remark 7.

1. *Note that for quadratic functionals like the $\mathbf{D}(u) : \mathbf{D}(v)$ functional, the gradient is simply a matrix, which can be assembled and used in a linear solver. Here, the discrete gradient is a vector valued nonlinear function. Any iterative solver producing a sequence of iterates (Φ_k) will require the gradient to be evaluated at each iterate, which is very costly.*

2. *Recall that there might be problems with the Euler-Lagrange equations* (11)*.*

3. *Because $\frac{\partial \det(\nabla\Phi(x))}{\partial A} = \det(\nabla\Phi(x))(\nabla\Phi(x))^{-T} : \nabla\eta(x)$, care must be taken when using numerical integration because the integrand will be a rational functions.*

5.1 Tools for nonlinear optimisation

If $\mathcal{F}_h$ is sufficiently smooth, we could solve our nonlinear equation $\mathcal{F}_h'(\Phi_h^*) = 0$ on the finite dimensional space $\mathcal{D}_h$ with Newton's method. However:

1. The method is only convergent if we start close enough to Φ_h^*, which might be difficult to achieve.

2. If an iterate $\Phi^{(k)}$ is far away from Φ_h^*, there is nothing that guarantees that $\mathcal{F}_h''$ is nonsingular. Moreover, since the stored energy function is not convex, the Hessian $\mathcal{F}_h''$ might be indefinite, so that the *Newton direction* $d^{(k)}$ obtained by solving the (linear) system

$$\mathcal{F}_h''(\Phi^{(k)})d^{(k)} = -\mathcal{F}_h(\Phi^{(k)}) \tag{21}$$

   is not even a descent direction.

3. In our case, the Hessian $\mathcal{F}_h''(\Phi^{(k)})$ is quite expensive to compute and will in general not possess any structure that lends itself to highly efficient iterative solvers. Using a direct solver might be the only option to solve the linear system (21). However, for other problems with less pronounced nonlinearity, iterative methods have been applied successfully for solving (21) (see [12] for some examples), even if the true Hessian is not available and has to be approximated using a divided difference scheme [17].

These difficulties can be addressed in various ways which mostly revolve around approximating the Hessian $\mathcal{F}_h''(\Phi^{(k)})$. Some examples are

1. Trust region methods: Minimising a quadratic model and rejecting steps if the difference between predicted and true functional value is too large [39, Chapter 4].

2. Line search methods: Successive minimisation in search directions [39, Chapter 3].

3. Quasi Newton methods: Recursive updating of an approximation of the (inverse) Hessian [39, Chapter 6].

4. Inexact or truncated Newton methods: (21) is solved only approximately, e.g. by using an iterative solver and terminating early and/or when the Hessian is indefinite [39, Chapter 7].

As many Newton-like methods can be expressed as preconditioned line search methods, only the nonlinear conjugate-gradient (NLCG) method will be discussed in more detail.

5.2 The nonlinear conjugate-gradient method

The NLCG algorithm was introduced in [20] and is a variant of the well known linear Conjugate Gradient method [31]. The main differences are:

1. $\mathcal{F}_h$ is no longer quadratic as in the linear CG method, so the step size computation by a truncated Taylor series is no longer exact and has to be replaced by a line search.

2. In the linear case, the conjugacy of the search directions is in the sense that $d^{(k+1)}, d^{(k)}$ are $\mathcal{F}_h''$-orthogonal. In the nonlinear case, the notion of conjugacy changes in every iteration, as $\mathcal{F}_h'' = \mathcal{F}_h''(\Phi)$ is no longer constant. This means that the search directions quickly loose conjugacy. There are several non-equivalent ways to chose the search direction update parameter $\beta^{(k)}$, see Remark 8.

Remark 8 (Choosing $\beta^{(k)}$)**.** *There are many possible ways to set the search direction update parameter $\beta^{(k)}$, which are all non-equivalent but reduce to the same update as in the linear case if $\mathcal{F}_h$ is quadratic (and the nonlinear CG method reduces to the linear version).*

Define $y^{(k+1)} := \mathcal{F}_h'(\Phi^{(k+1)}) - \mathcal{F}_h'(\Phi^{(k)})$.

1. *The Hestenes-Stiefel update* [31] *sets* $\beta_{HS}^{(k+1)} = \frac{(B^{-1}\mathcal{F}_h'(\Phi^{(k+1)}), y^{(k+1)})}{(y^{(k+1)}, d^{(k)})}$.

Algorithm 1 Preconditioned nonlinear Conjugate Gradient (NLCG) algorithm.

Given an initial guess $\Phi^{(0)}$, a preconditioner $B : \mathbb{R}^d \to \mathbb{R}^d$ and an initial search direction $d^{(0)}$:

for $k = 0, \dots,$ max_iter **do**

Compute a step length $\alpha^{(k)}$ such that $\mathcal{F}_h(\Phi^{(k)} + \alpha d^{(k)}) < \mathcal{F}_h(\Phi^{(k)})$.

Set $\Phi^{(k+1)} = \Phi^{(k)} + \alpha d^{(k)}$.

Compute a new descent direction $d^{(k+1)} = -B^{-1}\ \mathcal{F}_h'(\Phi^{(k)}) + \beta^{(k)} d^{(k)}$

end for

2. *The Dai-Yuan update* [16] *sets* $\beta_{DY}^{(k+1)} = \frac{\left(B^{-1}\mathcal{F}_h'(\Phi^{(k+1)}), \mathcal{F}_h'(\Phi^{(k)})\right)}{(y^{(k+1)}, d^{(k)})}$.

3. *The Dai-Yuan-Hestenes-Stiefel hybrid update sets* $\beta_{DYHS}^{(k+1)} = \max\{0, \min\{\beta_{DY}^{(k+1)}, \beta_{HS}^{(k+1)}\}\}$.

Other well-known updates are the Fletcher-Reeves [20] *and the Polak-Ribière-Polyak update* [42,43]. *See the survey article* [28] *for even more search direction updates.*

Remark 9 (The preconditioner B)**.** *In literature dealing with nonlinear optimisation, the notion of preconditioning NLCG rarely appears (see* [28, Section 8] *for one of the exceptions). This may be due to (algebraic) preconditioners that offer good convergence properties in practice (e.g. lBFGS, which is a preconditioner for the steepest descent method) being available and the fact that many nonlinear optimisation problems are discrete in nature anyway, with no possibility of e.g. deriving a PDE-based preconditioner on the continuous level and then discretising it.*

Another point is that a preconditioner that changes from iteration (meaning $B = B^{(k)}$) introduces new difficulties in the computation of the search direction update parameter $\beta^{(k)}$ from Remark 8 as it turns NLCG into a variable metric method and influences the notion of conjugacy. Formulating NLCG with left and right preconditioning yields that e.g. the Dai-Yuan update should read

$$\beta_{DY}^{(k+1)} = \frac{\left((B^{(k+1)})^{-\frac{1}{2}}\mathcal{F}_h'(\Phi^{(k+1)}), (B^{(k)})^{-\frac{1}{2}}\mathcal{F}_h'(\Phi^{(k)})\right)}{(y^{(k+1)}, d^{(k)})}.$$

This means that one needs to compute the square roots of the preconditioners, which might be both difficult and costly. However, since the notion of conjugacy changes anyway, one might simply approximate $B^{(k)}$ by $B^{(k+1)}$, which allows the use of the corresponding formula from Remark 8.2 for left-only preconditioning.

5.3 A PDE-based preconditioner

Recall that Algorithm 1 contained a descent direction $d^{(k+1)}$ which was written as

$$d^{(k+1)} = -B^{-1}\ \mathcal{F}_h'(\Phi^{(k)}) + \beta^{(k)} d^{(k)}.$$

The operator B^{-1} is the preconditioner and needs to be symmetric and positive definite to guarantee that $-B^{-1}\ \mathcal{F}_h'(\Phi^{(k)})$ is a descent direction. Quasi Newton methods like lBFGS or DFP are algebraic in nature, as they use the discrete representations $\mathcal{F}_h'(\Phi^{(k)})$ to compute a positive definite approximation to the true Hessian $\mathcal{F}_h''(\Phi^{(k)})$.

However, it is possible to construct a preconditioner based on a second order operator for which efficient numerical methods are available. One obvious choice is the operator associated with the bilinear form

$$\mathbf{a} : \mathcal{D}^{\Phi_0,\Gamma_1}(\hat{\Omega}) \times \mathcal{D}^{0,\Gamma_1}(\hat{\Omega}) \to \mathbb{R}, \quad \mathbf{a}(\Phi, \eta) = \int_{\hat{\Omega}} \mathbf{D}(\Phi) : \mathbf{D}(\eta) dx, \tag{22}$$

where $\hat{\Omega}$ is some reference domain (see the vertex mapping (28) in Section 6.2). The discrete form

$$\mathbf{a}_h : \mathcal{D}_h^{\phi_0,\Gamma_1}(\hat{\Omega}, \hat{\mathcal{T}}_h) \times \mathcal{D}_h^{0,\Gamma_1}(\hat{\Omega}, \hat{\mathcal{T}}_h) \to \mathbb{R}, \quad \mathbf{a}_h(\Phi_h, \eta_h) = \int_{\hat{\Omega}_h} \mathbf{D}(\Phi_h) : \mathbf{D}(\eta_h) dx, \tag{23}$$

can be used to define the operator

$$A_h : \mathcal{D}_h^{\phi_0,\Gamma_1}(\hat{\Omega}, \hat{\mathcal{T}}_h) \to \mathcal{D}_h^{0,\Gamma_1}(\hat{\Omega}, \hat{\mathcal{T}}_h)', (A_h\Phi_h, \eta_h) := \mathbf{a}_h(\Phi_h, \eta_h),$$

which can be identified by a $\mathbb{R}^{n\times n}$ matrix. From now on this discrete form will be used and all quantities will be identified with their associated $\mathbb{R}^n$ coefficient vectors.

This is the motivation for defining a class of preconditioners by

$$B^{-1} : \mathcal{D}_h^{0,\Gamma_1}(\hat{\mathcal{T}}_h)' \to \mathcal{D}_h^{\phi_0,\Gamma_1}(\hat{\mathcal{T}}_h), \qquad B^{-1}d_h := \tilde{A}_h^{-1}d_h \tag{24}$$

using some approximation $\tilde{A}_h^{-1}$ to the operator A_h^{-1}. Since we do not need A_h^{-1} explicitly, but rather its *application*, we can just solve a linear system of equations. Usually the corresponding linear system of equations $A_h y = d_h$ is solved only approximately e.g. using an iterative solver, hence the definition $B^{-1}d_h := \tilde{A}_h^{-1}d_h$. This is a case where the preconditioner is motivated by the equations on the continuous level and then discretised, as opposed to e.g. the BFGS preconditioner, which is derived directly from the discrete nonlinear system.

Remark 10.

1. *The choice of the bilinear form was motivated by the fact that the functional $\mathcal{F}$ couples the components of a deformation Φ. Other choices are possible, as is adding a part coming from a zero order operator.*

2. *In the current form where $\mathbf{a}_h(\Phi_h, \eta_h) = \int_{\hat{\Omega}_h} \mathbf{D}(\Phi_h) : \mathbf{D}(\eta_h)dx$, the preconditioner does not depend on the current iterate $\Phi_h^{(k)}$ and the current mesh $\Phi_h^{(k)}(\mathcal{T}_h)$ since the integral is taken on $\hat{\Omega}_h$. It is possible to obtain a* variable metric method *by using the bilinear form*

$$\mathbf{a}_h^{(k)}(\Phi_h, \eta_h) = \int_{\Phi_h^{(k)}(\hat{\Omega}_h)} \mathbf{D}(\Phi_h) : \mathbf{D}(\eta_h)dx,$$

   which sets $\hat{\Omega}_h = \Phi^{(k)}(\Omega_h)$. This requires the reassembly of the matrix $A_h^{(k)}$ in every nonlinear solver iteration, which is undesirable.

3. *For the local functionals used, it is easy to bound $\forall A \in \mathrm{SL}_d : c_1\|A\|_F^2 \leq \mathcal{F}'_h(x, A)$. Because of the arithmetic-geometric mean inequality, we also trivially get the estimate $\forall A \in \mathrm{SL}_d : \det(A) \leq \frac{1}{d}\|A\|_F^2$. But because of the necessary regularity property $\lim_{\det(A)\to 0} \mathcal{L}(x, A) = \infty$ of the local integrand, bounding $\mathcal{F}'_h(x, A) \leq c\|A\|_F^2$ is not possible in general. This means that the preconditioner will fail to give good results if the stability term of the functional $\mathcal{F}_h$ becomes dominant.*

   On the other hand, the numerical experience in these situations is that the mesh is already nearly deteriorated if the stability term becomes dominant, which usually only happens if e.g. the boundary deformation is about to produce self intersections at the boundary. This indicates problems in whatever is governing the boundary deformation, so the arising ill-conditioning of the mesh optimisation problem is usually the least of all worries.

The way the preconditioner is constructed, derived from the continuous functional $\mathcal{F}$ on the continuous level, already gives some hints on the limitations to expect.

1. **Constant coefficients, reference domain dependent**: Varying local optimal scales h (Section 4.4) need to be expressed either by the reference mesh $\hat{\mathcal{T}}_h$ or varying coefficients.

2. **Only dependent on $\|\nabla\Phi\|_F$**: Since it the bilinear form $a_h(\cdot,\cdot)$ only uses the term $\|\nabla\Phi\|_F$, we cannot expect good results if the parts of $\mathcal{F}_h$ depending on the other invariants of $\nabla\Phi$ become dominant.

3. **Boundary conditions**: If unilateral boundary conditions are to be imposed on some parts of $\partial\Omega$, $\mathbf{a}_h(\cdot,\cdot)$ has to be defined on the same spaces. Imposing this boundary condition by the use of "simple" projection operators is not very stable with regard to preserving the orientation (see [41, Chapter 5.4.2]). More sophisticated projection operators are needed, which is beyond the scope of this work.

4. **Effective solver for** A_h: For the preconditioner to be effective, a fast solver for $A_h y = d_h$ needs to be available. In this work, geometric multigrid based methods are used, which are very efficient if $\hat{\mathcal{T}}_h$ is uniform, and may become inefficient if $\hat{\mathcal{T}}_h$ has anisotropies or cells of very different sizes. In these cases, other solvers need to be considered. Since this is just a preconditioner, it is also worth exploring how "close" $\tilde{A}_h^{-1}$ needs to be to A_h^{-1} for $\tilde{A}_h^{-1}$ to be an efficient preconditioner to the nonlinear system.

6 Numerical results

In this section, only a few numerical results can be presented. Refer to [41, Chapter 5] for a wide selection of numerical results obtained with the methods presented here applied to moving boundary problems, including r-adaptivity and surface alignment. Also, results on the preconditioners for the case of a moving nonconvex domain are presented in [41, Chapter 7].

As the quality of meshes needs to be quantified in this section, a notion of cell quality *independent* of the mesh quality functional is needed. Here, it seems intuitive to have a scaling invariant measure of the shape quality. See [41, Chapter 5.2] for considerations concerning the *cell size distribution defect*.

Definition 8 (Cell quantities)**.** *Let* $K \subset \mathbb{R}^d$*.*

1. $h(K) := \sup\{\|x_1 - x_0\|_2 : x_0, x_1 \in K\}$ *is called the* diameter *of* K*.*
2. $\rho(K) := 2 \sup\{r : \exists x_0 \in K : B_r(x_0) \subset K\}$ *is called the* in-circle diameter *of* K*.*
3. $\sigma(K) := \frac{h(K)}{\rho(K)}$ *is called the* aspect ratio *of* K*.*

To prove the convergence of a finite element solution to the solution of the continuous problem, an interpolation error estimate over a whole *family of meshes* is needed [13, Theorem 3.2.1].

Definition 9 (Regular family of meshes)**.** *Let* $(\mathcal{T}_i)_{i\in\mathbb{N}}$ *be a family of meshes. It is called a* regular family of meshes *iff*

1. $\sup\left\{\sigma(K) : K \in \bigcup_{i\in\mathbb{N}} \mathcal{T}_i\right\} =: \sigma < \infty$*,*
2. *For* $h(\mathcal{T}_i) := \max_{K\in\mathcal{T}_i} h(K)$ *it holds that* $\lim_{i\to\infty} h(\mathcal{T}_i) = 0$*.*

If there is no ambiguity, we identify $\mathcal{T}_i$ *with* $\mathcal{T}_{h_i}$ *or call the whole family of meshes just* $(\mathcal{T}_h)$*.*

We see that it is important to bound $\sigma_i = \sup\left\{\sigma(K) : K \in \bigcup_{i\in\mathbb{N}} \mathcal{T}_i\right\}$ from above for every $\mathcal{T}_i$ that is a member of a family $(\mathcal{T}_h)$ of meshes.

With the definitions of the minimum edge length $h_{\min}(K) = \min\{\mathrm{vol}(e) : e \in \mathcal{E}^1(K)\}$ and of the maximum of the cosine of angles between two edges of a cell K,

$$\gamma_{\max}(K) := \max\left\{\frac{|(v_i - v_j, v_k - v_j)|}{\|v_i - v_j\|_2 \|v_k - v_j\|_2} : \exists e_{ij}, e_{kj} \in \mathcal{E}^1(K) : v_i, v_j \in e_{ij}, v_k, v_j \in e_{jk}\right\},$$

we chose the following mesh shape quality heuristics:

Definition 10 (Shape quality heuristic)**.** *For a domain* $\Omega_h \subset \mathbb{R}^2$*, a mesh* $\mathcal{T}_h$ *in* Ω_h *and a cell* $K \in \mathcal{E}^2(\mathcal{T})$ *define*

$$\mathcal{Q}(K) := \begin{cases} \frac{h_{\min}(K)}{h(K)}\sqrt{1 - \gamma_{\max}}, & K \text{ is convex} \\ 0 & \text{else} \end{cases} \tag{25}$$

if K *is a 2-hypercube and*

$$\mathcal{Q}(K) = \frac{1}{h(K)} \sqrt[d]{\frac{\mathrm{vol}(K)}{\mathrm{vol}(\hat{S}_n)}} \tag{26}$$

if K *is a d-simplex (see also* [15, Sections 5 and 6]*), and define the* shape quality heuristic

$$\mathcal{Q}(\mathcal{T}_h) := \min_{K\in\mathcal{E}^2(\mathcal{T}_h)} \mathcal{Q}(K) \tag{27}$$

and the the average shape quality heuristic $\mathcal{Q}_a(\mathcal{T}_h) := \frac{1}{\mathrm{card}(\mathcal{E}^d(\mathcal{T}_h))} \sum_{K\in\mathcal{E}^d(\mathcal{T}_h)} \mathcal{Q}(K)$.

These heuristics are chosen so that $\mathcal{Q}(K) = 0$ if K is deteriorated (e.g. $\mathrm{vol}(K) = 0$ or if K nonconvex or degenerated to a triangle in the case of hypercubes) and that $\mathcal{Q}(\hat{S}_n) = \mathcal{Q}(\hat{Q}_n) = 1$ (hence the additional scaling factor $1/\sqrt[d]{\mathrm{vol}(S)_n}$ in (26)). To the best of the author's knowledge, there is no simple mesh quality heuristic based on geometric quantities for 3-hypercubes, as the faces will in general not be planar. This means that for 3-hypercubes, one would need to compute $|R_K^{-1}|_{1,\infty,K}$ directly.

The hypercube quality heuristic (25) has the important property that it tends to zero if an interior angle tends to zero or π, or if the length of an edge tends to zero. Note that $\gamma_{\min}(K) \to 0$ or $h_{\min} \to 0$ only imply $\rho(K) \to 0$ if K is an affine hypercube.

6.1 Software used

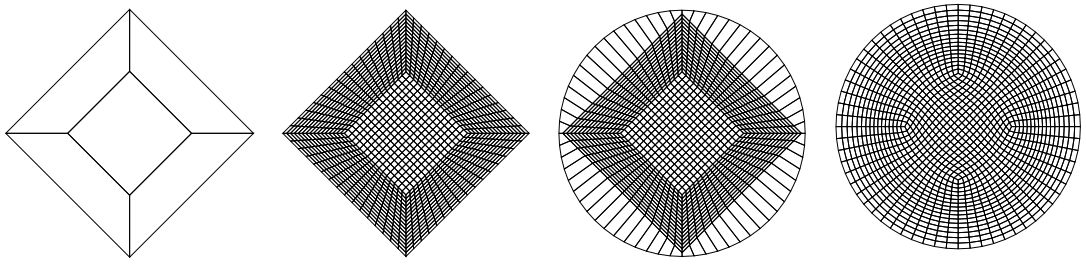
All computations in this chapter were done on compute servers at the Technische Universität Dortmund using the software FEAT3, which is part of the FEAT and FEATFLOW software family [33, 37, 48, 49]). It is a new C++ code designed to be used by researchers as well as in industry applications and features a very flexible solver structure and a great variety of finite elements including Argyris, Bogner-Fox-Schmit, conforming Lagrange, Crouzeix-Raviart, Rannacher-Turek and Zienkiewicz elements. The code is MPI parallel and GPU acceleration is available for a wide range of linear solvers, as well as a domain decomposition-based solver using the ScaRC architecture [35].

The nonlinear solvers used in this section are not implemented for GPUs, as there is no matrix that can be assembled in the main memory and then upload it to the GPU. Instead, an evaluation of the functional's gradient $\mathcal{F}_h'$ requires one gather operation per cell, which is detrimental for GPU performance. Although most solvers are also MPI parallel, some computations in this section were done in serial to be able to compare the results with solvers from `ALGLIB` [10], for which FEAT3 provides an interface.

6.2 Refinement of a unit circle mesh

In this case the aim is to have a simple geometric situation where an outer boundary is adjusted and the mesh is optimised afterwards. The same happens in every time step of an instationary problem involving a moving mesh, and will be used to study the convergence behaviour different solvers and preconditioners applied to a *degenerating family of meshes*, meaning it is no regular in the sense of Definition 9. Note that the quality of the resulting meshes is not discussed in detail (see [41, Chapter 5.3] for an in-depth analysis and comparison with the quadratic mesh quality functional based on the $\mathbf{D}(\Phi) : \mathbf{D}(\eta)$ bilinear form).

We consider the domain $\Omega = B_1(0) \subset \mathbb{R}^2$ with a polygonal approximation Ω_h and a mesh $\mathcal{T}_h$ of simplices or hypercubes of dimension 2.



(a) $\mathcal{T}_{h,0}$. (b) $\mathcal{T}_{h,4}$. (c) $\hat{\varphi}(\mathcal{T}_{h,4})$. (d) $\Phi^*(\mathcal{T}_{h,4})$.

Figure 5: Various meshes for the approximation of the unit circle, hypercubes.

We obtain this by using

$$\Omega_h := \{x \in \mathbb{R}^2 : \|x\|_1 \leq 1\},$$

with a mesh $\mathcal{T}_{h,0}$ which we then refined using regular refinement to obtain the mesh $\mathcal{T}_{h,l}$ which in turn defines the boundary $\partial\Omega_{h,l}$ and thus $\Omega_{h,l}$. In the following, the index l is omitted if this is

unambiguous or generic.

We now want to use this family of meshes to approximate Ω by imposing an appropriate displacement boundary condition through the mapping

$$\varphi : \mathbb{R}^2 \setminus \{0\} \to \partial\Omega : \varphi(x) = \frac{1}{\|x\|} x,$$

with which we can define the vertex mapping

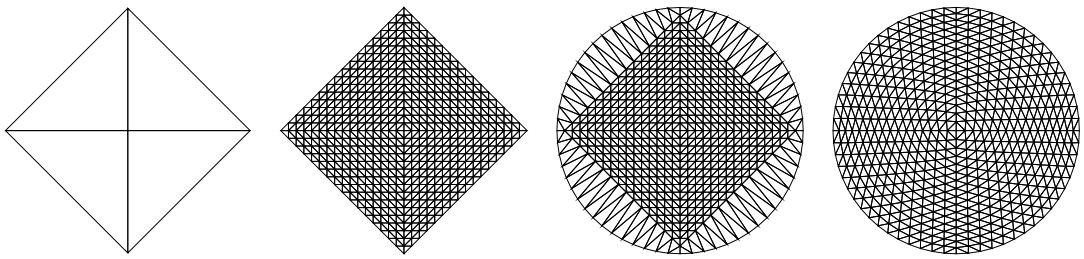
$$\hat{\varphi} : \mathcal{E}^0(\mathcal{T}_h) \to \mathbb{R}^2, \hat{\varphi}(v) = \begin{cases} \operatorname{tr} \varphi(v), & v \in \partial\mathcal{T}_h \\ v, & v \notin \partial\mathcal{T}_h(t_k) \end{cases}. \tag{28}$$

This can be interpreted as using the explicit knowledge of $\partial\Omega$ to adapt $\partial\Omega_{h,l}$ such that

$$\forall v \in \mathcal{E}^0(\mathcal{T}_{h,l}) : v \in \partial\Omega_{h,l} \Rightarrow v \in \partial\Omega,$$

which also arises in every time step in an instationary problem involving a moving mesh $\mathcal{T}_h(t)$.

The mapping $\hat{\varphi}$ also defines a member of $\mathcal{D}_h^{\phi,\partial\Omega_h}(\mathcal{T}_h, \Omega)$ as its values can be interpreted as the DoF of the appropriate discrete space. This finite element function will again be denoted by $\hat{\varphi}$ and used as the initial guess for all solvers. It also can be used to visualise the domain and the mesh after applying the boundary deformation.



(a) $\mathcal{T}_{h,0}$. (b) $\mathcal{T}_{h,4}$. (c) $\hat{\varphi}(\mathcal{T}_{h,4})$. (d) $\Phi^*(\mathcal{T}_{h,4})$.

Figure 6: Various domains for the approximation of the unit circle, simplices.

To create a degenerating sequence of meshes, only $\partial\mathcal{T}_{h,l}$ is adapted (meaning $\hat{\varphi}$ is only applied after the lth refinement), which leads to cells of very poor quality in $\hat{\varphi}(\mathcal{T}_{h,l})$, which could e.g. be the output of an external mesh generation tool.

Define the spaces

$$V_h := \left\{ v \in \mathcal{D}(\mathcal{T}_h) : v_{|\partial\hat{\Omega}_h} = \varphi \right\}, \quad W_h := \left\{ w \in \mathcal{D}(\mathcal{T}_h) : w_{|\partial\hat{\Omega}_h} = 0 \right\}.$$

For the mesh quality functional, we use the local functionals

$$\mathcal{L}_K(\Phi) = \alpha_1(\|\nabla R_K(\Phi)\|_F^2 - d)^2 + \alpha_3 \det(\nabla R_K(\Phi))^r + \frac{c(r)}{\left(\det(\nabla R_K(\Phi)) + \sqrt{\delta_r^2 + \det(\nabla R_K(\Phi))^2}\right)^r},$$

which is a regularisation of the example (19).

In this example, the parameters are $\alpha_1 = 1, \alpha_3 = 1, \delta_r = 1\text{e}{-8}, r = 1, c(r) = \sqrt{\delta_r^2 + 1} + \delta_r^2 + 1$.

Solver configuration The nonlinear solvers used were NLCG, lBFGS and preconditioned NLCG. They all used a mixed quadratic/cubic interpolating line search implementing the strong Wolfe conditions [39, Chapter 3.1].

The line search used $c_1 = 1\text{e}{-3}, c_2 = 0.3$ for the strong Wolfe conditions, a step length criterion of $\epsilon_s = 5\text{e}{-14}$ and was allowed a maximum of 20 iterations. The NLCG was allowed 10 subsequent iterations without the line search finding a point satisfying the strong Wolfe conditions before aborting.

For the preconditioners, the configurations for obtaining an approximation $\tilde{A}_h^{-1}$ are:

lBFGS	
	$\epsilon_r = 1\text{e}{-8}$
	$\epsilon_f = 0$
	lBFGS_dim = 10
	max_iter = 10000

(a) lBFGS configuration

NLCG	
	$\epsilon_r = 1\text{e}{-8}$
	$\epsilon_s = 2.2204\text{e}{-16}$
	$\epsilon_f = 0$
	stag_iter = 10
	max_iter = 10000
Update	Dai-Yuan-Hestenes-Stiefel

(b) NLCG configuration

Table 1: Different nonlinear solver configurations.

1. $\left(\tilde{A}_h^w\right)^{-1}$: Solve the linear system given by the operator A_h approximately by applying exactly one multigrid V-cycle from either Table 2a or Table 2b ("weak" preconditioner).
2. $\left(\tilde{A}_h^s\right)^{-1}$: Solve the linear system given by the operator A_h approximately by with PCG-MG from either Table 2a or Table 2b("strong" preconditioner).

PCG	$\epsilon_r = 1\text{e}{-8}$ max_iter = 1000
MG	cycle = V coarsest level = 1
Smoother	Richardson-Jacobi
Smoother iterations	4 pre, 4 post
Coarse grid solver	PCG-Jacobi
Jacobi	$\omega = 0.7$
PCG-Jacobi	$\epsilon_r = 1e-8$ max_iter = 1000

(a) Configuration 1

PCG	$\epsilon_r = 1\text{e}{-8}$ max_iter = 1000
MG	cycle = V coarsest level = 1
Smoother	CG
Smoother iterations	4 pre, 4 post
Coarse grid solver	PCG-Jacobi
Jacobi	$\omega = 0.5$
PCG-Jacobi	$\epsilon_r = 1e-8$ max_iter = 1000

(b) Configuration 2

Table 2: Different solver configurations for applying $(\tilde{A}_h^{\cdot,F})^{-1}$.

In all tables with iteration numbers, cases in which a solver stagnated have the letter s, while the cases in which the nonlinear solver did stop early because of the step length or functional value criterion have an asterisk in the first column. In all of these cases, the NLCG stopped because the step length criterion was satisfied. This is an indicator for the problem being too badly conditioned for the nonlinear solver to make any further progress and might be related to the regularity of the minimiser and the Euler-Lagrange equations (see Remark 3). One could set $\epsilon_s = 0$, but in numerical experiments this led to more iterations where the output of the line search does not satisfy the strong Wolfe conditions, leading to stagnation of the solver after 10 iterations.

It should be noted that in all tables, the number of lBFGS or NLCG iterations is given, which is the correct metric to assess the influence of a preconditioner, but it does not describe the overall numerical effort. This is more accurately measured by the number of functional evaluations, which can be very different in everly line search iteration and not directly influenced by the preconditioner.

Remark 11. *Since the minimiser $\Phi^* = \operatorname{argmin}_\Phi \mathcal{F}(\Phi)$ is not unique, preconditioning the solver might cause it to converge to a completely different solution.*

There are also the effects of the step length stopping criterion and the stagnation criterion, which cause the solver to stop before the criteria on the absolute or relative residual norm are met. These criteria are chosen based on numerical experience to stop the solver early when it is highly unlikely or very costly for it to make any further progress. This also means that preconditioning might cause the solver to make progress towards a different local minimiser which it might not reach due to these criteria, or the preconditioned iteration makes further progress towards a different solution instead of stopping early, requiring more iterations. The nonuniqueness of the solution makes measuring the effect of using different solvers and preconditioners much more difficult.

Simplex meshes For all solvers for applying the preconditioners $(\tilde{A}_h^{\cdot,\cdot})^{-1}$, the solver configuration from Table 2a was used.

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\frac{\|\mathcal{F}'(\mathcal{T}_\mathbf{h})\|_\mathbf{2}}{\|\mathcal{F}'(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))\|_\mathbf{2}}$
3	44	75	7.13e-1	7.21e-1	7.68e-1	8.49e-1	8.71e-9
4	103	147	5.46e-1	6.94e-1	7.48e-1	8.51e-1	9.42e-9
5	208	257	4.02e-1	6.76e-1	7.45e-1	8.52e-1	9.02e-9
6	400	453	2.90e-1	6.63e-1	7.49e-1	8.53e-1	9.76e-9
7	1081	1608	2.07e-1	6.53e-1	7.53e-1	8.53e-1	9.52e-9
8	3908	9229	1.47e-1	6.44e-1	7.56e-1	8.53e-1	9.86e-9
9s	1156	4602	1.04e-1	4.63e-1	7.57e-1	7.77e-1	6.36e-7
10*	388	748	7.37e-2	2.74e-1	7.59e-1	7.13e-1	6.66e-7
11s	359	797	5.21e-2	3.40e-1	7.59e-1	7.43e-1	6.38e-7

Table 3: NLCG, simplex meshes.

In Table 3 we can see the number of iterations approximately doubling for every level of refinement up to level seven. Level eight is the last level where the solver converges with regard to ϵ_r, but the number of iterations and evaluations quadrupled compared to level seven. As mentioned before, this indicates the ill-conditioning of the problem. Also, the ratio between iterations and functional evaluations becomes worse, as the line search becomes more difficult with further refinement.

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\frac{\|\mathcal{F}'(\mathcal{T}_\mathbf{h})\|_\mathbf{2}}{\|\mathcal{F}'(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))\|_\mathbf{2}}$
3	40	72	7.13e-1	7.21e-1	7.68e-1	8.49e-1	9.66e-9
4	94	127	5.46e-1	6.94e-1	7.48e-1	8.51e-1	8.11e-9
5	184	259	4.02e-1	6.76e-1	7.45e-1	8.52e-1	8.57e-9
6	377	719	2.90e-1	6.63e-1	7.49e-1	8.53e-1	9.93e-9
7	791	1613	2.07e-1	6.53e-1	7.53e-1	8.53e-1	1.00e-8
8	1915	4546	1.47e-1	6.44e-1	7.56e-1	8.53e-1	9.94e-9
9s	389	1924	1.04e-1	1.63e-1	7.57e-1	7.54e-1	2.85e-5
10*	507	3102	7.37e-2	2.08e-1	7.59e-1	7.31e-1	9.78e-7
11*	759	1331	5.21e-2	4.92e-1	7.59e-1	7.31e-1	3.20e-8

Table 4: lBFGS, simplex meshes.

The iteration and functional evaluation numbers when using lBFGS (see Table 4) still show the same doubling with each level of refinement up until level eight. From level nine on, the solver stops early due to the functional value improvement criterion (recall that $\epsilon_f = 0$, meaning subsequent iterates yielded the same functional value). On levels nine and ten, the stopping iterate results in meshes with very poor shape quality heuristic $\mathcal{Q}$, while on level eleven, the result is significantly better. Note how the number of functional evaluations per iteration increases with further refinement and drops again on level eleven. There is no real explanation for this except for the unpredictability of solution process of the nonlinear problem. Also note that even though BFGS is considered the most efficient quasi Newton method and offers superlinear convergence rates for strongly convex functional under certain assumptions [39, Chapter 6.4], the limited memory variant applied to this nonconvex functional actually behaves quite similar to unpreconditioned NLCG (Table 3) up to level seven. This indicates that using the Newton direction (21) (in the cases where it actually is a descent direction) e.g. by appropriately preconditioning a steepest descent method or NLCG will most likely not give decisively different convergence rates.

If we apply the preconditioner $(A_h^w)^{-1}$ (the results can be found in Table 5), the number of iterations only increases slightly with each level of refinement, up to level seven. Like in the unpreconditioned case (see Table 3), the results are different from level eight on, where the number of iterations increases sharply. On levels nine and ten, the solver stops due to the step length

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\frac{\|\mathcal{F}'(\mathcal{T}_\mathbf{h})\|_\mathbf{2}}{\|\mathcal{F}'(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))\|_\mathbf{2}}$
3	26	71	7.13e-1	7.21e-1	7.68e-1	8.49e-1	5.69e-9
4	33	61	5.46e-1	6.94e-1	7.48e-1	8.51e-1	5.72e-9
5	41	72	4.02e-1	6.76e-1	7.45e-1	8.52e-1	8.02e-9
6	47	76	2.90e-1	6.63e-1	7.49e-1	8.53e-1	9.97e-9
7	48	123	2.07e-1	6.53e-1	7.53e-1	8.53e-1	6.96e-9
8	255	2313	1.47e-1	6.45e-1	7.56e-1	8.53e-1	9.98e-9
9*	333	4090	1.04e-1	5.98e-1	7.57e-1	8.52e-1	9.85e-8
10*	503	6621	7.37e-2	6.43e-1	7.59e-1	8.48e-1	6.17e-8
11s	125	2095	5.21e-2	1.42e-2	7.59e-1	8.00e-1	2.98e-1

Table 5: NLCG-$\tilde{A}_h^w$, simplex meshes.

criterion, and the shape quality indicator $\mathcal{Q}(\Phi^*(\mathcal{T}_h))$ is higher than in the case of no preconditioning. On refinement level eleven, the solver stagnates without reaching a useful local minimum. Clearly, A_h^w has stopped being a good preconditioner, either because the approximation $(A_h^w)^{-1}$ obtained by applying a single multigrid V-cycle is no longer good enough, or because the approximation on the continuous level is not sufficient. We will see below that this is not the case, as using the "stronger" preconditioner $(A_h^F)^{-1}$ gives much better results. Note that the same increase in functional evaluations per NLCG iteration as before occurs.

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\frac{\|\mathcal{F}'(\mathcal{T}_\mathbf{h})\|_\mathbf{2}}{\|\mathcal{F}'(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))\|_\mathbf{2}}$
3	24	48	7.13e-1	7.21e-1	7.68e-1	8.49e-1	9.10e-9
4	30	61	5.46e-1	6.94e-1	7.48e-1	8.51e-1	7.76e-9
5	32	61	4.02e-1	6.76e-1	7.45e-1	8.52e-1	8.18e-9
6	36	66	2.90e-1	6.63e-1	7.49e-1	8.53e-1	6.97e-9
7	38	68	2.07e-1	6.53e-1	7.53e-1	8.53e-1	7.90e-9
8	41	90	1.47e-1	6.45e-1	7.56e-1	8.53e-1	8.27e-9
9	40	161	1.04e-1	6.30e-1	7.57e-1	8.53e-1	8.80e-9
10	79	724	7.37e-2	6.17e-1	7.59e-1	8.48e-1	8.25e-9
11	28	280	5.21e-2	5.24e-1	7.59e-1	8.41e-1	7.37e-9

Table 6: NLCG-$\tilde{A}_h^s$, simplex meshes.

Applying the strong preconditioner $\tilde{A}_h^s$ yields iteration number that increase only slightly with increasing refinement level (see Table 6). The solver converges with regard to the relative residual criterion for all levels, which is a very important improvement over the other preconditioners used so far. However, we see the number of functional evaluations increasing sharply from level nine on, meaning the line search requires more iterations, increasing the ratio of evaluations per iteration. This is another indicator for the systematic ill-conditioning of the system, where the preconditioner improves the search directions, but cannot make the finding of a step satisfying the strong Wolfe conditions any easier.

Notice how the resulting shape quality indicator $\mathcal{Q}(\Phi^*(\mathcal{T}_h))$ decreases with further refinement, even though the solver converged with regard to the relative residual norm. Apparently, this stopping criterion is not optimal achieving the best possible mesh quality. This also means that the performance of a solver or preconditioner cannot be evaluated based on iteration numbers alone, but the "quality" of the solutions found needs to be taken into account as well, both due to the nonuniqueness and the described effect.

Hypercube meshes The hypercube case will be discussed in less detail, as the results are qualitatively similar. For all solvers for applying the preconditioners $(\tilde{A}_h^\cdot)^{-1}$, the solver configuration from Table 2b was used because the configuration from Table 2a resulted in the multigrid V-cycle not reducing the norm of the defect of the system it was applied to.

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\frac{\|\mathcal{F}'(\mathcal{T}_\mathbf{h})\|_\mathbf{2}}{\|\mathcal{F}'(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))\|_\mathbf{2}}$
3	95	145	1.80e-1	4.22e-1	4.24e-1	5.70e-1	9.97e-9
4	190	239	1.86e-1	4.41e-1	4.22e-1	5.97e-1	7.19e-9
5	422	482	1.39e-1	4.52e-1	4.26e-1	6.12e-1	8.80e-9
6	717	781	7.18e-2	4.57e-1	4.29e-1	6.20e-1	9.91e-9
7	1209	1324	3.65e-2	4.60e-1	4.32e-1	6.24e-1	9.53e-9
8	1991	2109	1.84e-2	4.61e-1	4.33e-1	6.26e-1	9.64e-9
9	3256	3481	9.23e-3	4.65e-1	4.34e-1	6.36e-1	9.69e-9
10s	990	5559	4.62e-3	1.12e-1	4.35e-1	4.52e-1	7.12e-7
11*	1357	9936	2.31e-3	2.70e-2	4.35e-1	4.36e-1	1.14e-6

Table 7: NLCG, hypercube meshes.

Table 7 contains the results obtained by using the unpreconditioned NLCG solver. The main difference to the simplex case is that the number of iterations slightly less than doubles with every level of refinement only up to level nine, for up to which the solver converges with regard to the relative residual norm. Note that due to the nonuniqueness of the solution and the resulting highly unpredictable solver behaviour, this does not indicate any systematic advantages. Because of the high computation times of the unpreconditioned solver, refinement levels ten and eleven were done in parallel with 16 MPI processes. On level ten, the solver stagnates and on level eleven it stops due to the step length criterion, with a high ratio of functional evaluations per NLCG iteration, indicating the ill-conditioning of the problem.

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\frac{\|\mathcal{F}'(\mathcal{T}_\mathbf{h})\|_\mathbf{2}}{\|\mathcal{F}'(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))\|_\mathbf{2}}$
3	86	138	1.80e-1	4.22e-1	4.24e-1	5.70e-1	1.48e-8
4	167	238	1.86e-1	4.41e-1	4.22e-1	5.97e-1	8.63e-9
5	330	507	1.39e-1	4.52e-1	4.26e-1	6.12e-1	9.01e-9
6	564	932	7.18e-2	4.57e-1	4.29e-1	6.20e-1	9.85e-9
7	1337	3156	3.65e-2	4.60e-1	4.32e-1	6.24e-1	9.78e-9
8*	385	884	1.84e-2	4.00e-2	4.33e-1	6.12e-1	2.84e-6
9*	746	1225	9.23e-3	5.49e-2	4.34e-1	6.09e-1	3.00e-7
10*	567	1924	4.62e-3	1.82e-1	4.35e-1	5.30e-1	1.38e-7
11*	817	1457	2.31e-3	1.71e-1	4.35e-1	5.03e-1	5.59e-8

Table 8: lBFGS, hypercube meshes.

Using steepest descent preconditioned with lBFGS (Table 8) again gives qualitatively similar results as in the simplex case, with less iterations and functional evaluations required than with unpreconditioned NLCG up to refinement level six, with the same dependence on the refinement level. On level seven, more iterations compared to using NLCG are needed, and from level eight on the solver stops early without converging with regard to the relative residual norm. Clearly, the minimisation problem is such that the theoretical advantages of this solver over the unpreconditioned NLCG do not come into play.

When using the "weaker" preconditioner $\tilde{A}_h^w$ (the results are in Table 9), the number of iterations only slightly increases up to refinement level seven. Level eight appears to be the point where the preconditioner no longer gives the radical improvement of the lower levels, and the number of line search iterations needed in every NLCG iteration increases sharply. From level ten on, the solver stagnates with ten subsequent steps of steepest descent.

As in the simplex case, using the "stronger" preconditioner $\tilde{A}_h^s$ results in the solver being able to converge with regard to the relative residual norm for all levels of refinement (see Table 10). The number of iterations again only lightly depends on the refinement level. However, the interesting effect of the iteration numbers *decreasing* with further refinement can be observed, with only nine iterations for refinement level eleven. But the shape quality heuristic $\mathcal{Q}(\Phi^*(\mathcal{T}_h))$ shows

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\frac{\|\mathcal{F}'(\mathcal{T}_\mathbf{h})\|_\mathbf{2}}{\|\mathcal{F}'(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))\|_\mathbf{2}}$
3	59	136	1.80e-1	4.22e-1	4.24e-1	5.70e-1	1.30e-8
4	82	126	1.86e-1	4.41e-1	4.22e-1	5.97e-1	7.90e-9
5	102	150	1.39e-1	4.52e-1	4.26e-1	6.12e-1	7.97e-9
6	117	164	7.18e-2	4.57e-1	4.29e-1	6.20e-1	8.35e-9
7	105	154	3.65e-2	4.60e-1	4.32e-1	6.24e-1	9.96e-9
8	191	1450	1.84e-2	4.61e-1	4.33e-1	6.26e-1	9.32e-9
9	766	10050	9.23e-3	4.62e-1	4.34e-1	6.27e-1	9.36e-9
10s	95	1329	4.62e-3	1.05e-2	4.35e-1	4.67e-1	5.13e-2
11s	48	853	2.31e-3	3.01e-3	4.35e-1	4.47e-1	4.78e-1

Table 9: NLCG-$\tilde{A}_h^w$, hypercube meshes.

that the resulting mesh contains cells of much lower quality. Like in the simplex case, this indicates that stopping based on the relative residual with the same criterion on all levels is not sufficient for ensuring good mesh quality, but the effect is much stronger. For comparison purposes, the computation on level eleven was done with a lower stopping criterion of $\epsilon_r = 1\text{e}{-12}$ and can be found in the last row of Table 10. With the lower stopping criterion, the shape quality heuristic $\mathcal{Q}(\Phi^*(\mathcal{T}_h))$ is again in the same range as on the lower refinement levels, at the cost of many more iterations performed.

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\hat{\varphi}(\mathcal{T}_\mathbf{h}))$	$\mathcal{Q}_\mathbf{a}(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))$	$\frac{\|\mathcal{F}'(\mathcal{T}_\mathbf{h})\|_\mathbf{2}}{\|\mathcal{F}'(\mathbf{\Phi}^*(\mathcal{T}_\mathbf{h}))\|_\mathbf{2}}$
3	60	101	1.80e-1	4.22e-1	4.24e-1	5.70e-1	8.70e-9
4	78	121	1.86e-1	4.41e-1	4.22e-1	5.97e-1	9.72e-9
5	86	123	1.39e-1	4.52e-1	4.26e-1	6.12e-1	9.95e-9
6	91	137	7.18e-2	4.57e-1	4.29e-1	6.20e-1	8.02e-9
7	82	118	3.65e-2	4.60e-1	4.32e-1	6.24e-1	8.51e-9
8	66	101	1.84e-2	4.61e-1	4.33e-1	6.26e-1	8.84e-9
9	45	75	9.23e-3	4.61e-1	4.34e-1	6.27e-1	9.89e-9
10	29	63	4.62e-3	4.63e-1	4.35e-1	6.30e-1	9.77e-9
11	9	29	2.31e-3	9.69e-2	4.35e-1	5.54e-1	6.71e-9
11	196	620	2.31e-3	4.62e-1	4.35e-1	6.27e-1	9.71e-13

Table 10: NLCG-$\tilde{A}_h^s$, hypercube meshes.

For the example of the refinement of the unit circle mesh, the effect of the preconditioners is very strong, especially of the NLCG-$\tilde{A}_h^s$ variant. The use of these preconditioners also allows working with much finer meshes than otherwise possible. Going to these extremely fine meshes also revealed that the selection of the stopping criteria for the solver is more difficult than for linear problems, and that the relative residual norm might not be a good indicator for the quality of the corresponding solution. An application of this preconditioner to a moving nonconvex mesh can be found in [41, Chapter 7.2].

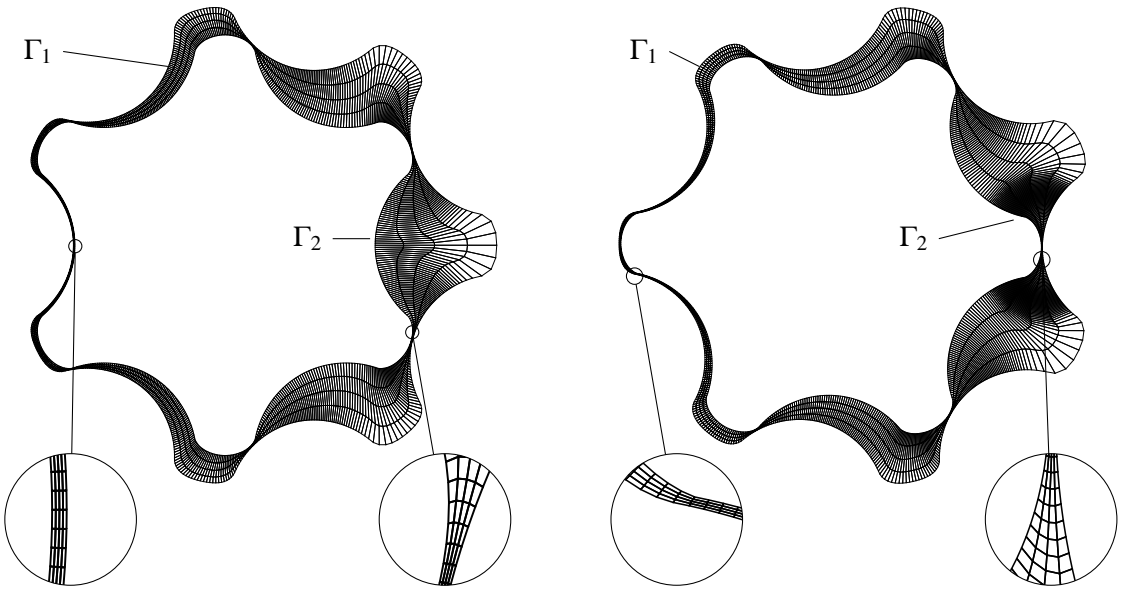
6.3 Application to a gerotor pump geometry

This example is inspired by the geometry of a gerotor micro gear pump. The design is well-known (see [32, 36] for some related patents), but still subject of development and optimisation. Two excentrically placed screws with n (inner) and $n+1$ teeth (outer) rotate at different angular velocities, which is a geometric requirement, as the screws must not touch. Their rotation compresses or expands the corresponding chambers, so leakage of fluid from high to low pressure chambers may result in an efficiency loss. This is hard to simulate numerically, as the complex moving domain is difficult to resolve.

Here, only $2d$ results for $n = 6$ are presented, without modelling of the inlet or outlet. The

aim is to show that the general purpose mesh optimisation method can be applied to a complex, moving domain on which the incompressible Navier-Stokes equations are then solved numerically. This is a setting where usually special purpose mesh movement techniques have to be used [18]. Lacking inlet and outlet, the flow will be (somewhat artificially) governed by the incompressibility constraint, which creates very high flow speeds.

One of the reasons why mesh optimisation in $2d$ is of practical importance are cases like this, where a $3d$ geometry can be expressed by an extruded $2d$ geometry. This is very easy to do for hypercube elements (see [41, Chapter 5.4.2]) but will not be discussed here.



(a) $t = 0$. (b) $t = 0.065$.

Figure 7: The mesh $\mathcal{T}_{h,1}$ for different times.

Denote by Γ_1 the inner boundary (meaning the outer boundary of the inner screw) and by Γ_2 the outer boundary (meaning the inner boundary of the outer screw). The domain of interest is the part between Γ_1 and Γ_2. This means that the gap width

$$\delta_g : \Omega \to \mathbb{R}, \quad \delta_g(x) = \sum_{i=1}^{2} \min\{\|x - x_i\|_2 : x_i \in \Gamma_i\}$$

greatly varies between

$$\begin{aligned} \delta_{\min} &= \min\{\|x_2 - x_1\|_2 : x_1 \in \Gamma_1, x_2 \in \Gamma_2\} = 0.02, \\ \delta_{\max} &= \max_{x \in \Omega_h} \{\min\{\|x_1 - x\|_2 + \|x_2 - x\|_2 : x_1 \in \Gamma_1, x_2 \in \Gamma_2\}\} = 1.15. \end{aligned}$$

This strong anisotropy means that the optimal scale of a cell K needs to be chosen according to its distance from Γ_1, Γ_2, a this changes according to the rotation and cells get compressed and expanded.

The goal is to allow for a full rotation of the outer screw, but since the screws rotate at different speeds, fixing the boundary vertices to their positions corresponding to the appropriate rigid body rotation by a displacement boundary condition (see Definition 6.3) quickly leads to mesh deterioration. Instead, a unilateral boundary condition of place (see Definition 6.4) needs to be used on one or even both boundaries. However, due to the strongly curved surfaces, mesh deformation methods not enforcing the orientation preserving property directly cannot use simple projection operators to enforce this boundary condition (see [41, Chapter 5.4.2] for a more in-depth discussion).

If a unilateral boundary condition of place is enforced at both boundaries, it means the cells can move freely throughout the mesh, although it might be advantageous to rule out rigid body rotations in the vertex movement (e.g. by enforcing a displacement boundary condition for one vertex on the outer boundary). Using unilateral boundary conditions of place on both boundaries helps to reduce the degree of anisotropy by allowing cells to dramatically change their sizes.

Denote by $\mathcal{T}_{h,0}$ the coarse mesh, which already captures the geometry and which is then refined l times (with boundary adaption similar to the vertex mapping (28)) for sufficient resolution of the incompressible Navier-Stokes equations. To reduce the numerical effort, the mesh optimisation is solved for $\mathcal{T}_{h,l_m}$, $0 \leq l_m < l$ and the solution is then prolongated to the finer mesh levels. However, the simple prolongation used does not take the orientation preserving property into account, so the choice of l_m depends on numerical experience. More sophisticated prolongation operators are possible, but the expected numerical effort is expected to offset any gain of solving the mesh optimisation on a coarser mesh in the first place.

Level	**DoF v**	**DoF p**	**DoF Φ**
0	7200	1080	2160
1	25920	3600	7200
2	97920	12960	25800
3	380160	48960	97920
4	1497600	190080	380160

Table 11: Degrees of freedom for different refinement levels.

The Navier-Stokes equations are discretised in space with conforming the LBB stable $\mathbb{Q}_2/\mathbb{Q}_1$ pair and in time with BDF2 using a pressure projection scheme [26] and a second order extrapolation in the nonlinear term. The mesh movement is treated by an ALE formulation. Because the flow and the mesh movement are very strong, the resulting system is very nonsymmetric and a ill-conditioned, so a strong solver/preconditioner combination is necessary (see Table 12). To sufficiently resolve the resulting vortices, $l \geq 4$ is needed, which makes $l_m = 2$ necessary to ensure sufficient mesh quality. The resulting number of DoF can be found in Table 11.

FGMRES(7)-MG	ϵ_r = 1e−8 max_iter = 100
MG	cycle = F coarsest level = 0
Smoother	FGMRES(32)-Jacobi$_1$
Smoother iterations	32 pre, 32 post
Jacobi$_1$	$\omega = 0.4$
Coarse grid solver	Richardson-Jacobi$_2$
Richardson	min_iter = 1 max_iter = 1
Jacobi$_2$	$\omega = 0.7$

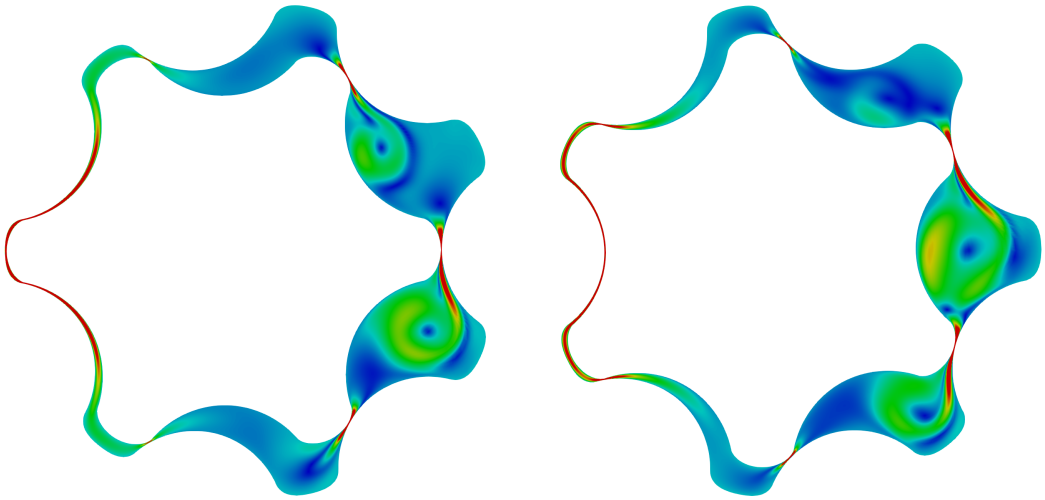
(a) Velocity solver

PCG-MG	ϵ_r = 1e−8 max_iter = 1000
MG	cycle = V coarsest level = 0
Smoother	Richardson-Jacobi$_1$
Smoother iterations	8 pre, 8 post
Jacobi$_1$	$\omega = 0.4$
Coarse grid solver	PCG-Jacobi$_2$
PCG-Jacobi$_2$	$\epsilon_r = 1e-8$ max_iter = 1000
Jacobi$_2$	$\omega = 0.5$

(b) Pressure solver

Table 12: Flow solver configuration.

The equations were solved on the time interval $[0, 1]$ with time step size δ_t = 1e−4, with the outer boundary performing exactly one full rotation. For the flow computations $Re = 10$ was used, but due to the nondimensionalisation of the domain's rotation, the angular velocity and the resulting flow speeds of $\|v_{\max}\|_2 \approx 650$, this corresponds to a real Reynolds number of approximately $Re = 6500$. The flow field for two time steps is shown in Figure 8, where the colour scale is chosen to account for the jet-like flow. While the flow can be resolved on coarser levels, $l = 4$ is necessary to resolve the vortices in the flow field, as instabilities can be observed otherwise. After a short startup phase, the flow becomes quasi-periodic and the pressure is uniform in each chamber, as it is governed by its compression or expansion.



(a) $t = 0.072$ (b) $t = 0.144$

Figure 8: The flow field with $\|v\|_2$ plotted from 0 (blue) to 60 (red).

7 Conclusion

The presented method is a useful generalisation of the work in [46] and can be used as a black box for optimising moving meshes. Because it directly enforces the orientation preserving property, it is very robust and allows the use of unilateral boundary conditions of place. Furthermore, it is formulated in the principal invariants of the deformation gradient and offers fine-grained control over the deformation of entities of all shape dimensions.

One theoretical shortcoming is that the minimiser that can be proven to exist might not be a solution of the Euler-Lagrange equations in the corresponding space (Remark 3). However, even if a mesh is not optimal in the sense defined by the energy functional, the resulting deformation is still orientation preserving, which is the critical property.

Solution techniques and a PDE-based preconditioner have been presented, although no theoretical bounds for the resulting condition numbers could be obtained. As the necessary but problematic stability property (A5) rules out the convexity of the resulting functional, it cannot be used in the preconditioner as highly efficient numerical methods are not available for this type of problem.

The preconditioner's performance was showcased in an example, showing iteration numbers only mildly depending on the mesh refinement level. As the minimiser of the energy functional is not unique, the PDE-based preconditioner even allowed finding minimisers resulting in better mesh quality. Furthermore, the solver did not stagnate as it did for the unpreconditioned case or when using an lBFGS preconditioner. The applicability of the method for solving the incompressible Navier-Stokes equations in a complex geometric situation was demonstrated. Using unilateral boundary conditions of place, the general purpose mesh optimisation method was able to obtain a stable geometry, for which previously special purpose methods were required.

Because of the method's robustness, it is also possible to use it as a basis for r-adaptivity [41] and alignment of meshes to implicit surfaces (see [9, 41]), which was not discussed in this work.

Acknowledgement

The financial support of DFG is gratefully acknowledged (TU 102/50-1).

References

[1] S. S. Antman, *Regular and singular problems for large elastic deformations of tubes, wedges, and cylinders*, Archive for Rational Mechanics and Analysis **83** (1983), no. 1, 1–52.

[2] T. A. Baer, R. A. Cairncross, R. R. Rao, P. A. Sackinger, and P. R. Schunk, *A finite element method for free surface flows of incompressible fluids in three dimensions. Part I. Boundary fitted mesh motion*, International Journal for Numerical Methods in Fluids **33** (2000), 375–403.

[3] J. M. Ball, *Convexity conditions and existence theorems in nonlinear elasticity*, Archive for Rational Mechanics and Analysis **63** (1976), no. 4, 337–403.

[4] E. Bänsch, *Simulation of instationary, incompressible flows*, Acta mathematica Universitatis Comenianae **67, no. 1** (1998), 101–114.

[5] ———, *Finite element discretization of the Navier-Stokes equations with a free capillay surface*, Numerische Mathematik **88** (2001), no. 2, 203–235.

[6] E. Bänsch, S. Basting, and R. Krahl, *Numerical simulation of two-phase flows with heat and mass transfer*, Discrete and Continuous Dynamical Systems **35** (2015), no. 6, 2325–2347.

[7] E. Bänsch, J. Paul, and A. Schmidt, *An ALE finite element method for a coupled Stefan problem and Navier-Stokes equations with free capillary surface*, International Journal for Numerical Methods in Fluids **71** (2013), no. 10, 1282–1296.

[8] S. Basting, *An interface fitted finite element method for multiphysics simulations*, Ph.D. Thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2016.

[9] S. Basting and M. Weismann, *A hybrid level set front tracking finite element approach for fluid–structure interaction and two-phase flow applications*, Journal of Computational Physics **255** (2013), 228 –244.

[10] S. Bochkanov, *ALGLIB*. www.alglib.net.

[11] D. Braess, *Finite elements: Theory, fast solvers, and applications in solid mechanics*, 3rd ed., Cambridge University Press, 2007.

[12] P. R. Brune, M. G. Knepley, B.F. Smith, and X. Tu, *Composing scalable nonlinear algebraic solvers*, SIAM Review **57** (2015), no. 4, 535–565.

[13] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, Studies in Mathematics and its Applications, Elsevier Science, 1978.

[14] ———, *Mathematical elasticity Vol. I: Three-dimensional elasticity*, Studies in Mathematics and its Applications, vol. 20, North-Holland Publishing Co., Amsterdam, 1988.

[15] P.G. Ciarlet and P.-A. Raviart, *Interpolation theory over curved elements, with applications to finite element methods*, Computer Methods in Applied Mechanics and Engineering **1** (1972), no. 2, 217 –249.

[16] Y. H. Dai and Y. Yuan, *A nonlinear conjugate gradient method with a strong global convergence property*, SIAM J. on Optimization **10** (May 1999), no. 1, 177–182.

[17] H. Damanik, J. Hron, A. Ouazzi, and S. Turek, *Monolithic Newton-multigrid solution techniques for incompressible nonlinear flow models*, International Journal for Numerical Methods in Fluids **71** (2013), no. 2, 208–222.

[18] H. Ding, X. J. Lu, and B. Jiang, *A CFD model for orbital gerotor motor*, IOP Conference Series: Earth and Environmental Science **15** (2012), no. 6, 062006.

[19] L. C. Evans, *Partial differential equations*, Graduate studies in mathematics, American Mathematical Society, 1998.

[20] R. Fletcher and C. M. Reeves, *Function minimization by conjugate gradients*, The Computer Journal **7** (1964), no. 2, 149–154, available at `http://comjnl.oxfordjournals.org/content/7/2/149.full.pdf+html`.

[21] M. Fossati, R. A. Khurram, and W.G. Habashi, *An ale mesh movement scheme for long-term in-flight ice accretion*, International Journal for Numerical Methods in Fluids **68** (2012), no. 8, 958–976.

[22] L. A. Freitag and P. M. Knupp, *Tetrahedral mesh improvement via optimization of the element condition number*, International Journal for Numerical Methods in Engineering **53** (2002), no. 6, 1377–1391.

[23] R. Glowinski and O. Pironneau, *Finite element methods for Navier-Stokes equations*, Annular Review of Fluid Mechanics **24** (1992), 167–204.

[24] C. Gross and R. Krause, *On the convergence of recursive trust-region methods for multiscale nonlinear optimization and applications to nonlinear mechanics*, SIAM Journal on Numerical Analysis **47** (2009), no. 4, 3044–3069, available at `http://dx.doi.org/10.1137/08071819X`.

[25] ———, *A recursive trust-region method for non-convex constrained minimization*, 18th international conference on domain decomposition methods, 2009, pp. 137–144.

[26] J. L. Guermond, P. Minev, and J. Shen, *An overview of projection methods for incompressible flows*, Computer Methods in Applied Mechanics and Engineering **195** (2006), no. 4447, 6011 –6045.

[27] M. E. Gurtin, *On the nonlinear theory of elasticity*, Contemporary Developments in Continuum Mechanics and Partial Differential Equations: Proceedings of the International Symposium on Continuum Mechanics and Partial Differential Equations, Rio de Janeiro, August 1977, 1978, pp. 237–253.

[28] W. W. Hager and H. Zhang, *A survey of nonlinear conjugate gradient methods*, Pacific journal of Optimization **2** (2006), no. 1, 35–58.

[29] M. R. Hanisch, *Multigrid Preconditioning For The Biharmonic Dirichlet Problem*, SIAM Journal on Numerical Analysis **30** (1993), 184–214.

[30] B. T. Helenbrook, *Mesh deformation using the biharmonic operator*, International Journal for Numerical Methods in Engineering **56** (2003), no. 7, 1007–1021.

[31] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards **49** (1952December), no. 6, 409–436.

[32] E. Hill, *Rotary-pump pressure control*, Hill Compressor & Pump Company, 1924. US Patent 1,486,836.

[33] M. Hinze, M. Köster, and S. Turek, *Space–Time Newton–Multigrid Strategies for Nonstationary Distributed and Boundary Flow Control Problems*, Trends in PDE Constrained Optimization, 2014.

[34] W. Huang and R. D. Russel, *Adaptive moving mesh methods*, Applied Mathematical Sciences, vol. 174, Springer, 2011.

[35] S. Kilian and S. Turek, *An example for parallel ScaRC and its application to the incompressible Navier–Stokes equations*, Technical Report 98–06, SFB 359, Universität Heidelberg, 1998.

[36] W. Kobald, *Zahnradmaschine (Pumpe oder Motor)*, Robert Bosch GmbH, 1977. DE Patent App. DE19,762,606,898.

[37] M. Köster, A. Ouazzi, F. Schieweck, S. Turek, and P. Zajac, *New robust nonconforming finite elements of higher order*, Applied Numerical Mathematics **62** (2012), no. 3, 166 –184.

[38] M. D. Mihajlović and D. J. Silvester, *Efficient parallel solvers for the biharmonic equation*, Parallel Computing **30** (2004), no. 1, 35 –55.

[39] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed., Springer, New York, 2006.

[40] W. Noll, *A general framework for problems in the statics of finite elasticity*, Contemporary Developments in Continuum Mechanics and Partial Differential Equations: Proceedings of the International Symposium on Continuum Mechanics and Partial Differential Equations, Rio de Janeiro, August 1977, 1978, pp. 363–387.

[41] J. Paul, *Nonlinear hyperelasticity-based mesh optimisation*, Ph.D. Thesis, TU Dortmund, 2016.

[42] E. Polak and G. Ribière, *Note sur la convergence de méthodes de directions conjuguées*, ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique **3** (1969), no. R1, 35–43.

[43] B. T. Polyak, *The conjugate gradient method in extremal problems*, USSR Computational Mathematics and Mathematical Physics **9** (1969), no. 4, 94 –112.

[44] P. Quintela-Estevez, *Critical points in the energy of hyperelastic materials*, ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique **24** (1990), no. 1, 103–132 (eng).

[45] T. Richter, *A monolithic geometric multigrid solver for fluid-structure interactions in ALE formulation*, International Journal for Numerical Methods in Engineering **104** (2015), no. 5, 372–390. nme.4943.

[46] M. Rumpf, *A variational approach to optimal meshes*, Numerische Mathematik **72** (1996), 523–540.

[47] G. Strang and G. J. Fix, *An analysis of the finite element method*, Wellesley-Cambridge Press, 1988.

[48] S. Turek, *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*, Lecture Notes in Computational Science and Engineering, Springer-Verlag, 1999.

[49] S. Turek, D. Göddeke, C. Becker, S. Buijssen, and H. Wobker, *FEAST – Realisation of hardware–oriented Numerics for HPC simulations with Finite Elements*, Concurrency and Computation: Practice and Experience **6** (2010May), 2247–2265. Special Issue Proceedings of ISC 2008. doi:10.1002/cpe.1584.

[50] S. Turek and D. Wan, *Fictious boundary and moving mesh methods for the numerical simulation of rigid pariculate flows*, Journal of Computational Physics **222** (2007), 28–56.

[51] T. Wick, *Fluid-structure interactions using different mesh motion techniques*, Computers & Structures **89** (2011), no. 13–14, 1456 –1467.